Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-вычислительных систем
(КИБЭВС)

НЕПРОТИВОРЕЧИВОСТЬ И РЕПЛИКАЦИЯ

Отчёт по лабораторной работе №2 по дисциплине «Распределенные автоматизированные информационные системы»

Студент гр. 724:
_____ А.А. Крупина

Руководитель:
Старший преподаватель кафедры
КИБЭВС
_____ _____ Г.А. Праскурин
_____

Томск 2018

1 Введение

Цель работы – разработать систему, состоящую из нескольких серверов (2-3) и клиентов (2-3). На серверах хранятся реплицированные файлы. Один из серверов является главным - хранит актуальный список файлов и состояние файлов. Остальные (резервные) серверы "синхронизируются" с главным сервером.

При обращении к любому из серверов любой клиент должен получать (просматривать) актуальную версию файла. На главном сервере клиент может изменить содержимое файла, добавить или удалить файл.

При добавлении или удалении файла клиентом сервер должен "продвинуть" информацию об этих изменениях на другие серверы, чтобы клиент при работе со всеми серверами видел актуальный список файлов. Т.е. должен сообщить остальным серверам, на которых хранятся реплики, о том, что в файловой системе произошли изменения.

При изменении файла на главном сервере актуальное содержимое файла "извлекается" остальными серверами. Т.е. актуальное содержимое файла загружается на резервный сервер только тогда, когда клиент запрашивает изменённый файл.

Примечание. Файлы могут быть заменены на любые другие информационные структуры - массивы чисел, строки, списки и т.д.

2 Ход работы

2.1 Функционал клиентов

Код для клиентов:

```
import socket

command = (b'0',  # read
           b'1',  # add
           b'2',  # chn
           b'3',  # del
           b'q')  # close

sock = socket.socket()
sock.connect(('localhost', 9090))  # port 9090: main; 9000: srv;
sock.send(command[1])

data = sock.recv(1024).decode()
sock.close() if data is None else print(data)
```

Вывод для чтения:

{'2b76b53cd6b8b98ee742c926277ea86625228451': 'file_1', '7c69b3e909a5d0c05d4b446bba777d08b202f85c': 'file_2', 'c2431989fc6fc721adc35987943c2ac5eddff7d4': 'file_3'}

Вывод для добавления:

{'2b76b53cd6b8b98ee742c926277ea86625228451': 'file_1', '7c69b3e909a5d0c05d4b446bba777d08b202f85c': 'file_2', 'c2431989fc6fc721adc35987943c2ac5eddff7d4': 'file_3', 'b649b04dc19e6cf81e0e42663c4d0b9dc580909b': 'file_4'}

Вывод для изменения:

{'2b76b53cd6b8b98ee742c926277ea86625228451': 'file_1', '7c69b3e909a5d0c05d4b446bba777d08b202f85c': 'file_2', 'c2431989fc6fc721adc35987943c2ac5eddff7d4': 'file_3', '59989b7a4407a70a9360c79c28da67e26fe3f61e': 'file_4v2'}

Вывод для удаления:

{'2b76b53cd6b8b98ee742c926277ea86625228451': 'file_1', '7c69b3e909a5d0c05d4b446bba777d08b202f85c': 'file_2', 'c2431989fc6fc721adc35987943c2ac5eddff7d4': 'file_3'}

2.2 Главный сервер

Код для главного сервера:

```
from hashlib import sha1
import socket

list_files = ['file_1', 'file_2', 'file_3']


def get_files():
    hash_files = [sha1(str.encode(item)).hexdigest() for item in list_files]
    files = dict(zip(hash_files, list_files))
    return files


def del_files(index=3):
    if check_index(int(index)):
        del list_files[index]
        return get_files()
    else:
        print('Указан ошибочный индекс!')


def add_files(name_files='file_4'):
    list_files.append(name_files)
    return get_files()


def chn_name_files(index=3, name='file_4v2'):
    if check_index(int(index)):
        list_files.insert(index, str(name))
        del_files(index + 1)
        return get_files()
    else:
        print('Указан ошибочный индекс!')


def check_index(index):
    return True if 0 <= index < len(list_files) else False


if __name__ == '__main__':
    sock = socket.socket()
    sock.bind(('localhost', 9090))
    sock.listen(4)
    while True:
        try:
            conn, addr = sock.accept()
            data = conn.recv(1024).decode()
            if data == '1':
                result = add_files()
            elif data == '2':
                result = chn_name_files()
            elif data == '3':
                result = del_files()
            else:
                result = get_files()
            conn.send(bytes(str(result).encode()))
        except Exception as ex:
            print('Error: {}!'.format(ex.args[1]))
```

2.3 Резервный сервер

Код для резервного сервера:

```
import socket


def get_files():
    sock_main = socket.socket()
    sock_main.connect(('localhost', 9090))
    sock_main.send(b'0')
    backup_files = sock_main.recv(1024).decode()
    return backup_files


if __name__ == '__main__':
    sock = socket.socket()
    sock.bind(('localhost', 9000))
    sock.listen(4)
    while True:
        conn, addr = sock.accept()
        data = conn.recv(1024).decode()
        result = get_files()
        conn.send(bytes(str(result).encode()))
```

3 Заключение

В ходе работы была разработана система, состоящая из нескольких серверов (2-3) и клиентов (2-3) с возможностью репликации и синхронизации.