Прокси-сервер HTTP

Задача

разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции прокси-сервера для протокола HTTP.

Основные возможности.

Приложение должно реализовывать следующие функции:

1. Обработка подключения клиента
2. Получение запросов от клиента и перенаправление их серверу
3. Получение от сервера файлов и кэширование их
4. Передача клиенту ответа от сервера или кэшированного запроса
5. Обеспечение одновременной работы нескольких клиентов
6. Протоколирование сеанса связи клиентом с сервером

Поддерживаемые команды.

Разработанное приложение должно реализовывать следующие команды протокола SMTP:

- GET – для обеспечения загрузки Web-страниц и медиа-элементов
- HEAD – для передачи заголовков Web-страниц и медиа-элементов
- POST – для получения от клиента параметров Web-форм.

Методика тестирования.

Для проверки работоспособности приложения используются браузеры, установленные в лаборатории (Mozilla Firefox, Internet Explorer, Opera, Netscape). В качестве параметров прокси-сервера устанавливается IP-адрес и TCP-порт, занятые разработанным приложением. Проверяется корректность загрузки различных Web-сайтов, имеющихся в сети Internet. Особое внимание уделяется корректности параллельной загрузки медиа-элементов.

Теоритические сведения о работе прокси-сервера HTTP

Предоставленная информация соответсвует RFC 2068 — Протокол HTTP

Базовая структура

Прокси-сервер — служба, 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 кэша. В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Прокси-сервер позволяет защищать компьютер клиента от некоторых сетевых атак и помогает сохранять анонимность клиента.

Обзор методов прокси-сервера HTTP

HEAD - метод для запроса у сервера заголовка. 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.

GET - используется для запроса содержимого указанного ресурса.Согласно стандарту HTTP, запросы типа GET считаются идемпотентными, т. е. повторный запрос (возможно) не изменит ответного результата. Что делает возможным осуществлять кэширование ответов на данного запрос. 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.

POST - Применяется для передачи пользовательских данных заданному ресурсу. В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты.

Архитектура приложения

Представленное приложение реализовано на языке Python v3.4.

Порт реализован на хосте '127.0.0.1' и порту №8080. При соединении с HTTP-сервером, он обращается к порту №80. Реализованный прокси-сервер поддерживает работу с HTTP версии /1.1. Также поддерживает информацию в кодировке 'utf-8', о чем прокси предупреждает сервер при взаимодействии с ним.

Прокси-сервер поддерживает следующие команды:

- HEAD
- GET
- POST

При организации соединения в зависимости от заданной команды прокси-сервер переылает запрос к HTTP серверу, либо обрабатывает его самостоятельно.

При получении команды *HEAD* прокси-сервер пересылает запрос к серверу, а полученный заголовок ресурса с положительным кодом ответа - обратно клиенту. В случае, если код ответа указывает на ошибку, прокси-сервер оповещает об этом клиента.

Формат запроса от клиента - прокси(обязательные поля):

HEAD [uri] HTTP/1.1
Host: [host]
CRLF
CRLF

Формат запроса от прокси - серверу(обязательные поля):

HEAD [uri] HTTP/1.1
Host: [host]
Accept-Encoding: [charset]
CRLF
CRLF

В данной реализации *charset* соответствует *utf-8*.

Также, как и команда *HEAD*, команда *POST* не обрабатывается и не кэшируется на прокси-сервере, а пересылается серверу. Если от сервера возвращается ответ с положительным кодом, то результат запроса пересылается клиенту, иначе клиент получает код ошибки и описание проблемы.

Формат запроса от клиента - прокси(обязательные поля):

HEAD [uri] HTTP/1.1

Host: [host]
Content-Length: [length of query]
[field1=query1&field2=query2&...]
CRLF
CRLF

Формат запроса от прокси - серверу(обязательные поля):

HEAD [uri] HTTP/1.1
Host: [host]
Accept-Encoding: [charset]
Content-Length: [length of query]
[field1=query1&field2=query2&...]
CRLF
CRLF

В отличии от предыдущих команд, команда *GET* является идемпотентной. Поэтому сначала она обрабатывается прокси-сервером. Приполучении запроса от клиента прокси-сервер сначала проверяет, находится ли в кэше заданный ресурс.

- Если да, то прокси перенаправляет поступивший GET-запроссерверу, добавляет в заголовок запроса еще одно поле: *If-Modified-Since: [mtime]*, где mtime - это последнее время модификации соответствующего ресурса в кэше. В ответ на этот запрос возвращается заголовок со следующими возможными положтельными кодами ответа:

1. 200 OK - Если после указанной даты ресурс изменялся (идентичен ответу на обычный запрос GET)
2. 304 Not Modified - Если ресурс не изменялся после указанной даты

Такой GET-запрос называется условным. Использование такого метода направлено на разгрузку сети, так как он позволяет не передавать по сети избыточную информацию. Таким образом, в случае, когда код ответа - 200, запрос от клиента пересылается серверу, и при положительном коде ответа от HTTP-сервера - прокси-серверу, последний пересылает содержимое ресурса клиенту. После чего начинает анализ полученных данных.

Прокси вновь проверяет заголовок полученого ответа от сервера, в котором содержится информация о возможности кэширования данного ресурса. Реализованный прокси-сервер для этого проверяет заголовок ответа на наличие поля *Cache-Control* и параметра max-age, значение которого для последующего кэширования должно быть больше нуля. В том случае, если содержимое ресурса больше нельзя кэшировать (а старая версия хранится в кэше), то предыдущая версия удаляется.

Если же от сервера был получен ответ с кодом 304, то прокси-сервер отправляет содержимое ресурса из кэша.

- Если же изначально требуемого файла в кэше не существует, то выполняются аналогичные описанным выше шаги: пересылка запроса клиента серверу без дополнительных полей, пересылка результата клиенту, анализ заголовка ответа, и при необходимости, кэширование.

Формат запроса от клиента - прокси(обязательные поля):

GET [uri] HTTP/1.1
Host: [host]
CRLF
CRLF

Формат условного запроса от прокси - серверу(обязательные поля):

GET [uri] HTTP/1.1
Host: [host]
Accept-Encoding: [charset]
If-Modified-Since: [mtime] CRLF
CRLF

Формат mtime:

[weekday, day month year hh:mm:ss GMT]

Возможные коды ответа:

Коды успешной операции

- 200 OK
- 304 Not Modified

Коды ошибок

- 206 Partial Content
- 301 Moved Permanently
- 302 Found
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

Приложение содержит один основной класс - *class Proxy(socketserver.BaseRequestHandler)*, и следующие методы:

handle(self) - соединение с HTTP-сервером и получение запросов от клиента

HEAD(self, uri, header) - обработка команды HEAD

POST(self, uri, content) - обработка команды POST

GET(self, uri, client_header) - обработка команды GET

_removed_cached_page(self, path) - удаление старой кэшированной страницы

_cache_page(self, server_header, body, uri="/") - кэширование страницы

_get_max_age(self, header) - получение значение пааметра заголовка ответа max-age

_connect_to_HTTP_server(self, host, port) - содединение с сервером

_send_cmd(self, str) - отправка запроса серверу

get_request_from_client(self, sock) - получение запроса от клиента

_send_response_to_client(self, header, body) - отправка ответа клиенту

_isListEmpty(self, inList) - вспомогательный метод для проверки на пустоту списка списков

Тестирование приложения

Для тестирования приложения из браузера необходимо осуществить начальную настройку браузера. В данном случае был использован браузер MozillaFirefox v.37.0.2. и выполнены следующие настройки сети:

Меню -> Настройки -> Дополнительные -> Сеть -> Соединение - Настройка параметров соединения Firefox с Интернетом -> Настроить... -> Ручная настройка сервиса прокси: ->

HTTP прокси: localhost Порт:8080.

После выполненой настройки при подключении к серверам по HTTP каналу осуществляется требуемое соединение.

Ниже приведенены заголовки полчаемые и отправляемые прокси в ходе соединения с клиентом на localhost и сервером www.example.com

Заголовок запроса от клиента-прокси:

Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Accept-Encoding: utf-8

Заголовок ответа от сервера-прокси:

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Thu, 23 Apr 2015 18:26:34 GMT
Etag: "359670651"
Expires: Thu, 30 Apr 2015 18:26:34 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (iad/182A)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

После чего осуществляется кэширование страницы TRY TO CACHE PAGE PATH: www.example.com\index.html PAGE CACHED

Повторная загрузка той же страницы:

Заголовок запроса от клиента-прокси:

Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Connection: keep-alive
Accept-Encoding: utf-8

FILE EXIST

Заголовок ответа от сервера-прокси:

HTTP/1.1 304 Not Modified
Accept-Ranges: bytes
Cache-Control: max-age=604800
Date: Thu, 23 Apr 2015 18:30:52 GMT
Etag: "359670651"
Expires: Thu, 30 Apr 2015 18:30:52 GMT

Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (iad/182A)
X-Cache: HIT
x-ec-custom-error: 1

READ FROM FILE

Таким образом, прокси-сервер сам пересылает содержимое ресурса клиенту.

Вывод

В результате выполнения данной работы было разработано приложение для операционных систем семейства Windows, обеспечивающее функции прокси-сервера протокола HTTP: 1. Обработка подключения клиента 2. Получение запросов от клиента и перенаправление их серверу 3. Получение от сервера файлов и кэширование их 4. Передача клиенту ответа от сервера или кэшированного запроса 5. Обеспечение одновременной работы нескольких клиентов 6. Протоколирование сеанса связи клиентом с сервером Для организации корректной работы прокси были реализваны следующие команды протокола HTTP: * GET – для обеспечения загрузки Web-страниц и медиа-элементов * HEAD – для передачи заголовков Web-страниц и медиа-элементов * POST – для получения от клиента параметров Web-форм. Готовое приложение успешно протистрировано в различных браузерах (Firefox, GoogleChrome, InternetExplorer). Однако в работе созданного приложения существуют некоторые ограничения, а именно: не реализовано соединение с HTTPS сервером, нет поддержки некоторых форматов кодирования, например, gzip.