**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра компьютерных систем и программных технологий**

**Отчет**

**Дисциплина: сетевые технологии**

**Тема: изучение протоколов TCP/UDP на основе разработки**

**системы мгновенного обмена сообщений между пользователями**

**Студентка гр.43501/3: \_\_\_\_\_\_\_\_ Замотаева Ю.**

**Преподаватель: \_\_\_\_\_\_\_\_ Вылегжанина К.**

**Санкт-Петербург**

**2014**

Глава 1

Система мгновенного обмена сообщений между пользователями

1.1 Функциональные требования

1.1.1 Задание

Разработать приложение-клиент и приложение сервер, обеспечивающие функции мгновенного обмена сообщений между пользователями.

1.1.2 Основные возможности

**Серверное** приложение должно реализовывать следующие функции:

* Прослушивание определенного порта
* Обработка запросов на подключение по этому порту от клиентов
* Поддержка одновременной работы нескольких клиентов через механизм нитей
* Передача текстового сообщения одному клиенту
* Передача текстового сообщения всем клиентам
* Прием и ретрансляция входящих сообщений от клиентов
* Обработка запроса на отключение клиента
* Принудительное отключение указанного клиента

**Клиентское** приложение должно реализовывать следующие функции:

* Установление соединения с сервером
* Передача сообщения всем клиентам
* Передача сообщения указанному клиенту
* Прием сообщения от сервера с последующей индикацией
* Разрыв соединения
* Обработка ситуации отключения клиента сервером

1.1.3 Настройки приложений

Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого сервера и номера порта, используемого сервером.

1.2 Нефункциональные требования

1.2.1 Требования к реализации

Соединение начинает клиент, отправляя серверу фиксированную строку (пакет определенного формата), говорящую о начале соединения. Таким образом клиент удостоверяется в наличии сервера в сети. Далее происходит обмен необходимым набором сообщений (пакетами): регистрация пользователя, получение списка онлайн-пользователей, отправка и получение пользовательских сообщений (сервер принимает и ретранслирует входящие сообщения от клиентов). Также сервер контролирует наличие пользователя в сети, периодически посылая ему соответствующий запрос. Пользователь должен ответить на данный запрос, чтобы оставаться в сети и сервер не отключил его. Клиент может самостоятельно разорвать соединениe, посылая при этом серверу соответвующее увеломление.

1.2.2 Требования к надежности

Длина всего отправляемого пакета от клиента серверу (и наоборот) должна проверяться на максимальное значение, так мы защищаем сервера и клиента от «падения» при отправке слишком длинного пакета. Также в определенных полях пакета хранятся не только данные, но и информация об их размере (длина символов) для корректного их извлечения при получении пакета.

1.3 Накладываемые ограничения

• Ограничение на длину пакетов: максимальный размер пакета 256 символов.

Глава 2

Реализация для работы по протоколу TCP

2.1 Прикладной протокол

Рассмотрим общую структуру данных (пакет), с помощью которой сервер и клиент обмениваются сообщениями. Формат пакета представлен в следующей таблице:

|  |  |  |  |
| --- | --- | --- | --- |
| Длина всего пакета | Тип команды | Ip-адрес отправителя | Данные |
| 3 байта | 2 байта | 15 байт | 3 байта |

Тип команды – код операции:

|  |  |  |
| --- | --- | --- |
| 00 | ECHO\_REQUEST | клиент проверяет наличие сервера в сети |
| 01 | ECHO\_ANSWER | сервер отвечает клиенту о своем присутствии в сети |
| 02 | LOGIN\_REQUEST | клиент посылает свой логин серверу |
| 03 | LOGIN\_ANSWER | сервер отвечает клиенту о регистрации в сети |
| 04 | USERS\_REQUEST | клиент запрашивает список онлайн-пользователей |
| 05 | USERS\_ANSWER | сервер выдает клиенту список онлайн-пользователей |
| 06 | BAD\_PACKAGE\_ANSWER |  |
| 07 | UNAUTHORIZED\_ACTION |  |
| 08 | MESSAGE\_SEND | клиент посылает сообщение другому клиенту (через сервер) |
| 09 | MESSAGE\_INCORRECT\_NAME | сервер уведомляет клиента, что пользователь с запрашиваемым именем не найден |
| 10 | MESSAGE\_SUCCESS\_SENDED | сервер уведомляет клиента об успешной доставке сообщения другому клиенту |
| 11 | MESSAGE\_USER\_OFFLINE | сервер уведомляет клиента, что пользователь с запрашиваемым именем оффлайн |
| 12 | MESSAGE\_DELIVERY | сервер отправляет сообщение нужному клиенту от другого клиента |
| 13 | ADVERT\_MESSAGE | Рассылка рекламы сервером |
| 14 | ALIVE\_REQUEST | сервер спрашивает о наличии клиента в сети |
| 15 | ALIVE\_ANSWER | клиент отвечает серверу, что он онлайн |
| 16 | QUIT\_REQUEST | клиент уведомляет сервера о своем отключении |

Рассмотрим последовательность действий:

После запуска сервер начинает прослушивать определенный порт, он ждет подключение от клиента.

• Клиент хочет удостовериться о наличии сервера в сети, для этого он отправляет серверу соответствующий запрос (ECHO\_REQUEST) .

Рассмотрим содержимое передаваемого пакета в данном случае:

**021**00**192.168.001.004**1**,** где 021 – длина всего пакета;

00 – тип команды: ECHO\_REQUEST;

**192.168.001.004**– ip-адрес клиента;

1 – поле данных (любые данные)

• После принятия данного пакета, сервер отвечает клиенту о своем присутствии в сети (ECHO\_ANSWER).

Содержимое передаваемого пакета в данном случае: **021**01**192.168.001.002**1**,** где

021 – длина всего пакета;

01 – тип команды: ECHO\_ANSWER;

**192.168.001.002**– ip-адрес сервера;

1 – поле данных (любые данные)

• После того, как соединение установлено, клиент может вводить свой логин - на стороне клиента появляется сообщение:

Please enter login: julia

Клиент вводит свой логин.

Рассмотрим содержимое передаваемого пакета в данном случае: **027**02**192.168.001.004**05julia

027 – длина всего пакета;

02 – тип команды: LOGIN\_REQUEST;

**192.168.001.004**– ip-адрес клиента;

05julia – поле данных; 05-число символов (2байта) ; julia – имя

• После принятия пакета сервер анализирует данный логин – нет ли еще пользователя с таким же логином, далее отправляет клиенту ответ ( LOGIN\_ ANSWER).

Рассмотрим содержимое передаваемого пакета в случае, если логин уже занят: **022**02**192.168.001.002**00

022 – длина всего пакета;

02 – тип команды: LOGIN\_ ANSWER;

**192.168.001.002**– ip-адрес сервера;

00 – поле данных (логин занят)

Содержимое передаваемого пакета в случае, если логин свободен:

**022**02**192.168.001.002**11 **,** где

022 – длина всего пакета;

02 – тип команды: LOGIN\_ ANSWER;

**192.168.001.002**– ip-адрес сервера;

11 – поле данных (логин свободен)

На стороне клиента пояляется одно из сообщений: Ok. You are online.

Sorry! Please, enter login again.

• Клиент отправляет серверу запрос на получение списка клиентов онлайн (USERS\_REQUEST).

Содержимое передаваемого пакета:

**021**04**192.168.001.004**1**,** где

021 – длина всего пакета;

04 – тип команды: USERS\_REQUEST;

**192.168.001.004**– ip-адрес клиента;

1 – поле данных (любые данные)

• Сервер в ответ отправляет клиенту пакет со списком клиентов онлайн (USERS\_ANSWER):

01 user1

02 user2

• Далее клиент должен выбрать номер другого клиента и написать ему сообщение:

01 hello!

• Cервер получает данное сообщение, отсылает его требуемому клиенту (если тот существует и находится в сети ). Так начинается общение клиентов друг с другом (через сервер).

• Сервер контролирует наличие пользователя в сети, периодически посылая ему соответствующий запрос ALIVE\_REQUEST . Пользователь должен ответить на данный запрос ALIVE\_ANSWER, чтобы оставаться в сети и сервер не отключил его.

Клиент может самостоятельно разорвать соединениe, посылая при этом серверу соответвующее увеломление QUIT\_REQUEST.

Sequence Diagram протокола представлена на рис.1

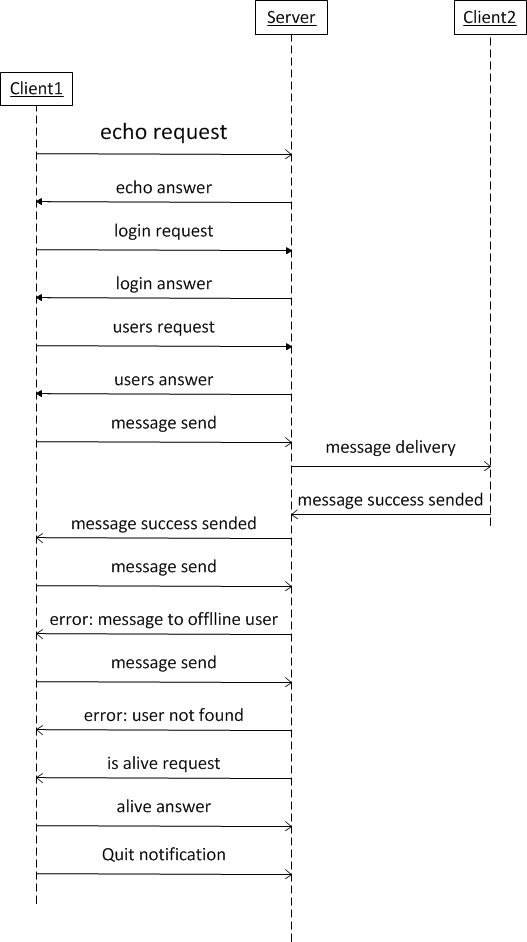


Рис.1 Sequence Diagram.