Веб-технологии

Предметом данного курса являются технологии глобальной сети World Wide Web (сокращенно WWW или просто Web). На русском языке распространенным вариантом является название "Веб".

В частности, в рамках курса будут рассмотрены такие вопросы как:

1. Структура и принципы Веб (базовые понятия, архитектура, стандарты и протоколы);
2. Технологии сети Веб (языки разметки и программирования веб-страниц, инструменты разработки и управления веб-контента и приложений для Веб, средства интеграции веб-контента и приложений в Веб).

Сеть Веб представляет собой глобальное информационное пространство, основанное на физической инфраструктуре Интернета и протоколе передачи данных HTTP. Зачастую, говоря об Интернете, подразумевают именно сеть Веб.

Что такое Интернет?

Поскольку физической основой сети Веб является Интернет, то для более глубокого понимания многих вопросов данного курса потребуется кратко ознакомиться со структурой и протоколами Интернета.

Что же такое Интернет?

По сути, это самая большая в мире сеть, не имеющая единого центра управления, но работающая по единым правилам и предоставляющая своим пользователям единый набор услуг. Интернет можно рассматривать как "сеть сетей", каждая из которых управляется независимым оператором – поставщиком услуг Интернета (ISP, Internet Service Provider).

С точки зрения пользователей Интернет представляет собой набор информационных ресурсов, рассредоточенных по различным сетям, включая ISP-сети, корпоративные сети, сети и отдельные компьютеры домашних пользователей. Каждый отдельный компьютер в данной сети называется *хостом* (от английского термина host).

Сегодняшний Интернет обязан своему появлению объединенной сети ARPANET, которая начиналась как скромный эксперимент в новой тогда технологии коммутации пакетов (табл. 1.1). Сеть ARPANET была развернута в 1969 г. и состояла поначалу всего из четырех узлов с коммутацией пакетов, используемых для взаимодействия горстки хостов и терминалов. Первые линии связи, соединявшие узлы, работали на скорости всего 50 Кбит/с. Сеть ARPANET финансировалась управлением перспективного планирования научно-исследовательских работ ARPA (Advanced Research Projects Agency) министерства обороны США и предназначалась для изучения технологии и протоколов коммутации пакетов, которые могли бы использоваться для кооперативных распределенных вычислений.

Таблица 1.1. Хронология развития Интернета (с 1966 по 2000 г.)

Год	Событие
1966	Эксперимент с коммутацией пакетов управления ARPA
1969	Первые работоспособные узлы сети ARPANET
1972	Изобретение распределенной электронной почты
1973	Первые компьютеры, подключенные к сети ARPANET за пределами США
1975	Сеть ARPANET передана в ведение управления связи министерства обороны США

1980	Начинаются эксперименты с TCP/IP
1981	Каждые 20 дней к сети добавляется новый хост
1983	Завершен переход на TCP/IP
1986	Создана магистраль NSFnet
1990	Сеть ARPANET прекратила существование
1991	Появление Gopher
1991	Изобретение Всемирной паутины. Выпущена система PGP. Появление Mosaic
1995	Приватизация магистрали Интернета
1996	Построена магистраль OC-3 (155 Мбит/с)
1998	Число зарегистрированных доменных имен превысило 2 млн.
2000	Количество индексируемых веб-страниц превысило 1 млрд.

На рисунке 1.1 представлен график, показывающий динамику роста числа хостов (как формально зарегистрированных и так активно функционирующих).

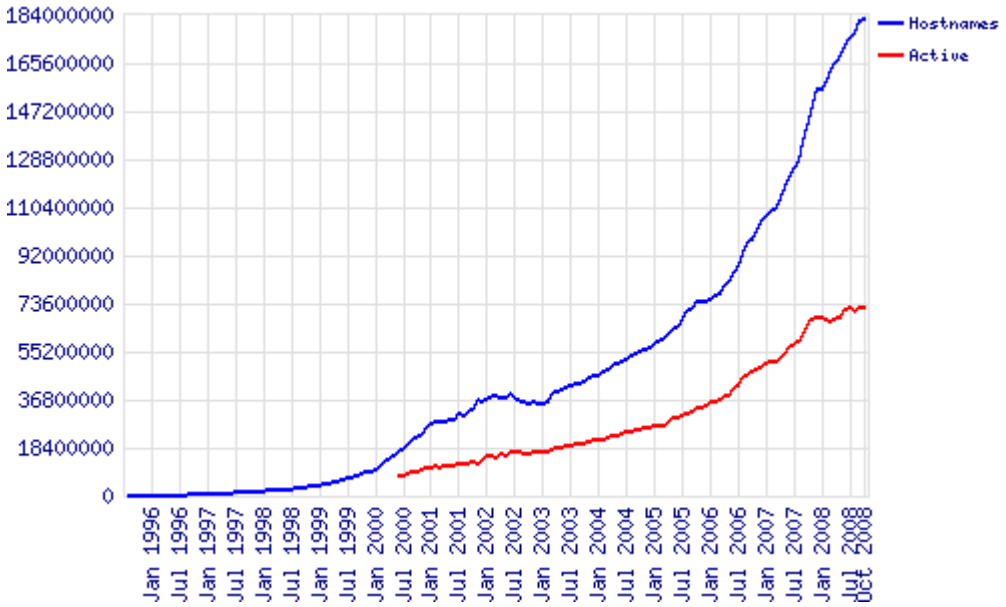

Рис. 1.1. Динамика роста числа хостов в Интернет (взято с сайта www.netcraft.com). Возможно ли централизованное управление в такой глобальной сети? Ответ на данный вопрос будет отрицательным, поскольку, во-первых, данная сеть является транснациональной и, во-вторых, в силу исторических предпосылок ее формирования.

Тем не менее, в Интернете могут проявляться опосредованные формы централизации в форме единой технической политики, согласованном наборе технических стандартов, назначении имен и адресов компьютеров и сетей, входящих в Интернет.

То есть Интернет является децентрализованной сетью, что имеет свои достоинства и недостатки.

1. Достоинства:
    - Легкость наращивания Интернета путем заключения соглашения между двумя ISP.
2. Недостатки:

- Сложность модернизации технологий и услуг Интернета, поскольку требуются согласованные усилия всех поставщиков услуг.
- Невысокая надежность услуг Интернета.
- Ответственность за работоспособность отдельных сегментов этой сети возлагается на поставщиков услуг Интернета.

Существуют различные типы поставщиков услуг Интернета:

- просто поставщик услуг Интернета выполняет транспортную функцию для конечных пользователей – передачу их трафика в сети других поставщиков услуг Интернета;
- поставщик интернет-контента имеет собственные информационно-справочные ресурсы, предоставляя их содержание в виде веб-сайтов;
- поставщик услуг хостинга предоставляет свои помещения, каналы связи и серверы для размещения внешнего контента;
- поставщик услуг по доставке контента занимается только доставкой контента в многочисленные точки доступа с целью повышения скорости доступа пользователей к информации;
- поставщик услуг по поддержке приложений предоставляет клиентам доступ к крупным универсальным программным продуктам, например SAP R3;
- поставщик биллинговых услуг обеспечивает оплату счетов по Интернету;

О роли стандартизации в Интернет

Как следует из всего вышеизложенного, Интернет является очень сложной сетью, и соответственно такой же сложной является задача организации взаимодействия между устройствами сети. Для решения такого рода задач используется *декомпозиция*, т.е. разбиение сложной задачи на несколько более простых задач-модулей. Одной из концепций, реализующих декомпозицию, является многоуровневый подход. Такой подход дает возможность проводить разработку, тестирование и модификацию каждого отдельного уровня независимо от других уровней. *Иерархическая декомпозиция* позволяет, перемещаясь в направлении от более низких к более высоким уровням переходить к более простому представлению решаемой задачи.

Специфика многоуровневого представления сетевого взаимодействия состоит в том, что в процессе обмена сообщениями участвуют как минимум две стороны, для которых необходимо обеспечить согласованную работу двух иерархий аппаратно-программных средств. Каждый из уровней должен поддерживать *интерфейс* с выше- и нижележащими уровнями собственной иерархии средств и интерфейс со средствами взаимодействия другой стороны на том же уровне иерархии. Данный тип интерфейса называется *протоколом* (см. рисунок 1.2).

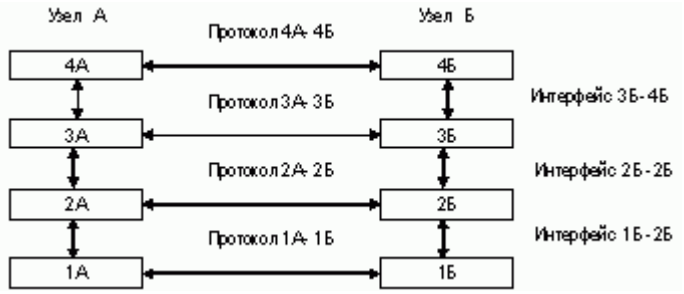

Рис. 1.2. Организация взаимодействия между уровнями иерархии при иерархической декомпозиции в сети Интернет
Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называется *стеком протоколов*.

В начале 80-х годов международные организации по стандартизации ISO (International Organization for Standardization), ITU (International Telecommunications Union) и другие разработали стандартную модель взаимодействия открытых систем OSI (Open System Interconnection). Назначение данной модели состоит в обобщенном представлении средств сетевого взаимодействия. Ее также можно рассматривать в качестве универсального языка сетевых специалистов (*справочной модели*).

Поскольку сеть – это соединение разнородного оборудования, актуальной является проблема совместимости, что в свою очередь, требует согласования всеми производителями общепринятых стандартов. *Открытой* является система, построенная в соответствии с *открытыми спецификациями*.

Спецификация представляет собой формализованное описание аппаратных (программных) компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, особых характеристик. Под *открытыми спецификациями* понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами. Использование открытых спецификаций при разработке систем позволяет третьим сторонам разрабатывать для этих систем аппаратно-программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Если две сети построены с соблюдением принципов открытости, это дает следующие преимущества:

- Возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся стандарта;
- Безболезненная замена отдельных компонентов сети другими, более совершенными;
- Легкость сопряжения одной сети с другой.

В рамках модели OSI средства взаимодействия делятся на семь уровней: прикладной, представления, сеансовый, транспортный, сетевой, канальный и физический. В распоряжение программистов предоставляется прикладной программный интерфейс, позволяющий обращаться с запросами к самому верхнему уровню, а именно, - уровню приложений.

Сеть Интернет строилась в полном соответствии с принципами открытых систем. В разработке стандартов этой сети принимали участие тысячи специалистов-пользователей сети из вузов, научных организаций и компаний. Результат работы по стандартизации воплощается в документах *RFC*.

RFC (англ. Request for Comments) — документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и Стандарты, широко применяемые во Всемирной сети. В настоящее время первичной публикацией документов *RFC* занимается *IETF* под эгидой открытой организации Общество Интернета (*ISOC*). Правами на *RFC* обладает именно Общество Интернет. Формат *RFC* появился в 1969 г. при обсуждении проекта *ARPANET*. Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, но уже с декабря 1969 г., когда заработали первые сегменты ARPANET, документы начали распространяться в электронном виде. В таблице 1.2 приведены некоторые из наиболее известных документов RFC.

Таблица 1.2. Примеры популярных RFC-документов.

Номер RFC	Тема
RFC 768	UDP
RFC 791	IP
RFC 793	TCP
RFC 822	Формат электронной почты, заменен RFC 2822
RFC 959	FTP
RFC 1034	DNS — концепция
RFC 1035	DNS — внедрение

RFC 1591	Структура доменных имен
RFC 1738	URL
RFC 1939	Протокол POP версии 3 (POP3)
RFC 2026	Процесс стандартизации в Интернете
RFC 2045	MIME
RFC 2231	Кодировка символов
RFC 2616	HTTP
RFC 2822	Формат электронной почты
RFC 3501	IMAP версии 4 издание 1 (IMAP4rev1)

Основным организационным подразделением, координирующим работу по стандартизации Интернет, является *ISOC* (Internet Society), объединяющее порядка 100 тысяч участников, которые занимаются различными аспектами развития данной сети. *ISOC* курирует работу *IAB* (Internet Architecture Board), включающую две группы:

- *IRTF* (Internet Research Task Force). Координирует долгосрочные исследовательские проекты, относящиеся к TCP/IP;
- *IETF* (Internet Engineering Task Force). Инженерная группа, определяющая спецификации для последующих стандартов Интернет.

Разработкой стандартов для сети Веб, начиная с 1994 года, занимается Консорциум W3C (World Wide Web Consortium), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли.

Консорциум W3C — организация, разрабатывающая и внедряющая технологические стандарты для Интернета и WWW. Миссия W3C формулируется следующим образом: "Полностью раскрыть потенциал Всемирной паутины путем создания протоколов и принципов, гарантирующих долгосрочное развитие Сети". Две другие важнейшие задачи Консорциума — обеспечить полную "интернационализацию Сети" и сделать ее доступной для людей с ограниченными возможностями.

W3C разрабатывает для WWW единые принципы и стандарты, называемые " *Рекомендациями* ", которые затем внедряются разработчиками программ и оборудования. Благодаря *Рекомендациям* достигается совместимость между программными продуктами и оборудованием различных компаний, что делает сеть WWW более совершенной, универсальной и удобной в использовании.

Все *Рекомендации* W3C открыты, то есть, не защищены патентами и могут внедряться любым человеком без каких-либо финансовых отчислений Консорциуму.

Для удобства пользователей Консорциумом созданы специальные *программы-валидаторы* (англ. Online Validation Service), которые доступны по сети и могут за несколько секунд проверить документы на соответствие популярным Рекомендациям W3C. Консорциумом также созданы многие другие утилиты для облегчения работы веб-мастеров и программистов. Большинство утилит — это программы с открытым исходным кодом, все они бесплатные. В последнее время, повинуясь мировым тенденциям, Консорциум, в целом, гораздо больше внимания уделяет проектам с открытым исходным кодом.

В российском сегменте Интернета имеется своя организация - Российский НИИ Развития Общественных Сетей *РОСНИИРОС* (Russian Institute for Public Networks, RIPN). *РОСНИИРОС* занимается координацией российских исследований и разработок в Интернете.

Прежде чем перейти к описанию структуры, принципов работы и основных протоколов сети Веб, рассмотрим основной стек протоколов сети Интернет - стек TCP/IP.

Стек протоколов TCP/IP

Эти протоколы изначально ориентированы на глобальные сети, в которых качество соединительных каналов не идеально. Он позволяет создавать глобальные сети, компьютеры в которых соединены друг с другом самыми разными способами от высокоскоростных оптоволоконных кабелей и спутниковых каналов до коммутируемых телефонных линий. TCP/IP соответствует модели OSI достаточно условно и содержит 4 уровня. Прикладной уровень стека соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому.

В сети данные всегда передаются блоками относительно небольшого размера. Каждый блок имеет префиксную часть (заголовок), описывающую содержимое блока, и суффиксную, содержащую, например, информацию для контроля целостности передаваемого блока данных.

Название стека протоколов TCP/IP состоит из названий двух разных протоколов. Протокол IP (Internet Protocol) представляет собой протокол нижнего (сетевого) уровня и отвечает за передачу пакетов данных в сети. Он относится к так называемым протоколам *датаграмм* и работает без подтверждений. Последнее означает, что при его использовании доставка пакетов данных не гарантируется и не подтверждается. Не гарантируется также и то, что пакеты достигнут пункта назначения в той последовательности, в которой они были отправлены.

К протоколам сетевого уровня относится также протокол межсетевых управляющих сообщений *ICMP* (Internet Control Message Protocol), предназначенный для передачи маршрутизатором источнику информации об ошибках при передаче пакета.

Очевидно, что намного удобнее передавать данные по каналу, который работает корректно, доставляя все пакеты по порядку. Поэтому над протоколом IP работает протокол передачи данных более высокого (транспортного) уровня — TCP (Transmission Control Protocol). Посылая и принимая пакеты через протокол IP, протокол TCP гарантирует доставку всех переданных пакетов данных в правильной последовательности.

Следует отметить, что при использовании протокола IP обеспечивается более быстрая передача данных, так как не тратится время на подтверждение приема каждого пакета. Есть и другие преимущества. Одно из них заключается в том, что он позволяет рассылать пакеты данных в широковещательном режиме, при котором они достигают всех компьютеров физической сети. Что же касается протокола TCP, то для передачи данных с его помощью необходимо создать канал связи между компьютерами. Он и создается с использованием протокола IP.

Для идентификации сетевых интерфейсов используются 3 типа адресов:

- аппаратные адреса (или MAC-адреса);
- сетевые адреса (IP-адреса);
- символьные (доменные) имена.

В рамках IP протокола для создания глобальной системы адресации, не зависящей от способов адресации узлов в отдельных сетях, используется пара идентификаторов, состоящая из номера сети и номера узла. При этом IP-адрес идентифицирует не отдельный компьютер или маршрутизатор, а одно сетевое соединение в составе сети, в которую он входит; то есть конечный узел может входить в несколько IP-сетей.

Система доменных имен DNS

Несмотря на то, что аппаратное и программное обеспечение в рамках TCP/IP сетей для идентификации узлов использует IP-адреса, пользователи предпочитают *символьные имена* (*доменные имена*).

Первоначально в локальных сетях из небольшого числа компьютеров применялись плоские имена, состоящие из последовательности символов без разделения их на отдельные части, например *MYCOMP*. Для установления соответствия между символьными именами и числовыми адресами использовались широковещательные запросы. Однако для больших территориально распределенных сетей, работающих на основе протокола TCP/IP такой способ оказался неэффективным. Поэтому для установления соответствия между доменным именем и IP-адресом используется специальная система доменных имен (DNS, Domain Name System), которая основана на создаваемых администраторами сети таблиц соответствия.

В сетях TCP/IP используется доменная система имен, имеющая иерархическую (в виде дерева) структуру. Данная структура имен напоминает иерархию имен, используемую во многих файловых системах. Запись доменного имени начинается с самой младшей составляющей, затем после точки следует следующая по старшинству символьная часть имени и так далее. Последовательность заканчивается корневым именем, например: *company.yandex.ru*.

Построенная таким образом система имен позволяет разделять административную ответственность по поддержке уникальности имен в пределах своего уровня иерархии между различными людьми или организациями.

Совокупность имен, у которых несколько старших составных частей совпадают, образуют *домен* имен.

Корневой домен управляется центральными органами Интернета: *IANA* и *Internic*.

Домены верхнего уровня назначаются для каждой страны, а также для различных типов организаций. Имена этих доменов должны следовать международному стандарту *ISO 3166*. Для обозначения стран используются двухбуквенные аббревиатуры, например *ru* (Российская Федерация), *us* (США), *it* (Италия), *fr* (Франция).

Для различных типов организаций используются трехбуквенные аббревиатуры:

- *net* – сетевые организации;
- *org* – некоммерческие организации;
- *com* - коммерческие организации;
- *edu* – образовательные организации;
- *gov* – правительственные организации.

Администрирование каждого домена возлагается на отдельную организацию, которая делегирует администрирование поддоменов другим организациям.

Для получения доменного имени необходимо зарегистрироваться в соответствующей организации, которой организция *InterNIC* делегировала свои полномочия по распределению доменных имен.

Регистратором доменных имен в зоне *ru* до 2005 г. являлся Российский научно-исследовательский институт развития общественных сетей (*РосНИИРОС*). В настоящее время регистрация доменов осуществляется одним из действующих *регистраторов*.

В TCP/IP сетях соответствие между доменными именами и IP-адресами может устанавливаться как локальными средствами, так и централизованными службами. Первоначально соответствие задавалось с помощью создаваемого вручную на хосте

файла hosts.txt, состоящего из строк, содержащих пару вида "доменное имя – IP-адрес". Однако с активным ростом Интернета такое решение оказалось немасштабируемым.

Альтернативное решение – централизованная служба DNS, использующая распределенную базу отображений "доменное имя – IP-адрес". Сервер домена хранит только имена, которые заканчиваются на следующем ниже по дереву уровне. Это позволяет распределять более равномерно нагрузку по разрешению имен между всеми DNS-серверами. Каждый DNS-сервер помимо таблицы отображения имен содержит ссылки на DNS-серверы своих поддоменов.

Существуют две схемы разрешения DNS-имен.

Нерекурсивная процедура:

1. DNS-клиент обращается к корневому DNS-серверу с указанием полного доменного имени;
2. DNS-сервер отвечает клиенту, указывая адрес следующего DNS-сервера, обслуживающего домен верхнего уровня, заданный в следующей старшей части имени;
3. DNS-клиент делает запрос следующего DNS-сервера, который отсылает его к DNS-серверу нужного поддомена и т.д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Сервер дает окончательный ответ клиенту.

Рекурсивная процедура:

1. DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, которому принадлежит клиент;
2. Далее
3. Если локальный DNS-сервер знает ответ, он возвращает его клиенту
4. Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу. После получения ответа сервер передает его клиенту.

Таким образом, при рекурсивной процедуре клиент фактически перепоручает работу своему серверу. Для ускорения поиска IP-адресов DNS-серверы широко применяют кэширование (на время от часов до нескольких дней) проходящих через них ответов.

Структура и принципы WWW

Сеть WWW образуют миллионы *веб-серверов*, расположенных по всему миру. *Веб - сервер* является программой, запускаемой на подключенном к сети компьютере и передающей данные по протоколу HTTP.

Для идентификации ресурсов (зачастую файлов или их частей) в WWW используются идентификаторы ресурсов *URI* (Uniform Resource Identifier). Для определения местонахождения ресурсов в этой сети используются локаторы ресурсов *URL* (Uniform Resource Locator). Такие URL-локаторы представляют собой комбинацию URI и системы DNS.

Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетевого интерфейса), на котором работает программа веб-сервер.

На клиентском компьютере для просмотра информации, полученной от веб-сервера, применяется специальная программа — *веб-браузер*. Основная функция веб-браузера - отображение гипертекстовых страниц (веб-страниц). Для создания гипертекстовых страниц в WWW изначально использовался язык HTML. Множество веб-страниц образуют *веб-сайт*.

Прокси-серверы

Прокси-сервер (proxy-server) — служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс, расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного *кеша* (если имеется). В некоторых случаях запрос клиента или ответ сервера может быть изменен прокси-сервером в определенных целях. Также прокси-сервер позволяет защищать клиентский компьютер от некоторых сетевых атак.

Чаще всего прокси-серверы применяются для следующих целей:

- обеспечение доступа с компьютеров локальной сети в Интернет;
- кеширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации.
- сжатие данных: прокси-сервер загружает информацию из Интернета и передает информацию конечному пользователю в сжатом виде.
- защита локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они "видят" только прокси-сервер).
- ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определенным веб-сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.
- анонимизация доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также искажающие прокси-серверы, которые передают целевому серверу ложную информацию об истинном пользователе.

Протоколы Интернет прикладного уровня

Самый верхний уровень в иерархии протоколов Интернет занимают следующие протоколы прикладного уровня:

- *DNS* - распределенная система доменных имен, которая по запросу, содержащему доменное имя хоста сообщает IP адрес;
- *HTTP* - протокол передачи гипертекста в Интернет;
- *HTTPS* - расширение протокола HTTP, поддерживающее шифрование;
- *FTP* (File Transfer Protocol - RFC 959) - протокол, предназначенный для передачи файлов в компьютерных сетях;
- *Telnet* (TELecommunication NETwork - RFC 854) - сетевой протокол для реализации текстового интерфейса по сети;
- *SSH* (Secure Shell - RFC 4251) - протокол прикладного, позволяющий производить удаленное управление операционной системой и передачу файлов. В отличие от Telnet шифрует весь трафик;
- *POP3* – протокол почтового клиента, который используется почтовым клиентом для получения сообщений электронной почты с сервера;
- *IMAP* - протокол доступа к электронной почте в Интернет;
- *SMTP* – протокол, который используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю;

- *LDAP* - протокол для доступа к службе каталогов X.500, является широко используемым стандартом доступа к службам каталогов;
- *XMPP* (Jabber) - основанный на XML расширяемый протокол для мгновенного обмена сообщениями в почти реальном времени;
- *SNMP* - базовый протокол управления сети Internet.

Рассмотрим более подробно некоторые из этих протоколов.

1. **FTP**

   FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер; кроме того, возможен режим передачи файлов между серверам; FTP позволяет обмениваться файлами и выполнять операции над ними через TCP-сети. Данный протокол работает независимо от операционных систем. Исторически протокол FTP предложил открытую функциональность, обеспечивая прозрачный перенос файлов с одного компьютера на другой по сети. Это не так тривиально, как может показаться, так как у разнотипных компьютеров могут различаться размеры слов, биты в словах могут храниться в неодинаковом порядке или использоваться разные форматы слов.

2. **Telnet**

   Название "telnet" имеют также некоторые утилиты, реализующие клиентскую часть протокола. Протокол *telnet* работает в соответствии с принципами архитектуры "клиент-сервер" и обеспечивает эмуляцию алфавитно-цифрового терминала, ограничивая пользователя режимом командной строки. Приложение *telnet* предоставило язык для общения терминалов с удаленными компьютерами. Когда появилась сеть ARPANET, для каждой компьютерной системы требовались собственные терминалы. Приложение *telnet* стало общим знаменателем для терминалов. Достаточно было написать для каждого компьютера программное обеспечение, поддерживающее "терминал *telnet* ", чтобы один терминал мог взаимодействовать с компьютерами всех типов.

3. **SSH**

   Сходен по функциональности с протоколами telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH-клиенты и SSH-серверы имеются для большинства операционных систем.

4. **Почтовые протоколы**.

   Хотя *telnet* и FTP были (и остаются) полезными, первым приложением, совершившим переворот в сознании пользователей компьютеров сети ARPANET, стала электронная почта. До сети ARPANET существовали системы электронной почты, но все они были однокомпьютерными системами. В 1972 г. *Рэй Томлинсон* (Ray Tomlinson) из компании BBN написал первый пакет, предоставляющий распределенные почтовые услуги в компьютерной сети из нескольких компьютеров. Уже к 1973 г. исследования управления ARPA показали, что три четверти всего трафика сети ARPANET составляла электронная почта. Польза электронной почты оказалась столь велика, что все больше пользователей стремилось подключиться к сети ARPANET, в результате чего возрастала потребность в добавлении новых узлов и использовании высокоскоростных линий. Таким образом, появилась тенденция, сохраняющаяся и по сей день.

      - *POP3* (Post Office Protocol Version 3 - RFC 1939) — протокол, который используется почтовым клиентом для получения сообщений электронной почты с почтового сервера;

- *IMAP* (Internet Message Access Protocol - RFC 3501) — протокол доступа к электронной почте. Аналогичен POP3, однако предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем.
- *SMTP* (Simple Mail Transfer Protocol — RFC 2821) — протокол, предназначенный для передачи электронной почты. Используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю. Для приема почты почтовый клиент должен использовать протоколы POP3 или IMAP.

Базовым протоколом сети гипертекстовых ресурсов Веб является протокол HTTP. В его основу положено взаимодействие " *клиент-сервер* ", то есть предполагается, что:

1. Потребитель- *клиент* инициировав соединение с поставщиком-сервером посылает ему запрос;
2. Поставщик- *сервер*, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.

   При этом возможны два способа организации работы компьютера-клиента:

   - *Тонкий клиент* - это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями.
   - *Толстый клиент*, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Прежде чем перейти к конкретным клиент-серверным веб-технологиям, рассмотрим основные принципы и структуру базового протокола HTTP.

Протокол HTTP

HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного уровня для передачи гипертекста.

Центральным объектом в HTTP является *ресурс*, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя изначально данный протокол предназначен для передачи символьной информации. На первый взгляд это может показаться излишней тратой ресурсов. Действительно, данные в символьном виде занимают больше памяти, сообщения создают дополнительную нагрузку на каналы связи, однако подобный формат имеет много преимуществ. Сообщения, передаваемые по сети, удобочитаемы, и, проанализировав полученные данные, системный администратор может легко найти ошибку и устранить ее. При необходимости роль одного из взаимодействующих приложений может выполнять человек, вручную вводя сообщения в требуемом формате.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера. Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Например, клиентское веб-приложение, посылающее запросы, может отслеживать

задержки ответов, а веб-сервер может хранить IP-адреса и заголовки запросов последних клиентов.

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.

Основными клиентами являются *браузеры* например: Internet Explorer, Opera, Mozilla Firefox, Netscape Navigator и другие. Наиболее популярными реализациями веб-серверов являются: Internet Information Services (IIS), Apache, lighttpd, nginx. Наиболее известные реализации прокси-серверов: Squid, UserGate, Multiproxy, Naviscope.

"Классическая" схема HTTP-сеанса выглядит так.

1. Установление TCP-соединения.
2. Запрос клиента.
3. Ответ сервера.
4. Разрыв TCP-соединения.

Таким образом, клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Обычно запрос клиента представляет собой требование передать HTML-документ или какой-нибудь другой ресурс, а ответ сервера содержит код этого ресурса.

В состав HTTP-запроса, передаваемого клиентом серверу, входят следующие компоненты.

- Строка состояния (иногда для ее обозначения используют также термины строка-статус, или строка запроса).
- Поля заголовка.
- Пустая строка.
- Тело запроса.

Строку состояния вместе с *полями заголовка* иногда называют также *заголовком запроса*.

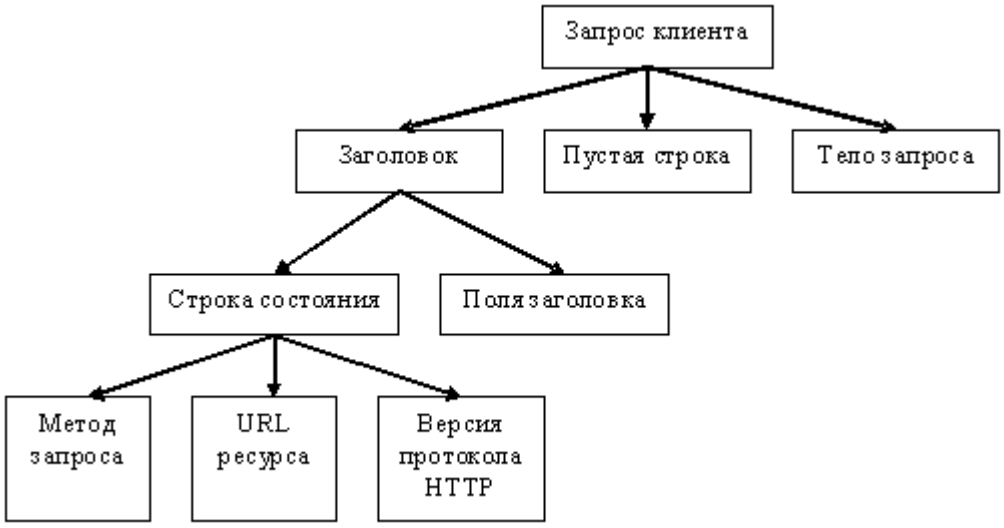

Рис. 2.1. Структура запроса клиента.
Строка состояния имеет следующий формат:

```
метод_запроса   URL_ресурса        версия_протокола_HTTP
```

Рассмотрим компоненты строки состояния, при этом особое внимание уделим методам запроса.

Метод, указанный в строке состояния, определяет способ воздействия на ресурс, URL которого задан в той же строке. Метод может принимать значения `GET`, `POST`, `HEAD`, `PUT`, `DELETE` и т.д. Несмотря на обилие методов, для веб-программиста по-настоящему важны лишь два из них: `GET` и `POST`.

- `GET`. Согласно формальному определению, метод `GET` предназначается для получения ресурса с указанным URL. Получив запрос `GET`, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс, URL которого передается в составе запроса, не обязательно должен представлять собой HTML-страницу, файл с изображением или другие данные. URL ресурса может указывать на исполняемый код программы, который, при соблюдении определенных условий, должен быть запущен на сервере. В этом случае клиенту возвращается не код программы, а данные, сгенерированные в процессе ее выполнения. Несмотря на то что, по определению, метод `GET` предназначен для получения информации, он может применяться и в других целях. Метод `GET` вполне подходит для передачи небольших фрагментов данных на сервер.
- `POST`. Согласно тому же формальному определению, основное назначение метода `POST` - передача данных на сервер. Однако, подобно методу `GET`, метод `POST` может применяться по-разному и нередко используется для получения информации с сервера. Как и в случае с методом `GET`, URL, заданный в строке состояния, указывает на конкретный ресурс. Метод `POST` также может использоваться для запуска процесса.
- Методы `HEAD` и `PUT` являются модификациями методов GET и POST.

Версия протокола HTTP, как правило, задается в следующем формате:

```
HTTP/версия.модификация
```

Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос, т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

```
Имя_поля: Значение
```

Назначение поля определяется его именем, которое отделяется от значения двоеточием.

Имена некоторых наиболее часто встречающихся в запросе клиента полей заголовка и их назначение приведены в таблице 2.1.

Таблица 2.1. Поля заголовка запроса HTTP.

Поля заголовка HTTP - запроса	Значение
Host	Доменное имя или IP-адрес узла, к которому обращается клиент
Referer	URL документа, который ссылается на ресурс, указанный в строке состояния
From	Адрес электронной почты пользователя, работающего с клиентом
Accept	MIME-типы данных, обрабатываемых клиентом. Это поле может иметь

	несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент
Accept-Language	Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом
Accept-Charset	Перечень поддерживаемых наборов символов
Content-Type	MIME-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Content-Length	Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка)
Range	Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть
Connection	Используется для управления TCP-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать TCP-соединение, чтобы оно могло быть использовано для последующих запросов
User-Agent	Информация о клиенте

Во многих случаях при работе в Веб тело запроса отсутствует. При запуске CGI-сценариев данные, передаваемые для них в запросе, могут размещаться в теле запроса.

Ниже представлен пример HTML-запроса, сгенерированного браузером

```
GET http://oak.oakland.edu/ HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.04 [en]   (Win95;  I)
Host: oak.oakland.edu
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Получив от клиента запрос, сервер должен ответить ему. Знание структуры ответа сервера необходимо разработчику веб-приложений, так как программы, которые выполняются на сервере, должны самостоятельно формировать ответ клиенту.

Подобно запросу клиента, ответ сервера также состоит из четырех перечисленных ниже компонентов.

- Строка состояния.
- Поля заголовка.
- Пустая строка.
- Тело ответа.

Ответ сервера клиенту начинается со строки состояния, которая имеет следующий формат:

```
Версия_протокола        Код_ответа     Пояснительное_сообщение
```

- **Версия_протокола** задается в том же формате, что и в запросе клиента, и имеет тот же смысл.
- **Код_ответа** - это трехзначное десятичное число, представляющее в закодированном виде результат обслуживания запроса сервером.
- **Пояснительное_сообщение** дублирует код ответа в символьном виде. Это строка символов, которая не обрабатывается клиентом. Она предназначена для системного администратора или оператора, занимающегося обслуживанием системы, и является расшифровкой кода ответа.

Из трех цифр, составляющих код ответа, первая (старшая) определяет класс ответа, остальные две представляют собой номер ответа внутри класса. Так, например, если запрос был обработан успешно, клиент получает следующее сообщение:

```
HTTP/1.0        200     OK
```

Как видно, за версией протокола HTTP 1.0 следует код 200. В этом коде символ 2 означает успешную обработку запроса клиента, а остальные две цифры (00) — номер данного сообщения.

В используемых в настоящее время реализациях протокола HTTP первая цифра не может быть больше 5 и определяет следующие классы ответов.

- 1 - специальный класс сообщений, называемых информационными. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса. При обмене данными между HTTP-клиентом и HTTP-сервером сообщения этого класса используются достаточно редко.
- 2 - успешная обработка запроса клиента.
- 3 - перенаправление запроса. Чтобы запрос был обслужен, необходимо предпринять дополнительные действия.
- 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.

Примеры кодов ответов, которые клиент может получить от сервера, и поясняющие сообщения приведены в таблице 2.2.

Таблица 2.2. 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

В ответе используется такая же структура полей заголовка, как и в запросе клиента. Поля заголовка предназначены для того, чтобы уточнить ответ сервера клиенту. Описание некоторых из полей, которые можно встретить в заголовке ответа сервера, приведено в таблице 2.3.

Таблица 2.3. 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

В теле ответа содержится код ресурса, передаваемого клиенту в ответ на запрос. Это не обязательно должен быть HTML-текст веб-страницы. В составе ответа могут передаваться изображение, аудио-файл, фрагмент видеоинформации, а также любой другой тип данных, поддерживаемых клиентом. О том, как следует обрабатывать полученный ресурс, клиенту сообщает содержимое поля заголовка Content - type.

Ниже представлен пример ответа сервера на запрос, приведенный в предыдущем разделе. В теле ответа содержится исходный текст HTML-документа.

```
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

<html>
<body>
<form action='http://localhost/Scripts/test.pl'>
<p>Operand1: <input type='text' name='A'></p>
<p>Operand2: <input type='text' name='B'></p>
<p>Operation:<br>
<select name='op'>
<option value='+'>+</option>
<option value='-'>-</option>
<option value='*'>*</option>
<option value='/'>/</option>
```

```
<select></p>
<input type='submit' value='Calculate!'>
</from>
</body>
</html>
```
Поля заголовка и тело сообщения могут отсутствовать, но строка состояния является обязательным элементом, так как указывает на тип запроса/ответа.

Поле с именем `Content-type` может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается `MIME` -тип содержимого запроса или ответа. `MIME` -тип также передается в поле заголовка `Accept`, присутствующего в запросе.

Спецификация `MIME` (Multipurpose Internet Mail Extension — многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем. Однако применение `MIME` не исчерпывается электронной почтой. Средства MIME успешно используются в WWW и, по сути, стали неотъемлемой частью этой системы.

Стандарт `MIME` разработан как расширяемая спецификация, в которой подразумевается, что число типов данных будет расти по мере развития форм представления данных. Каждый новый тип в обязательном порядке должен быть зарегистрирован в IANA (Internet Assigned Numbers Authority).

До появления `MIME` компьютеры, взаимодействующие по протоколу HTTP, обменивались исключительно текстовой информацией. Для передачи изображений, как и для передачи любых других двоичных файлов, приходилось пользоваться протоколом FTP.

В соответствии со спецификацией MIME, для описания формата данных используются *тип* и *подтип*. *Тип* определяет, к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. *Подтип* уточняет формат. Тип и подтип отделяются друг от друга косой чертой:

`тип/подтип`
Поскольку в подавляющем большинстве случаев в ответ на запрос клиента сервер возвращает исходный текст HTML-документа, то в поле `Content-type` ответа обычно содержится значение `text/html`. Здесь идентификатор `text` описывает тип, сообщая, что клиенту передается символьная информация, а идентификатор `html` описывает подтип, т.е. указывает на то, что последовательность символов, содержащаяся в теле ответа, представляет собой описание документа на языке HTML.

Перечень типов и подтипов `MIME` достаточно велик. В таблице 2.4 приведены примеры `MIME` -типов, наиболее часто встречающиеся в заголовках HTML-запросов и ответов.

Таблица 2.4. MIME типы данных.

Тип/подтип	Расширение файла	Описание
application/pdf	.pdf	Документ, предназначенный для обработки Acrobat Reader
application/msexcel	.xls	Документ в формате Microsoft Excel
application/postscript	.ps, .eps	Документ в формате PostScript
application/x-tex	.tex	Документ в формате TeX
application/msword	.doc	Документ в формате Microsoft Word
application/rtf	.rtf	Документ в формате RTF, отображаемый с помощью Microsoft Word

image/gif	.gif	Изображение в формате GIF
image/ jpeg	.jpeg, .jpg,	Изображение в формате JPEG
image/tiff	.tiff, .tif	Изображение в формате TIFF
image/x-xbitmap	.xbm	Изображение в формате XBitmap
text/plain	.txt	ASCII-текст
text/html	. html , . htm	Документ в формате HTML
audio/midi	.midi, .mid	Аудиофайл в формате MIDI
audio/x-wav	.wav	Аудиофайл в формате WAV
message/rfc822		Почтовое сообщение
message/news		Сообщение в группы новостей
video /mpeg	.mpeg, .mpg, .mpe	Видеофрагмент в формате MPEG
video/avi	.avi	Видеофрагмент в формате AVI

Для однозначной идентификации ресурсов в сети Веб используются уникальные идентификаторы URL.

Единообразный идентификатор ресурса URI (Uniform Resource Identifier) представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс. URI не указывает на то, как получить ресурс, а только идентифицирует его. Это дает возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через Интернет (имена, названия и т.п.). Самые известные примеры URI - это URL и URN.

- URL (Uniform Resource Locator) - это URI, который, помимо идентификации ресурса, предоставляет еще и информацию о местонахождении этого ресурса.
- URN (Uniform Resource Name) - это URI, который идентифицирует ресурс в определенном пространстве имен, но, в отличие от URL, URN не указывает на местонахождение этого ресурса.

URL имеет следующую структуру:

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

где:

- `схема` - схема обращения к ресурсу (обычно сетевой протокол);
- `логин` - имя пользователя, используемое для доступа к ресурсу;
- `пароль` - пароль, ассоциированный с указанным именем пользователя;
- `хост` - полностью прописанное доменное имя хоста в системе DNS или IP-адрес хоста;
- `порт` - порт хоста для подключения;
- `URL-путь` - уточняющая информация о месте нахождения ресурса.

Общепринятые схемы (протоколы) URL включают протоколы: *ftp, http, https, telnet*, а также:

- gopher — протокол Gopher;
- mailto — адрес электронной почты;
- news — новости Usenet;
- nntp — новости Usenet через протокол NNTP;
- irc — протокол IRC;
- prospero — служба каталогов Prospero Directory Service;
- wais — база данных системы WAIS;
- xmpp — протокол XMPP (часть Jabber);
- file — имя локального файла;
- data — непосредственные данные (Data: URL);

Обеспечение безопасности передачи данных HTTP

Поскольку протокол HTTP предназначен для передачи символьных данных в открытом (незашифрованном) виде, то лица, имеющие доступ к каналу передачи данных между клиентом и сервером, могут без труда просматривать весь трафик и использовать его для совершения несанкционированных действий. В связи с этим предложен ряд расширений базового протокола направленных на повышение защищенности интернет-трафика от несанкционированного доступа.

Самым простейшим является расширение HTTPS, при котором данные, передаваемые по протоколу HTTP, "упаковываются" в криптографический протокол SSL или TLS, тем самым обеспечивая защиту этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Чтобы подготовить веб-сервер для обработки HTTPS соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера.

SSL (Secure Sockets Layer) - криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет. При его использовании создается защищенное соединение между клиентом и сервером. SSL изначально разработан компанией Netscape Communications. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший название TLS. Этот протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надежность передачи данных за счет использования корректирующих кодов и безопасных хэш-функций. На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (POP3, IMAP, SMTP или HTTP). Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнет передавать и получать данные.

Для доступа к веб-страницам, защищенным протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https — 443. Для работы SSL требуется, чтобы на сервере имелся SSL -*сертификат*.

В сети Веб поддерживаются 3 типа аутентификации при клиент-серверных взаимодействиях:

- Basic - базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов. Пароль при этом не шифруется и присутствует в чистом виде в кодировке base64. 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.

Cookie

Поскольку HTTP-сервер не помнит предыстории запросов клиентов, то каждый запрос обрабатывается независимо от других, и у сервера нет возможности определить, исходят ли запросы от одного клиента или разных клиентов.

Если сервер будет проверять TCP-соединения и запоминать IP-адреса компьютеров-клиентов, он все равно не сможет различить запросы от двух браузеров, выполняющихся на одной машине. И даже если допустить, что на компьютере работает лишь одна клиент-программа, то никто не может утверждать, что в промежутке между двумя запросами она не была завершена, а затем запущена снова уже другим пользователем.

Тем не менее, если вы когда-нибудь пользовались почтовым ящиком на mail.ru или на другом сервере, предоставляющем почтовые услуги пользователям Веб, вспомните, как вел себя клиент после того, как вы создали для себя почтовый ящик на сервере. Когда вы в следующий раз обратились с того же компьютера к mail.ru, вы, вероятно, заметили, что после загрузки веб-страницы ваше регистрационное имя уже отображалось в соответствующем поле ввода.

Такие сведения позволяет получить дополнительное средство под названием `cookie`. Механизм `cookie` позволяет серверу хранить информацию на компьютере клиента и извлекать ее оттуда.

Инициатором записи `cookie` выступает сервер. Если в ответе сервера присутствует поле заголовка `Set-cookie`, клиент воспринимает это как команду на запись `cookie`. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка `Set-cookie`, помимо прочей информации он передает серверу данные `cookie`. Для передачи указанной информации серверу используется поле заголовка `Cookie`.

Для того чтобы в общих чертах представить себе, как происходит обмен данными `cookie`, рассмотрим следующий пример. Предположим, что клиент передает запросы на серверы **A**, **B** и **C**. Предположим также, что сервер **B**, в отличие от **A** и **C**, передает клиенту команду записать `cookie`. Последовательность запросов клиента серверу и ответов на них будет выглядеть приблизительно следующим образом.

1. Передача запроса серверу **A**.
2. Получение ответа от сервера **A**.
3. Передача запроса серверу **B**.
4. Получение ответа от сервера **B**. В состав ответа входит поле заголовка `SetCookie`. Получив его, клиент записывает `cookie` на диск.
5. Передача запроса серверу **C**. Несмотря на то что на диске хранится запись `cookie`, клиент не предпринимает никаких специальных действий, так как значение `cookie` было записано по инициативе другого сервера.
6. Получение ответа от сервера **C**.
7. Передача запроса серверу **A**. В этом случае клиент также никак не реагирует на тот факт, что на диске хранится `cookie`.
8. Получение ответа от сервера **A**.
9. Передача запроса серверу **B**. Перед тем как сформировать запрос, клиент определяет, что на диске хранится запись `cookie`, созданная после получения ответа от сервера **B**. Клиент проверяет, удовлетворяет ли данный запрос некоторым требованиям, и, если проверка дает положительный результат, включает в заголовок запроса поле `Cookie`.

Таким образом, процедуру записи и получения `cookie` можно представить себе как своеобразный "запрос" сервера, инкапсулированный в его ответе клиенту. Соответственно получение `cookie` также можно представить себе как ответ клиента, инкапсулированный в составе запроса тому же серверу.

Рассмотрим подробнее, какие данные передаются в поле заголовка `Set-cookie` и как они влияют на поведение клиента.

Поле `Set-cookie` имеет следующий формат:

```
Set-cookie: имя = значение; expires = дата; path = путь; domain = имя_домена, secure
```

где

- Пара `имя = значение` – именованные данные, сохраняемые с помощью механизма `cookie`. Эти данные должны храниться на клиент-машине и передаваться серверу в составе очередного запроса клиента.
- Дата, являющаяся значением параметра `expires`, определяет время, по истечении которого информация `cookie` теряет свою актуальность. Если ключевое слово `expires` отсутствует, данные `cooki` е удаляются по окончании текущего сеанса работы браузера.
- Значение параметра `domain` определяет домен, с которым связываются данные `cookie`. Чтобы узнать, следует ли передавать в составе запроса данные `cookie`, браузер сравнивает доменное имя сервера, к которому он собирается обратиться, с доменами, которые связаны с записями `cookie`, хранящимися на клиент-машине. Результат проверки будет считаться положительным, если сервер, которому направляется запрос, принадлежит домену, связанному с `cookie`. Если соответствие не обнаружено, данные cookie не передаются.
- Путь, указанный в качестве значения параметра `path`, позволяет выполнить дальнейшую проверку и принять окончательное решение о том, следует ли передавать данные `cookie` в составе запроса. Помимо домена с записью cookie связывается путь. Если браузер обнаружил соответствие имени домена значению параметра `domain`, он проверяет, соответствует ли путь к ресурсу пути, связанному с `cookie`. Сравнение считается успешным, если ресурс содержится в каталоге, указанном посредством ключевого слова `path`, или в одном из его подкаталогов. Если и эта проверка дает положительный результат, данные cookie передаются серверу. Если параметр `path` в поле `Set-Cookie` отсутствует, то считается, что запись cookie связана с URL конкретного ресурса, передаваемого сервером клиенту.
- Последний параметр, `secure`, указывает на то, что данные `cookie` должны передаваться по защищенному каналу.

Для передачи данных cookie серверу используется поле заголовка `Cookie`. Формат этого поля достаточно простой:

```
Cookie: имя=значение; имя=значение;  ...
```

С помощью поля `Cookie` передается одна или несколько пар `имя = значение`. Каждая из этих пар принадлежит записи cookie, для которой URL запрашиваемого ресурса соответствуют имени домена и пути, указанным ранее в поле `Set-cookie`.