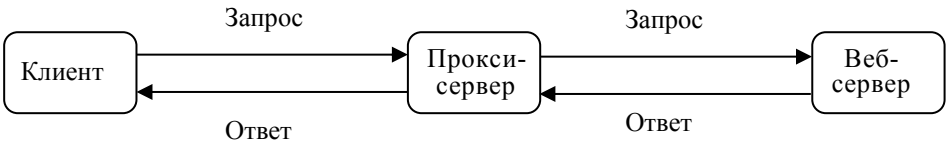
Задача по программированию сокетов 4: Прокси-сервер

В данной работе вы познакомитесь с тем, как работают прокси-серверы в сети и изучите их основную функцию — кэширование.

Ваша задача заключается в разработке небольшого прокси-сервера, который способен кэшировать веб-страницы. Это элементарный прокси-сервер, который понимает только простые GET-запросы, но может обрабатывать все виды объектов — не только HTML-страницы, но и изображения.

Обычно, когда клиент делает запрос, то отправляет его на веб-сервер. Веб-сервер обрабатывает запрос и отправляет ответное сообщение клиенту. Для повышения производительности мы создаем прокси-сервер между клиентом и веб-сервером. Теперь и сообщение-запрос, отправленное клиентом, и ответное сообщение от веб-сервера проходят через прокси-сервер. Другими словами, клиент запрашивает объекты через прокси-сервер, который пересылает запрос веб-серверу. Веб-сервер генерирует ответное сообщение и доставляет его прокси-серверу, а тот, в свою очередь, перенаправляет его клиенту.



Код

Ниже представлена заготовка для клиентской программы. Вам нужно ее завершить. Места, которые вы должны заполнить своим кодом, помечены как **`#Начало вставки`** *и* **`#Конец вставки`**. В каждом таком месте может содержаться одна или несколько строк вашего кода.

Запуск прокси-сервера

Запустите свой прокси-сервер из командной строки, а затем запросите веб-страницу с помощью вашего браузера. Направьте запросы на прокси-сервер, используя свой IP-адрес и номер порта. Например,

```
http://localhost:8888/www.google.com
```

Чтобы использовать вариант прокси-сервера и браузера на разных компьютерах, вам понадобится IP-адрес, на котором прокси-сервер запущен. В этом случае, вместо `localhost` просто подставляете этот IP-адрес. Также обратите внимание на номер используемого порта. Вы должны заменить стоящий здесь `8888` на номер порта, который вы использовали в серверном коде, то есть тот, на котором прокси-сервер слушает запросы.

Настройка браузера

Вы можете также настроить непосредственно веб-браузер на использование вашего прокси-сервера. В Internet Explorer это можно сделать, выбрав команду меню **Сервис ⇒ Свойства обозревателя** (Tools > Internet Options) и на вкладке **Подключения** (Connections) нажав кнопку **Настройка сети** (LAN Settings). В Mozilla Firefox следует выбрать команду меню **Инструменты ⇒ Настройки** (Tools ⇒ Options) и на вкладке **Дополнительные ⇒ Сеть** (Advanced ⇒ Network) нажать кнопку **Настроить** (Settings). В обоих случаях вам нужно будет указать адрес прокси-сервера и номер порта, который вы использовали при запуске прокси-сервер. Без особых проблем вы можете запустить прокси-сервер и браузер на одном и том же компьютере. В этом случае для получения веб-страницы через прокси-сервер вы вводите URL-адрес этой страницы в браузере. Например,

```
http://www.google.com
```

Задание

Завершить код вашего прокси-сервера и сделать снимки экрана на стороне клиента, подтверждающие получение веб-страницы через прокси-сервер.

Шаблон кода прокси-сервера на языке Python

```
from socket import *
import sys
if len(sys.argv) <= 1:
     print 'Используйте : "python ProxyServer.py
server_ip"\n[server_ip – IP-адрес
          прокси-сервера]'
     sys.exit(2)
# Создаем серверный сокет, привязываем его к порту и начинаем
слушать
tcpSerSock = socket(AF_INET, SOCK_STREAM)
# Начало вставки.

# Конец вставки.
while 1:
     # Начинаем получать данные от клиента
     print 'Готов к обслуживанию...'
     tcpCliSock, addr = tcpSerSock.accept()
     print 'Установлено соединение с:', addr
     message = # Начало вставки.   # Конец вставки.
     print message
     # Извлекаем имя файла из сообщения
     print message.split()[1]
     filename = message.split()[1].partition("/")[2]
     print filename
     fileExist = "false"
     filetouse = "/" + filename
     print filetouse
     try:
          # Проверяем, есть ли файл в кэше
          f = open(filetouse[1:], "r")
          outputdata = f.readlines()
          fileExist = "true"
          # Прокси-сервер определяет попадание в кэш и генерирует
          ответное сообщение
          tcpCliSock.send("HTTP/1.0 200 OK\r\n")
          tcpCliSock.send("Content-Type:text/html\r\n")
          # Начало вставки.

          # Конец вставки.
               print 'Читаем из кэша'
     # Обработка ошибки, когда файл в кэше не найден
     except IOError:
          if fileExist == "false":
               # Создаем сокет на прокси-сервере
               c = # Начало вставки.    # Конец вставки.
               hostn = filename.replace("www.","",1)
               print hostn
```

```
                try:
                    # Соединяемся с сокетом по порту 80
                    # Начало вставки.
                    # Конец вставки.
                    # Создаем временный файл на этом сокете и
                    запрашиваем порт 80 файл, который нужен
                    клиенту
                    fileobj = c.makefile('r', 0)
                    fileobj.write("GET "+"http://" + filename + "
                        HTTP/1.0\n\n")
                    # Читаем ответ в буфер
                    # Начало вставки.

                    # Конец вставки.
                    # Создаем новый файл в кэше для
                    запрашиваемого файла
                    # А также отправляем ответ из буфера и
                    соответствующий файл на сокет клиента
                    tmpFile = open("./" + filename,"wb")
                    # Начало вставки.

                    # Конец вставки.
                except:
                    print "Неверный запрос"
            else:
                # HTTP-ответ для файла не найден
                # Начало вставки.

                # Конец вставки.
        # Закрываем сокеты клиента и сервера
        tcpCliSock.close()
# Начало вставки.

# Конец вставки.
```

Дополнительные задания

1. В данном варианте прокси-сервер не производит никакой обработки ошибок. Это может вызвать проблемы, особенно, когда клиент запрашивает объект, который не доступен, так как ответ 404 Not Found, как правило, не имеет тела, а прокси-сервер предполагает, что тело есть и пытается прочитать его.
2. Простой прокси-сервер поддерживает только метод GET протокола HTTP. Добавьте поддержку метода POST путем добавления в запрос тела запроса.
3. *Кэширование*: стандартный прокси-сервер кэширует веб-страницы каждый раз, когда клиент выполняет определенный запрос впервые. Основные функциональные возможности кэширования работают следующим образом. Когда прокси-сервер получает запрос, он проверяет, есть ли запрашиваемый объект в кэше, и если да, то возвращает объект из кэша, без соединения с веб-сервером. Если объекта в кэше нет, прокси-сервер извлекает его с веб-сервера, возвращает клиенту и кэширует копию для будущих запросов. В действительности прокси-сервер должен убедиться, что ответы, находящиеся в кэше, актуальны, и что они на самом деле отвечают запросу клиента. Более подробно о процессе кэширования для HTTP вы можете найти в документе RFC 2068. Добавьте простую функцию

кэширования, описанную выше. Вам не нужно проводить каких-либо процедур замены или проверки. Но ваша программа должны быть способна записывать ответы на диск (то есть, в кэш) и извлекать их с диска в случае попадания в кэш при запросе. Для этого необходимо реализовать некоторую внутреннюю структуру данных в прокси-сервере, чтобы отслеживать, какие объекты скэшированы, и где именно они находятся. Вы можете держать эту структуру данных в оперативной памяти; нет необходимости хранить ее на диске как постоянную.