Протокол HTTP

Протоколы Интернет прикладного уровня

- DNS
- HTTP
- HTTPS ;
- FTP (File Transfer Protocol - RFC 959)
- Telnet (TELecommunication NETwork - RFC 854)
- SSH (Secure Shell - RFC 4251);
- POP3 ;
- IMAP ;
- SMTP ;
- LDAP ;
- XMPP (Jabber)
- SNMP

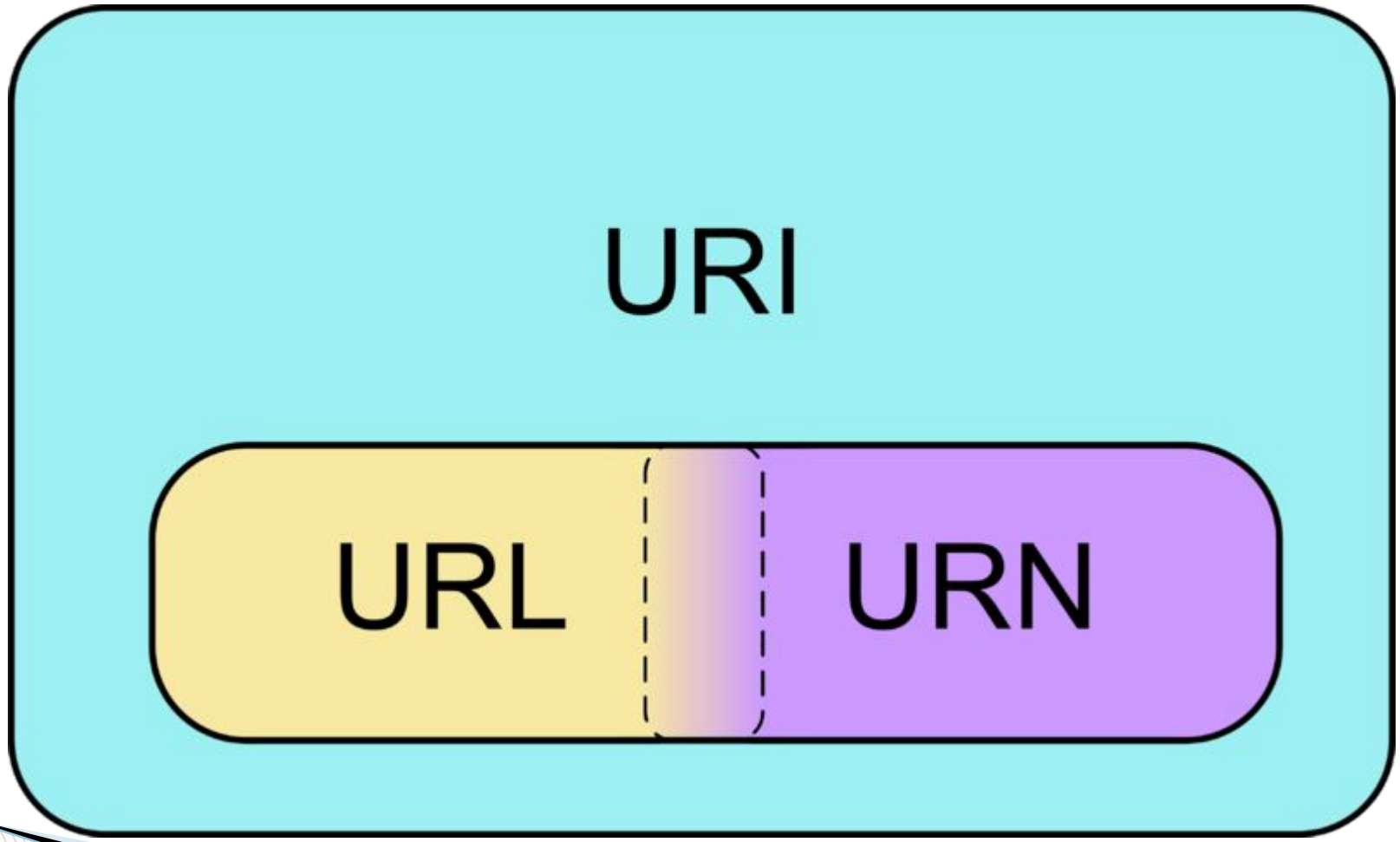
Повторение

- Толстый клиент
- Тонкий клиент

Протокол HTTP

- HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного уровня для передачи гипертекста.
- Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (англ. Uniform Resource Identifier) в запросе клиента.

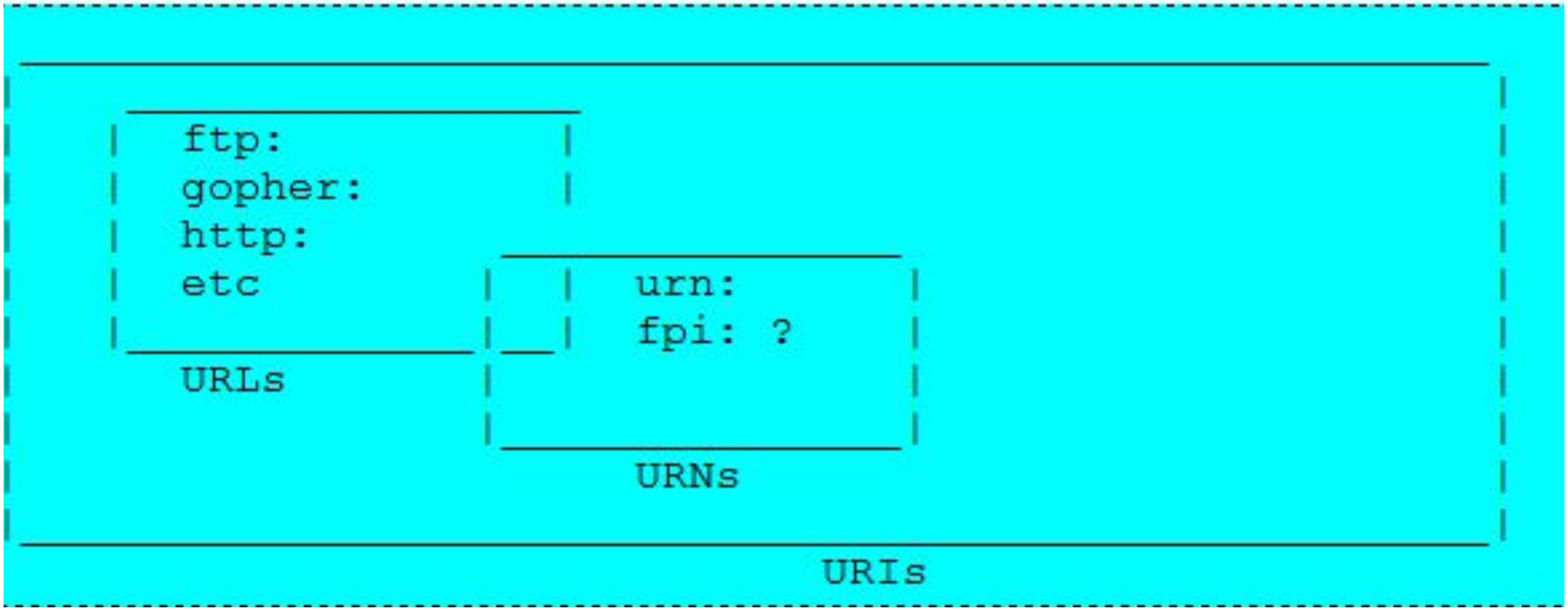
URI



Запросы RFC

- **URI**: запрос RFC1630, выпущенный в июне 1994 г., был назван "Универсальные идентификаторы ресурсов в WWW: единый синтаксис для выражения имен и адресов объектов в сети, используемый во Всемирной сети Интернет;
- **URL**: запрос RFC1738, выпущенный в декабре 1994 г., был назван "Унифицированные указатели информационных ресурсов" (Uniform Resource Locators)
- **URN**: запрос RFC1737, выпущенный в декабре 1994 г., был назван "Функциональные требования для унифицированных имен ресурсов" (Functional Requirements for Uniform Resource Names)

Общая схема формирования адресов:



<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>

Примеры

- Допустим, файл с именем "internet.zip" лежит на FTP-сервере ftp.ict.nsc.ru в директории /pub/winsite/www/. Тогда **URL адрес** этого файла будет выглядеть так: file://ftp.ict.nsc.ru/pub/winsite/www/internet.zip
- URL – строка, содержащая протокол + имя хоста http://web-ru.net
- URN – строка, содержащая имя папки или файла

История развития

- HTTP/0.9 **HTTP был предложен в марте 1991 года Тимом Бернерсом-Ли**
- HTTP/1.0 **В мае 1996 года**
- HTTP/1.1 **Текущая версия протокола, принята в июне 1999 года**

Преимущества

- Простота
- Расширяемость
- Распространённость

Все программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:

- **Серверы** - поставщики услуг хранения и обработки информации (обработка запросов).
- **Клиенты** - конечные потребители услуг сервера (отправка запросов).
- **Прокси-серверы** для поддержки работы транспортных служб

Четыре этапа транзакций HTTP

1. Перед тем как клиент и сервер смогут обмениваться данными, им необходимо сперва установить соединение. **В Интернет это делается с использованием TCP/IP.**
2. Далее клиент запрашивает данные у сервера
3. Сервер отвечает ему и передаёт необходимую информацию. **Клиенты и серверы используют HTTP для выполнения таких операций.**
4. Соединение устанавливается на время только одной транзакции и далее закрывается сервером по её окончании.

Шаг 1: Устанавливаем соединение

File Transfer Protocol (FTP)	21
TELNET Protocol	23
Simple Mail Transfer Protocol (SMTP)	25
Trivial File Transfer Protocol	69
Gopher Protocol	70
Finger Protocol	79
HTTP Protocol	80

Шаг 2: Запрос клиента

GET URI

— для версии протокола 0.9.

Метод URI HTTP/Версия

— для остальных версий.

Шаг 3: Ответ сервера

Формат ответ сервера

Версия_протокола	Код_ответа	Пояснительное_сообщение
HTTP/1.0	200	OK

Формат ответ сервера

Версия_протокола	Код_ответа	Пояснительное_сообщение
HTTP/1.0	200	OK

Шаг 4: Сервер разрывает соединение

Структура протокола

- Стартовая строка (англ. Starting line) — определяет тип сообщения;
- Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
- Тело сообщения (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Строка запроса

- GET URI — для версии протокола 0.9.

  Метод URI HTTP/Версия — для остальных версий.

  Метод (англ. Method) — название запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод GET, список запросов для версии 1.1 представлен ниже.
- URI определяет путь к запрашиваемому документу.

Стартовая строка ответа

- HTTP/Версия КодСостояния Пояснение
  - Версия — пара разделённых точкой арабских цифр как в запросе.
  - КодСостояния (англ. Status Code) — три арабские цифры. По коду статуса определяется дальнейшее содержимое сообщения и поведение клиента.
  - Пояснение (англ. Reason Phrase) — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.
- В случае успешного выполнения запроса сервер ответит строкой: HTTP/1.0 200 Ok

Код состояния HTTP ответа сервера

- 1 - специальный класс сообщений, называемых информационными. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса.
- 2 - успешная обработка запроса клиента.
- 3 - перенаправление запроса.
- 4 - ошибка клиента. Как правило, код ответа, начинающийся с цифры 4, возвращается в том случае, если в запросе клиента встретилась синтаксическая ошибка.
- 5 - ошибка сервера. По тем или иным причинам сервер не в состоянии выполнить запрос.

Классы кодов ответа сервера

Код	Расшифровка	Интерпретация
100	Continue	Часть запроса принята, и сервер ожидает от клиента продолжения запроса
200	OK	Запрос успешно обработан, и в ответе клиента передаются данные, указанные в запросе
201	Created	В результате обработки запроса был создан новый ресурс
202	Accepted	Запрос принят сервером, но обработка его не окончена. Данный код ответа не гарантирует, что запрос будет обработан без ошибок.
206	Partial Content	Сервер возвращает часть ресурса в ответ на запрос, содержавший поле заголовка Range
301	Multiple Choice	Запрос указывает более чем на один ресурс. В теле ответа могут содержаться указания на то, как правильно идентифицировать запрашиваемый ресурс
302	Moved Permanently	Затребованный ресурс больше не располагается на сервере
302	Moved *Temporarily*	Затребованный ресурс временно изменил свой адрес
400	Bad Request	В запросе клиента обнаружена синтаксическая ошибка
403	Forbidden	Имеющийся на сервере ресурс недоступен для данного пользователя
404	Not Found	Ресурс, указанный клиентом, на сервере отсутствует
405	Method Not Allowed	Сервер не поддерживает метод, указанный в запросе
500	Internal *Server Error*	Один из компонентов сервера работает некорректно
501	Not Implemented	Функциональных возможностей сервера недостаточно, чтобы выполнить запрос клиента
503	Service *Unavailable*	Служба временно недоступна
505	HTTP Version not Supported	Версия HTTP, указанная в запросе, не поддерживается сервером

Поля заголовка ответа веб-сервера

Имя поля	Описание содержимого
Server	Имя и номер версии сервера
Age	Время в секундах, прошедшее с момента создания ресурса
Allow	Список методов, допустимых для данного ресурса
Content-Language	Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс
Content-Type	MIME-тип данных, содержащихся в теле ответа сервера
Content-Length	Число символов, содержащихся в теле ответа сервера
Last-Modified	Дата и время последнего изменения ресурса
Date	Дата и время, определяющие момент генерации ответа
Expires	Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей
Location	В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса
Cache-Control	Директивы управления кэшированием. Например, no - cache означает, что данные не должны кэшироваться

Пример ответа сервера на запрос клиента

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
X-Powered-By: ASP.NET
Date: Mon, 20 OCT 2008 11:25:56 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sat, 18 Oct 2008 15:05:44 GMT
ETag: "b66a667f948c92:8a5"
Content-Length: 426
```

Методы запросов протокола HTTP

- **Метод HTTP (англ. HTTP Method)** — последовательность из любых символов кроме управляющих и разделителей, указывающая на основную операцию над ресурсом.
- Обычно метод представляет собой короткое английское слово записанное заглавными буквами.

HTTP методы

- Основная операция, которую необходимо выполнить над ресурсом. В названии могут использоваться любые символы, кроме управляющих последовательностей и разделителей, как правило это короткое слово на английском языке. Имена методов HTTP зависимы от регистра.
- Любой веб сервер обязан работать, по крайней мере с двумя методами GET и HEAD. Если сервер не смог определить метод, указанный в заголовке запроса клиента, он должен вернуть код статуса 501 (Not Implemented), если-же метод серверу известен, но неприменим к данному ресурсу, будет возвращен код статуса 405 (Method Not Allowed).

- Название метода чувствительно к регистру. Если метод серверу неизвестен, от отвечает ошибкой 501 (Method not implemented).
- Если серверу метод известен, но он не применим к конкретному ресурсу, то возвращается сообщение с кодом 405.

Методы обработки запросов HTTP

HTTP методы

- **Метод OPTIONS**
- **Метод GET**
- **Метод HEAD**
- **Метод POST**
- **Метод PUT**
- **Метод PATCH**
- **Метод DELETE**
- **Метод TRACE**
- **Метод LINK**
- **Метод UNLINK**

Метод OPTIONS

- Данный метод используется **для выяснения поддерживаемых веб-сервером возможностей или параметров соединения с конкретным ресурсом**.
- Сервер включает в ответный запрос заголовок *Allow*, со списком поддерживаемых методов и возможно информацию о поддерживаемых расширениях.
- Тело запроса клиента, содержит информацию об интересующих его данных, но на данном этапе формат тела и порядок работы с ним, не определен, пока, сервер должен его игнорировать.
- С ответным запросом сервера, происходит аналогичная ситуация.

Метод GET

- Метод *GET*, **применяется для для запроса содержимого указанного ресурса**. Также с помощью *GET*, может быть инициирован некий процесс, при этом, в тело ответа, включается информация о ходе выполнения инициированного запросом действия.
- Параметры для выполнения запроса, передаются в *URI* запрашиваемого ресурса, после символа "**?**". Запрос в таком случае выглядит примерно так: *GET /some/resource?param1=val1&param2=val2 HTTP/1.1*.
- Как установлено в стандарте *HTTP*, запросы методом *GET*, являются идемпотентными, то есть, повторная отправка одного и того-же запроса, методом *GET*, должна приводить к одному и тому-же результату, в случае, если сам ресурс, в промежутках между запросами, изменен не был, что позволяет кэшировать результаты, выдаваемые на запрос методом *GET*.

Метод HEAD

- Данный метод, аналогичен методу *GET*, с той лишь разницей, что сервер не отправляет тело ответа. Метод *HEAD*, как правило **используется для получения метаданных ресурса, проверки *URL*** (есть-ли указанный ресурс на самом деле) и для выяснения факта изменения ресурса с момента последнего обращения к нему.
- Заголовки ответа могут быть закэшированы, при несоответствии метаданных и информации в кэше, копия ресурса помечается как устаревшая.

Метод POST

- Метод *POST*, **используется для передачи пользовательских данных на сервер, указанному ресурсу**. Примером может послужить *HTML* форма с указанным атрибутом *Method="POST"*, для отправки комментария к статье. После заполнения необходимых полей формы, пользователь нажимаем кнопку "Отправить" и данные, методом *POST*, передаются серверному сценарию, который в свою очередь выводит их на странице комментариев. Таким-же образом, с помощью метода *POST*, можно передавать файлы.
- В отличии от *GET*, метод *POST*, не является идемпотентным, то есть неоднократное повторение запроса *POST*, может выдавать разные результаты. В нашем случае, будет появляться новая копия комментария при каждом запросе.

Метод PUT

- Используется **для загрузки данных запроса на указанный *URI*.** В случае отсутствия ресурса по указанному в заголовке *URI*, сервер создает его и возвращает код статуса *201 (Created)*, если ресурс присутствовал и был изменен в результате запроса *PUT*, выдается код статуса *200 (Ok)* или *204 (No Content)*. Если какой-то из переданных серверу заголовков *Content-**, не опознан или не может быть использован в данной ситуации, сервер возвращает статус ошибки *501 (Not Implemented)*.
- Главное различие методов *PUT* и *POST* в том, что при методе *POST*, предполагается, что по указанному *URI*, будет производиться обработка, передаваемых клиентом данных, а при методе *PUT*, клиент подразумевает, что загружаемые данные уже соответствуют ресурсу, расположенному по данному *URI*.
- Ответы сервера при методе *PUT* не кэшируются.

- **Метод PATCH –** работает аналогично методу *PUT*, но применяется только к определенному фрагменту ресурса.
- **Метод DELETE –** удаляет ресурс, расположенный по заданному URI.
- **Метод TRACE –** при запросе методом TRACE, клиент может увидеть, какие изменения были сделаны в запросе, промежуточными серверами.
- **Метод LINK –** связывает указанный ресурс с другим ресурсом.
- **Метод UNLINK –** снимает привязку одного ресурса к другому.

Пример диалога по протоколу HTTP

- Запрос:
  GET /wiki/HTTP HTTP/1.1
  Host: ru.wikipedia.org
  User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
  Accept: text/html
  Connection: close
- Ответ:
  HTTP/1.0 200 OK
  Server: nginx/0.6.31
  Content-Language: ru
  Content-Type: text/html; charset=utf-8
  Content-Length: 1234
  Connection: close
  <содержимое запрошенной страницы>

Обеспечение безопасности передачи данных HTTP

- **SSL** (Secure Sockets Layer) - криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет.

3 типа аутентификации при клиент-серверных взаимодействиях

- **Basic**
- **Digest**
- **Integrated**.

Basic

- Базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов.
- Пароль при этом не шифруется.
- Для данного типа аутентификации использование SSL является обязательным.

Digest

- Дайджест-аутентификация, при которой пароль пользователя передается в хешированном виде.
- По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только хеш от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование SSL является обязательным.

Integrated

- Интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов NTLM или Kerberos.
- Этот тип аутентификации защищен от перехвата удостоверений пользователей, поэтому для него не требуется протокол SSL.
- Только при использовании данного типа аутентификации можно работать по схеме http, во всех остальных случаях необходимо использовать схему https.

ДЗ, дать определение

- **Cookie**
- **Абсолютные и относительные URL**