Лабораторная работа № 5

Программирование сетевого взаимодействия по протоколам HTTP/FTP

1. Теоретические сведения

HTTP (сокр. от англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (англ. Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что- то абстрактное. 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 данный протокол является текстовым

HTTP — протокол прикладного уровня, аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Структура протокола

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

1. Стартовая строка (англ. Starting line) — определяет тип сообщения;

2. Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;

3. Тело сообщения (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа только тело сообщения.

Стартовая строка
Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

GET URI — для версии протокола 0.9.
Метод URI HTTP/Версия — для остальных версий.

Здесь:

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.
Версия (англ. Version) — пара разделённых точкой арабских цифр. Например: 1.0.

Чтобы запросить страницу данной статьи, клиент должен передать строку:

GET /wiki/HTTP HTTP/1.0
Стартовая строка ответа сервера имеет следующий формат:
HTTP/Версия КодСостояния Пояснение

Здесь:
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.

Например, на предыдущий наш запрос клиентом данной страницы сервер ответил строкой:

HTTP/1.0 200 OK

Методы

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. Обратите внимание, что название метода чувствительно к регистру.
Каждый сервер обязан поддерживать как минимум методы GET и HEAD. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он не применим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Кроме методов GET и HEAD, часто применяется метод POST.

GET

Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Кроме обычного метода GET, различают ещё условный GET и частичный GET. Условные запросы GET содержат заголовки If-Modified-Since, If-Match, If-Range и подобные. Частичные GET содержат в запросе Range. Порядок выполнения подобных запросов определён стандартами отдельно.

HEAD
Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.

POST

Применяется для передачи пользовательских данных заданному ресурсу.
Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться одна копия этого комментария).

При результатах выполнения 200 (Ok) и 204 (No Content) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу

следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке

Location.
Сообщение ответа сервера на выполнение метода POST не кэшируется.

Коды состояния

Код состояния является частью первой строки ответа сервера. Он представляет собой целое число из трех арабских цифр. Первая цифра указывает на класс состояния.

За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.

Примеры:

201 Webpage Created
403 Access allowed only for registered users
507 Insufficient Storage

Клиент узнаёт по коду ответа о результатах его запроса и определяет, какие действия ему предпринимать дальше. Набор кодов состояния является стандартом, и они описаны в соответствующих документах RFC. Введение новых кодов должно производиться только после согласования с IETF. Клиент может не знать все коды состояния, но он обязан отреагировать в соответствии с классом кода.

В настоящее время выделено пять классов кодов состояния.

1xx Informational (русск. Информационный)

В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-сервера подобные сообщения должны отправлять дальше от сервера к клиенту.

2xx Success (русск. Успешно)

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

3xx Redirection (русск. Перенаправление)

Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4xx Client Error (русск. Ошибка клиента)

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.
Для запоминания значений кодов с 400 по 417 существуют приёмы иллюстративной мнемотехники

5xx Server Error (русск. Ошибка сервера)

Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

Заголовки

Заголовки HTTP (англ. HTTP Headers) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (см. RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Примеры заголовков:

Server: Apache/2.2.11 (Win32) PHP/5.3.0
Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT
Content-Type: text/plain; charset=windows-1251
Content-Language: ru

В примере выше каждая строка представляет собой один заголовок. При этом то, что находится до первого двоеточия, называется именем (англ. name), а что после неё — значением (англ. value).

Все заголовки разделяются на четыре основных группы:
1. General Headers (русск. Основные заголовки) — должны включаться в любое сообщение клиента и сервера.
2. Request Headers (русск. Заголовки запроса) — используются только в запросах клиента.
3. Response Headers (русск. Заголовки ответа) — только для ответов от сервера.
4. Entity Headers (русск. Заголовки сущности) — сопровождают каждую сущность сообщения.

Именно в таком порядке рекомендуется посылать заголовки получателю.
Все необходимые для функционирования HTTP заголовки описаны в основных RFC, ссылки на которые есть в конце этой статьи. При этом если вам не будет хватать существующих, то можете смело вводить свои. Традиционно к именам таких дополнительных заголовков добавляют префикс «X-» для избежания конфликта имён с возможно существующими. Например, как в заголовках X-Powered-By или X-Cache.

Некоторые разработчики используют свои индивидуальные префиксы. Примерами таких заголовков могут служить Ms-Echo-Request и Ms-Echo-Reply, введённые корпорацией

Microsoft для расширения WebDAV.

Примеры диалогов http

Обычный GET-запрос

Запрос клиента:

GET /wiki/страница 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 сервера:

HTTP/1.0 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close

(далее следует запрошенная страница в HTML)
Аналогично выглядит ответ 203. Что существенно, непосредственно запрашиваемые данные отделены от HTTP-заголовков с помощью CRLF CRLF (двух переводов строки)

2. Задание на лабораторную работу.

Для задания представленном ниже для коммуникации использовать HTTP протокол. (Вся работа должна быть выполнена с использованием стандартных сокетов, все HTTP заголовки формируются вручную)

Разработать http-сервер (использовать пул потоков для параллельной обработки запросов клиентов). В качестве клиентского приложения используется веб-браузер. Все заголовки формируются вручную. Браузер должен правильно отображать получаемую информацию. Рекомендуется на GET-запрос возвращать html страницу с формой ввода, и выполнять POST-запрос, на который сервер присылает решение.

Отчѐт должен содержать:

- Краткое описание алгоритма

- Описание используемых http заголовков
- Верификация

Варианты заданий

Передается только текстовая информация. Т.е. параметры задачи передаются в теле запроса, ответ приходит в виде html-страницы.	4-5 баллов
Задание передается на сервер в виде текста (вводится в поля ввода html-формы). В ответ возвращает прикрепленный файл, с результатом решения задачи. Формат файла разрешается любой, рекомендуется XML или простой текстовый файл.	6-8 баллов
Задание на сервер передается в виде файла, например изображения или текстового файла с параметрами задачи(в форме ввода используется input для отправки файла). В ответ сервер возвращает прикрепленный файл. Для заданий с фильтрами, это обработанное изображение, для остальных - текстовый файл в любом формате.	9-10 баллов

Варианты согласуются с преподавателем:

1.	Вычисление псевдослучайного числа используя регистр сдвига с линейной обратной связью (LFSR)
2	Одномерная оптимизация функции методом деления попалам
3	Mergesort
4	Одномерная оптимизация функции методом золотого сечения
5	Quicksort
6	Heapsort
7	Insertion Sort
8	Одномерная оптимизация функции методом равномерного поиска
9	Умножение матриц
10	Одномерная оптимизация функции методом Фибоначчи
11	Treesort
12	Одномерная оптимизация функции методом касательных
13	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать обратную польскую запись
14	Вычисление секанса (без использования математических библиотек, используя разложение в ряд Маклоррена)
15	Вычисление синуса (без использования математических библиотек, используя разложение в ряд Маклоррена)
16	Вычисление экспоненты (без использования математических библиотек,

	используя разложение в ряд Маклоррена)
17	Разбор математического выражения с учетом приоритетов операций (+ - * /), и тригонометрических функций (sin, cos, tg) . Использовать обратную польскую запись.
18	Вычислить число ПИ до 50 знака методом Монте-Карло
19	Вычислить число ПИ до 50 знака методом иглы Буффона
20	Вычислить число ПИ до 50 знака разложением в ряд Нилаканта
21	Вычислить квадратный корень из числа, используя итерационную формулу Герона
22	Вычислить квадратный корень из числа, используя алгоритм Ньютона
23	Вычисление натурального логарифма (без использования математических библиотек, используя разложение в ряд Маклоррена)
24	Вычисление гиперболического синуса(без использования математических библиотек, используя разложение в ряд Маклоррена)
25	Вычислить число ПИ до 50 знака разложением в ряд Лейбница
26	Решение СЛАУ через обратную матрицу
27	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, строковые константы, заключенные в двойные кавычки и одиночные символы, заключенные в одинарные кавычки. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
28	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знаков операций и скобок. (Римскими считать числа записанные большими буквами X, V и I). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
29	Разработать конечный автомат для анализа вырожения.

	Входной язык содержит HTML код, нужно проверить правильность объявления ссылки. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов href, alt, blank. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
30	Разработать конечный автомат для анализа вырожения. Входной язык содержит HTML код, нужно проверить правильность тега img. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов src, alt, width, height. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
31	Разработать конечный автомат для анализа корректности email. Найти все email в файле отправленном пользователем. Клиенту возвращается список всех найденных email в тексте
32	Найти обратную матрицу (используя метод Гаусса)
33	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать дерево выражений
34	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Флойда — Уоршелла
35	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Дейкстры
36	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Беллмана — Форда
37	Решение СЛАУ методом Гаусса
38	Решение СЛАУ методом Жардана-Гауса
39	Найти определитель матрицы
40	LDL^T разложение матрицы
41	Найти обратную матрицу (через единичную)
42	Разложение Холецкого
43	LU разложение матрицы
44	Умножение матриц алгоритмом Штрассена

45	Шифрование строки алгоритмом RSA
46	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка C#, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы следующих типов: int, double, float, char и String. Массивы инициализируют только фиксированной длины. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
47	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность команд ассемблера в форме: <метка>: <команда> <операнд1>,<операнд2> (метка, а также один или оба операнда могут отсутствовать). В качестве операндов могут выступать регистры процессора 80x86, идентификаторы, целые десятичные числа или целые шестнадцатеричные числа. (Предусмотреть наличие не менее 6 допустимых команд). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
48	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний записей (record) в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что записи могут содержать только поля скалярных типов integer, real, byte, word, char и строки string с возможным указанием длины строки в квадратных скобках. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
49	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы скалярных типов integer, real, byte, word и char. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
50	Разработать конечный автомат для анализа вырожения.

	Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, целые десятичные числа без знака, шестнадцатеричные числа, десятичные числа с плавающей точкой. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
51	Разработать конечный автомат для анализа вырожения. Входной язык содержит выражения над строковыми константами, разделенные символом ;(точка с запятой). Выражения состоят из идентификаторов, строковых констант, заключенных в двойные кавычки, одиночных символов, заключенных в одинарные кавычки и знаков операции конкатенации +. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
52	Разработать конечный автомат для анализа вырожения. Входной язык содержит упрощенные условные операторы типа if <логическое выражение> then <оператор присваивания> else <оператор присваивания>; (часть else в операторе может отсутствовать). Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и not. Оператор присваивания должен состоять из двух идентификаторов, разделенных знаком присваивания. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
53	Разработать конечный автомат для анализа вырожения. Входной язык содержит упрощенные операторы цикла типа while <логическое выражение> do <оператор присваивания>; Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и or. Оператор присваивания должен состоять из идентификатора, знака присваивания и целой десятичной константы без знака.

	Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
54	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знаков операций и скобок. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
55	решения систем обыкновенных дифференциальных уравнений Рунге-Кутты 4-го порядка
56	решения систем обыкновенных дифференциальных уравнений методом Эйлера
57	Разработать конечный автомат для анализа корректности номера телефона в формате.[+XXX (YY) XXX-XX-XX] Найти все номера телефонов в файле отправленном пользователем. Клиенту возвращается список всех найденных номеров телефонов в тексте
58	Разработать конечный автомат для анализа надежности и корректности пароля Длина пароля не менее 6 символов. Разрешены только латинские буквы, цифры и знаки: пробел _ - . В пароле обязательно должны присутствовать знаки в верхнем и нижнем регистре и цифры. Пользователю возвращается информация о том, можно ли использовать указанный пароль
59	Медианный фильтр изображения с возможностью выбора размера окна
60	Фильтр Гаусса для изображений с возможностью выбора размера окна
61	Фильтр Собеля (оператор Собеля) для выявления границ объектов изображения
62	Перекрестный оператор Робертса для выявления границ объектов на изображении
63	Оператор Прюитт для выделения границ на изображении
64	Проверка вхождения двух человек в одну группу. Исходные данные, имена людей, которые добавлены в друзья в социальной сети. Пользователь вводит 2 имени, нужно определить связаны ли они через общих друзей. (Find Union)

65	Компрессия/Декомпрессия данных. Алгоритм Лемпеля — Зива — Велча (LZW)
66	Компрессия/Декомпрессия данных. алгоритмом Хаффмана
67	Многомерная оптимизация функции методом деформируемого многогранника (Нелдера-Мида)
68	Многомерная оптимизация функции методом Хука Дживса
69	Многомерная оптимизация функции методом градиентного спуска (построить график)
70	Оператор Кенни для поиска границ объектов на изображении

3.Контрольные вопросы

1. Протокол HTTP
2. Виды запросов HTTP протокола
3. GET – запрос
4. POST – запрос
5. HEAD – запрос
6. PUT – запрос
7. TRACE – запрос
8. Отличие HTTPS протокола от HTTP
9. Основные коды ответов HTTP протокола
10. Основные процедуры и классы, для реализации сетевого взаимодействия