Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет о лабораторной работе

«Прокси-сервер для протокола FTP, функционирующий в активном режиме»

Дисциплина: «Сетевые технологии»

Выполнила студентка гр. 43501/3 _____А.В. Емельянова
(подпись)
Руководитель _____К.Д. Вылегжанина
(подпись)

Санкт - Петербург
2016

1. **Техническое задание.**

Разработать приложение, обеспечивающее функции прокси-сервера для протокола FTP. Прокси-сервер должен использовать активный режим протокола FTP.

1.1.Основные возможности:

1) Обработка подключения клиента
2) Разбор параметра username для определения реального имени FTP-сервера
3) Переадресация всех команд клиента FTP-серверу
4) Переадресация ответов сервера клиенту
5) Трансляция канала данных, открываемого сервером, клиенту
6) Кэширование данных, присланных сервером
7) Протоколирование соединения сервера с клиентом

1.2. Поддерживаемые команды протокола:

- USER – передача серверу идентификационной информации пользователя вместе с параметрами FTP-сервера
- PASS – передача серверу пароля пользователя
- PORT – передача на сервер параметров(адреса и порта) сокета, осуществляющего приём и передачу данных

Остальные команды протокола FTP должны транслироваться FTP-серверу.

1.3.Настройки приложения.
Разработанное приложение должно предоставлять пользователю настройку следующих параметров:
1) номер порта, прослушиваемого прокси-сервером
2) объём кэша.

1.4.Методика тестирования.

Для тестирования приложения следует использовать стандартные FTP-клиенты (Mozilla Firefox, MS Explorer, Far, Total Commander) и имеющиеся в сети Internet FTP-серверы(ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).
С помощью имеющихся клиентов протокола FTP осуществляется подключение к прокси-серверу с указанием различных реальных FTP-серверов. В процессе тестирования проверяются основные возможности прокси-сервера по трансляции команд и данных, кэшированию информации.

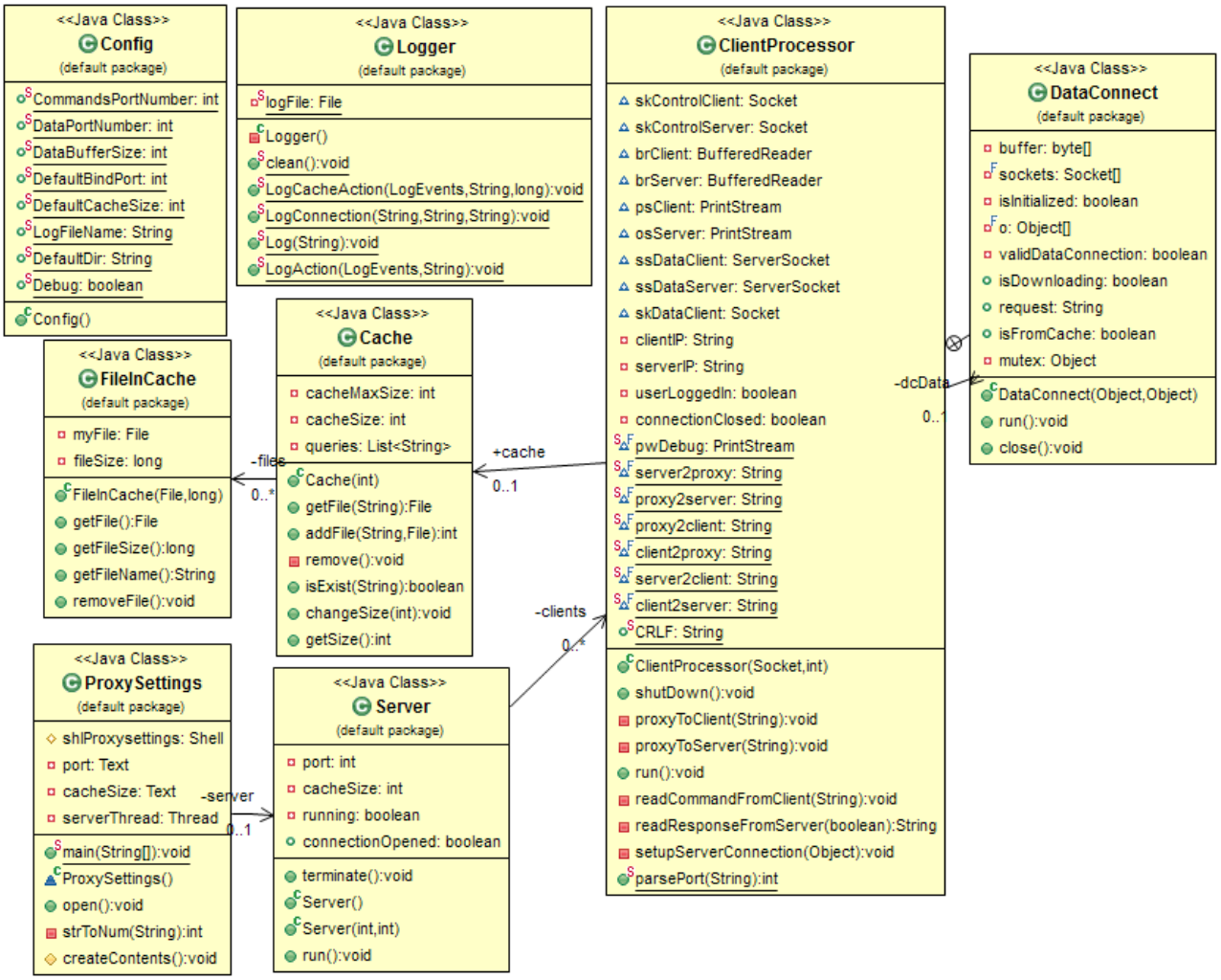
2. Разработка приложения.

2.1. Протокол для реализации.

Имя	Аргументы	Описание
USER	[имя]	Имя пользователя для входа на сервер.
PASS	[пароль]	Пароль
PORT	[сообщение]	Войти в активный режим. Например PORT 12,34,45,56,78,89. В отличие от пассивного режима для передачи данных сервер сам подключается к клиенту.
RETR	[сообщение]	Скачать файл. Перед RETR должна быть команда PASV или PORT.

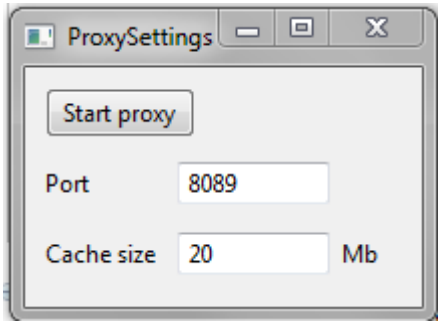
2.2.Архитектура приложения.

Настройки и запуск-остановка прокси осуществляются с помощью графического интерфейса ProxySettings. Для хранения кэша используются классы Cache и FileInCache. Прослушивание основного порта и подключение клиентов происходит в Server, обработка соединений для отдельных клиентов – ClientProcessor, содержащий внутренний класс для передачи файлов DataConnect. Запись логов осуществляется с помощью класса Logger.



3. Тестирование и результаты работы приложения.

Перед запуском сервера есть возможность настроить номер порта, прослушиваемого прокси-сервером, и объем кэша.



Запуск сервера и установка соединения:

[26.05.2016 20:09:42 MSK]: Server started
[26.05.2016 20:09:43 MSK]: [Parametres]: 8089 port, 1 Mbytes of cache.
[Client 127,0,0,1 thread]: P->C: 220 Java FTP Proxy Server (usage: USERID=user@site) ready.
[Client 127,0,0,1 thread]: P<-C: USER anonymous@ftp.funet.fi:21
[26.05.2016 20:10:09 MSK]: Connecting to ftp.funet.fi on port 21

Передача файлов:

Client 127,0,0,1 thread]: P->C: 200 PORT command successful.
[Client 127,0,0,1 thread]: S<-P: PORT 192,168,1,2,249,199
[Client 127,0,0,1 thread]: S->P: 200 PORT command successful
[Client 127,0,0,1 thread]: S<-P: RETR rfc-ref.txt
[Client 127,0,0,1 thread]: S->C: 150-Connecting to port 63943
[Client 127,0,0,1 thread]: S->C: 150 1449.6 kbytes to download
[Client 127,0,0,1 thread]: S->C: 226-File successfully transferred
[Client 127,0,0,1 thread]: S->C: 226 1.846 seconds (measured here), 0.77 Mbytes per second
[26.05.2016 20:11:16 MSK]: Sending file from ftp-server: RETR rfc-ref.txt
[26.05.2016 20:11:17 MSK]: Cache event. Cannot add to cache file (too large): rfc-ref.txt Size: 1484401bytes
[Client 127,0,0,1 thread]: P->C: 200 PORT command successful.
[Client 127,0,0,1 thread]: S<-P: PORT 192,168,1,2,249,210
[Client 127,0,0,1 thread]: S->P: 200 PORT command successful
[Client 127,0,0,1 thread]: S<-P: RETR rfc-retrieval.txt
[Client 127,0,0,1 thread]: S->C: 150 Connecting to port 63954
[Client 127,0,0,1 thread]: S->C: 226-File successfully transferred
[Client 127,0,0,1 thread]: S->C: 226 0.012 seconds (measured here), 67.56 Kbytes per second
[26.05.2016 20:11:25 MSK]: Sending file from ftp-server: RETR rfc-retrieval.txt
[26.05.2016 20:11:25 MSK]: Cache event. Added file: rfc-retrieval.txt Size: 848bytes
[Client 127,0,0,1 thread]: P->C: 200 PORT command successful.
[Client 127,0,0,1 thread]: S<-P: PORT 192,168,1,2,249,227
[Client 127,0,0,1 thread]: S->P: 200 PORT command successful
[26.05.2016 20:11:47 MSK]: Sending file to client from proxy: RETR rfc-retrieval.txt

Убедились, что клиент корректно подключается с помощью прокси к удаленному FTP-серверу, и что все необходимые команды транслируются серверу и клиенту. Также убедились, что файлы, размер которых превышает размер кэша, не сохраняются, при запросе к файлам, уже содержащимся в кэше, файл передается с прокси-сервера. В этом можно убедиться, изменив содержимое файла в кэше, и после передачи сравнив его с переданным на сторону клиента. Если в кэше недостаточно свободного места для добавления файла, старые файлы удаляются до тех пор, пока места не хватит.

4. Выводы.

Язык программирования Java является удобным инструментом для разработки сетевых приложений с использованием стэка протоколов TCP/IP, так как содержит стандартные компоненты, обеспечивающие простоту работы с данными протоколами, по сравнению с разработкой на языках C/C++.

В результате работы было разработано приложение, обеспечивающее функции прокси-сервера для протокола FTP. Прокси-сервер использует активный режим. Протокол FTP неудобен для программирования из-за наличия двух соединений, что усложняет логику многопоточного приложения. А также выявлены недостатки, связанные с тем, что для загрузки каждого файла требуется новое подключение, что снижает скорость.