Министерство цифрового развития, связи и массовых коммуникаций

Бурятский институт инфокоммуникаций (филиал) Федерального государственного бюджетного образовательного учреждения высшего образования «Сибирский государственный университет телекоммуникаций   
и информатики» в г. Улан-Удэ

|  |
| --- |
| Кафедра Информатики и |
| вычислительной техники |
| Допустить к защите |
| Зав.каф. И.Б.Елтунова |
|  |

|  |
| --- |
| КУРСОВАЯ РАБОТА |
| Сетевой чат (TCP IPv4) |
| Пояснительная записка |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | /Котлыков Д.В. |
|  |  |  |  |  |
| Факультет |  | ИВТ |  | /Группа И-101 |
|  |  |  |  |  |
| Руководитель |  |  |  | /Жигжитов М.В. |
|  |  |  |  |  |

г. Улан-Удэ, 2022г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc134029480)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc134029481)

[1.1 Анализ предметной области. 5](#_Toc134029482)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 7](#_Toc134029483)

[2.1 Реализация сервера сетевого чата. 7](#_Toc134029484)

[2.2 Реализация клиента сетевого чата. 11](#_Toc134029485)

[2.2 Тестирование клиента и сервера сетевого чата 13](#_Toc134029486)

[ЗАКЛЮЧЕНИЕ 16](#_Toc134029487)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 18](#_Toc134029488)

# ВВЕДЕНИЕ

Вопросы общения интересовали людей всегда. Для того, чтобы можно было обмениваться информацией не только при личной встрече, но и на огромных расстояниях, люди изобретали всё новые и новые технические средства, организовывали почтовые системы, протягивали кабели через континенты и океаны, запускали спутники связи. С развитием информационных технологий стали возможным еще более глобальные коммуникации. Историческим «докомпьютерным» предшественником чатов, несомненно, был телефон. Ни почта, ни телеграф не позволяли общаться в реальном времени и не были доступны в домашней обстановке. Изобретение и распространение телефона по планете вызвало настоящую революцию в средствах и способах общения. Возможность поговорить с собеседником на другой стороне Земли казалась настоящим чудом.

Программы для обмена текстовыми строками, несмотря на простоту самой идеи, появились не сразу. Базовым для них является единый протокол — TCP. TCP — один из основных протоколов передачи данных Интернета, предназначенный для управления передачей данных в сетях и подсетях TCP/IP. Выполняет функции протокола транспортного уровня в стеке протоколов IP. Интерфейсом прикладного программирования для сетевых приложений TCP/IP являются сокеты. Сокет — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения. Сокет — это комбинация IP-адреса и порта. Каждый процесс может создать слушающий сокет (серверный сокет) и привязать его к какому-нибудь порту операционной системы. Слушающий процесс обычно находится в цикле ожидания, то есть просыпается при появлении нового соединения.

Для нас это означает, что сервер будет слушать какой-либо порт и после подключения клиента, будем производить с ним какие-то действия.

Основанием для разработки программы послужила идея проверить свои возможности по написанию программы, работающей с сетью, такой как сетевой чат.

Чат — средство обмена сообщениями по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение. Характерной особенностью является коммуникация именно в реальном времени или близкая к этому, что отличает чат от форумов и других «медленных» средств. То есть, если на форуме можно написать вопрос и ждать, пока кто-нибудь посчитает нужным на него ответить (в то же время, можно получить и несколько ответов сразу от разных пользователей), то в чате общение происходит только с теми, кто присутствует в нём в настоящий момент, а результаты обмена сообщениями могут и не сохраняться.

Под словом чат обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ мгновенного обмена сообщениями, например, XMPP, ICQ или даже SMS. Основная цель — создания сетевого чата — мгновенный обмен сообщениями. Использование данного программного продукта поможет сократить время выполнения какой-либо работы или просто хорошо провести время, общаясь в кругу друзей. Сетевой чат может быть использован как для малого круга людей, т.е. в домашних условиях, так и на производстве небольшого предприятия.

Целью данной курсовой работы является закрепление знаний по использованию библиотеки socket в языке программирования python. Задачей курсовой работы является реализация сетевого чата с использованием библиотеки socket на языке программирования python.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Анализ предметной области.

Для реализации клиент сервер игры был выбран сетевой чат. Чтобы реализовать сетевой чат между клиентом и севером, нам необходимо написать код двух программ, а именно клиента и сервера. Для упрощения реализации будут использованы UML диаграммы.

UML диаграммы могут быть полезны для разработки игр, особенно в контексте сетевых игр. Они могут помочь разработчикам лучше понимать структуру и поведение системы, а также улучшить коммуникацию между членами команды разработки.

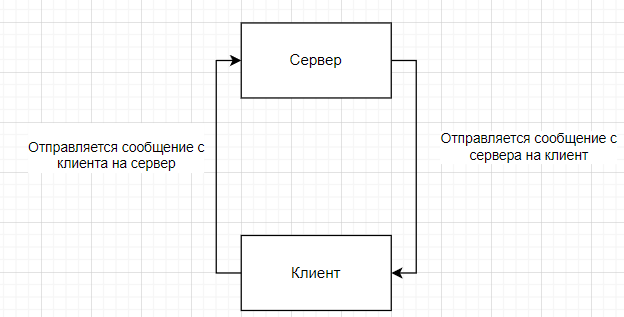


Рис. 1. Диаграмма компонентов.

На рис. 1 представлена диаграмма компонентов необходимых для реализации сетевого чата. Из представленной выше диаграммы видно, что необходимо реализовать два компонента, а именно клиент и сервер. Клиент должен иметь возможность отправлять сообщение на сервер и так же выводить сообщение сервера у себя. Сервер должен тоже самое что и клиент.

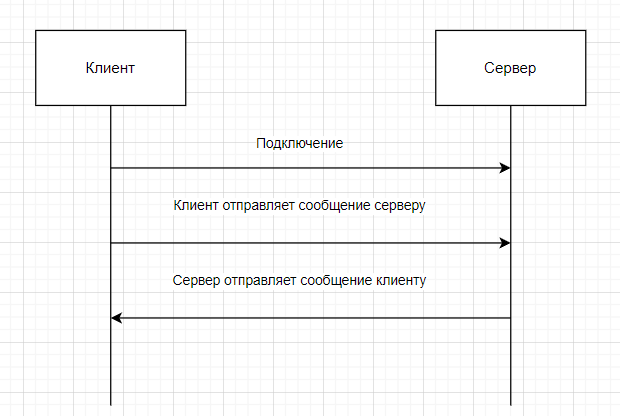


Рис.2. Диаграмма последовательностей.

На рис.2 представлена диаграмма последовательностей действий программ. Из диаграммы видно, что сначала происходит подключение клиента к серверу. После подключения клиента к серверу, сервер может отправлять сообщение клиенту, и клиент может отправлять сообщение серверу.

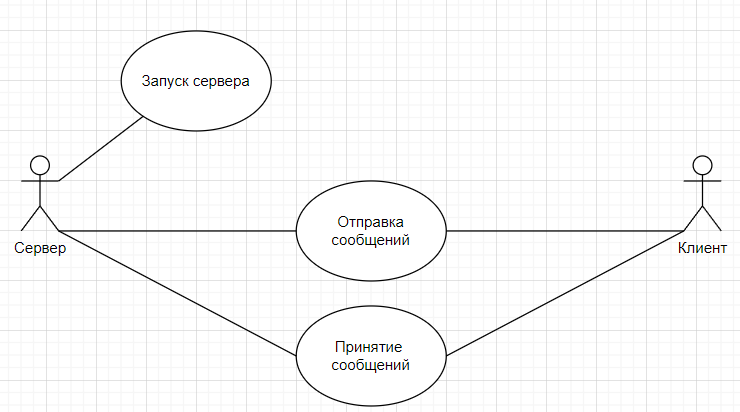


Рис.3. Диаграмма использования.

На рис.3 продемонстрирована диаграмма использования игры. Владелец сервера может запустить сервер для чата по сети, отправить сообщение и принять сообщение. Клиент, может присоединиться к серверу через клиентскую программу, отправлять сообщения и принимать их.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Реализация сервера сетевого чата.

Для реализации сервера был выбран язык программирования python и библиотека socket.

import socket

# Создаем сокет

serverSocket = socket(AF\_INET, SOCK\_STREAM)

Выше показано подключение библиотеки socket, а также создание сокета с помощью команды socket.socket.

# Устанавливаем адрес и порт сервера

HOST = 'localhost'

serverPort = 6789

serverSocket.bind((HOST, serverPort))

Выше представлена часть кода, которая связывает ip-адрес и порт сервера с помощью команды bind.

# Ожидаем подключения клиентов

serverSocket.listen(1)

print('Сервер чата готов к подключению к чат-клиенту')

Выше продемонстрировано подключение двух клиентов к серверу с помощью команды listen.

# Функция для получения сообщения от клиента

def recv\_from\_client(conn):

global FLAG

try:

# Получает сообщение с запросом от клиента

while True:

if FLAG == True:

break

message = conn.recv(1024).decode()

# если 'q' получен от клиента, сервер завершает работу

if message == 'q':

conn.send('q'.encode())

print('Закрытие соеденения')

conn.close()

FLAG = True

break

print('Клиент: ' + message)

except:

conn.close() # Закрываем соединение с игроками

# функция для отправки сообщения клиенту

def send\_to\_client(conn):

global FLAG

try:

while True:

if FLAG == True:

break

send\_msg = input('')

# сервер может предоставить "q" в качестве входных данных, если он хочет завершить работу

if send\_msg == 'q':

conn.send('q'.encode())

print('Закрытие соеденения')

conn.close()

FLAG = True

break

conn.send(send\_msg.encode())

except:

conn.close()

Ваше представлен полный код принятия и отправки сообщений.

После завершения цикла сервер закрывает общение с клиентом с помощью команды close.

# Бесконечный цикл для ожидания подключений клиентов

# это основная функция

def main():

threads = []

global FLAG

HOST = 'localhost'

serverPort = 6789

# Создайте сокет TCP-сервера

#(AF\_INET используется для протоколов IPv4)

#(SOCK\_STREAM используется для TCP)

serverSocket = socket(AF\_INET, SOCK\_STREAM)

# Привязать сокет к адресу сервера и порту сервера

serverSocket.bind((HOST, serverPort))

# Прослушивайте не более 1 соединения одновременно

serverSocket.listen(1)

# Сервер должен быть запущен и прослушивать входящие подключения

print('Сервер чата готов к подключению к чат-клиенту')

connectionSocket, addr = serverSocket.accept()

print('Сервер подключен к чат-клиенту\n')

t\_rcv = threading.Thread(target=recv\_from\_client, args=(connectionSocket,))

t\_send = threading.Thread(target=send\_to\_client, args=(connectionSocket,))

# вызовите функцию для получения сообщений сервером

#recv\_from\_server(clientSocket)

threads.append(t\_rcv)

threads.append(t\_send)

t\_rcv.start()

t\_send.start()

t\_rcv.join()

t\_send.join()

# закрытие serversocket перед выходом

print('выходящий')

serverSocket.close()

#Завершите работу программы после отправки соответствующих данных

sys.exit()

Выше продемонстрирована финальная часть кода сервера, а именно основная функция, где вызывается функции отправки и принятия сообщений, завершение работы программы, а также подключение клиента к серверу.

## 2.2 Реализация клиента сетевого чата.

Поскольку для реализации сервера сетевого чата были выбраны язык программирования python и библиотека socket, то и для клиента будут использованы тот же язык программирования и та же библиотека.

import socket

# Создаем сокет

clientSocket = socket(AF\_INET, SOCK\_STREAM)

# Устанавливаем адрес и порт сервера

HOST = 'localhost'

PORT = 6789

# Подключаемся к серверу

clientSocket = socket(AF\_INET, SOCK\_STREAM)

Выше представлено начало кода клиента, которое по большей части совпадает с началом кода сервера. Сначала подключается библиотека socket, после создаём сокет с помощью команды socket.socket, затем устанавливаем адрес и порт сервера. Далее код клиента отличается от кода сервера, так как вместо связывания ip-адреса и порта, клиент подключается к серверу с помощью команды connect.

# функция для получения сообщения от сервера

def recv\_from\_server(clsock):

global FLAG

while True:

data = clsock.recv(1024).decode()

if data == 'q':

print('Замыкающее соединение')

FLAG = True

break

print('Сервер: ' + data)

Выше продемонстрирована часть кода, которая получает сообщение серверу.

# функция для отправки сообщения от клиента

def send\_to\_server(clsock):

global FLAG

while True:

if FLAG == True:

break

send\_msg = input('')

clsock.sendall(send\_msg.encode())

# Получаем сообщение о начале игры

start\_message = client\_socket.recv(1024).decode()

print(start\_message)

# Запрашиваем выбор камня, ножниц или бумаги у игрока

while True:

choice = input('Сделайте свой выбор (камень/ножницы/бумага): ')

# Отправляем выбор серверу

client\_socket.sendall(choice.encode())

# Получаем результаты игры от сервера

result\_message = client\_socket.recv(1024).decode()

print(result\_message)

# Если есть победитель, выходим из цикла

if 'Победил: Игрок' in result\_message in result\_message:

break

# Закрываем соединение с сервером

client\_socket.close()

А основная часть кода такая же, как и на сервере, кроме функции connect.

## 2.2 Тестирование клиента и сервера сетевого чата

Так как были написаны и клиентская и серверная программы для сетевого чата необходимо провести тестирование этих программ.

Проверка работы клиента без запущенного сервера.

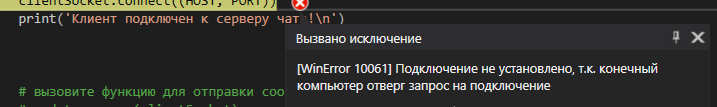


Рис. 4. Результат тестирования клиента без сервера.

На рис. 4 продемонстрирован результат работы программы клиента без запущенного сервера. Ожидаемым результатом была ошибка, так как без запущенного сервера клиент не сможет подключится к сокету, в связи с чем вообще не сможет запустится. Фактический результат тестирования полностью совпадает с ожидаемым, то есть клиентская программа не может запустится без сервера.

Проверка работы сервера без запущенного клиента.

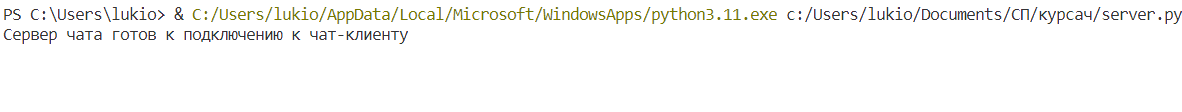


Рис. 5. Результат тестирования сервера без клиента.

На рис. 5 изображен результат работы программы сервера без запущенного клиента. Ожидаемым результатом был обычный запуск программы сервера без ошибок, так как сервер никак не зависит от клиента, в связи с чем ничто не должно помешать запуску сервера. Фактический результат совпадает с ожидаемым и сервер запустился без ошибок, выдав сообщение об ожидании подключении клиента.

Проверка работы клиента при запущенном сервере

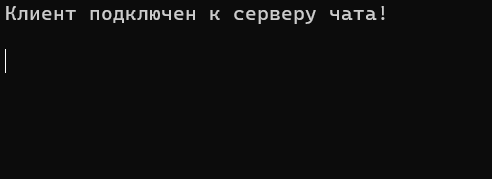


Рис. 6. Результат тестирования клиента с сервером.

На рис. 6 представлен результат работы программы клиента с запущенным сервером. Ожидаемым результатом был обычный запуск клиентской программы без ошибок, так как сервер был запущен, а значит был создан сокет, к которому смогла подключится клиентская программа. Фактический результат соответствует ожидаемому, так как клиентская программа запустилась без ошибок, что означает, что клиент смог подключится к серверу с помощью сокета.

Проверка работы отправки сообщения между клиентом и сервером.

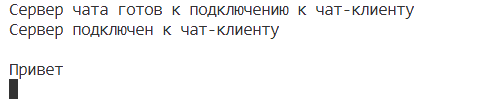


Рис. 7. Результат отправки сообщения с сервера.

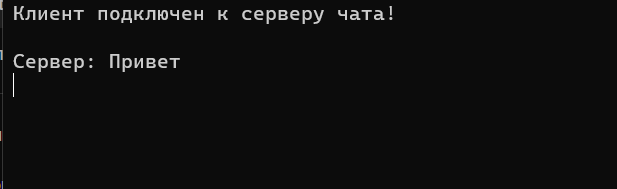


Рис. 8. Результат отправки сообщения с первого клиента.

На рис. 7 и рис. 8 показаны результаты тестирования отправки сообщения с сервера на клиент. В результате было отправлено сообщение «Сервер: Привет», где Сервер это от кого поступило сообщение.

Проверка работы программ при результате отправки сообщения с клиента на сервер.

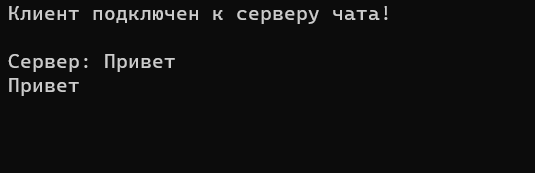


Рис. 9. Результат работы программы клиента.

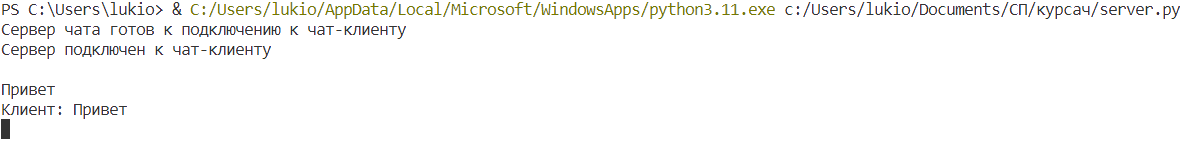


Рис. 10. Результат работы программы сервера.

Как видим на рис.9 и рис. 10 что было отправлено сообщение с клиента «Привет» и сервер благополучно получил от клиента сообщение. Фактический результат полностью совпадает с ожидаемым, что подтверждает правильность работы условий прописанных в коде программ.

# ЗАКЛЮЧЕНИЕ

В настоящее время удаленное общение становится все более популярным и востребованным. Сетевые чаты являются одним из наиболее распространенных способов общения в режиме реального времени. В ходе выполнения курсовой работы была разработана программа, позволяющая пользователям обмениваться сообщениями чата на основе протокола TCP IPv4.

Основной целью работы было изучение принципов работы сокетов и протокола TCP, а также способов организации чата. Для достижения этой цели были поставлены следующие задачи: изучить основные принципы работы сокетов и протокола TCP; разработать архитектуру программы; реализовать функции отправки и получения сообщений; провести тестирование программы.

Разработанный чат позволяет пользователям общаться в режиме реального времени, обмениваться сообщениями. Были реализованы функции отправки и получения сообщений.

Разработанный чат может быть использован как для общения в рамках организации, так и для общения между друзьями или коллегами. Он является удобным и простым в использовании инструментом для удаленного общения.

Однако, в процессе разработки программы были выявлены некоторые недостатки, такие как отсутствие защиты от DDoS-атак и возможность перехвата сообщений злоумышленниками.

Эти проблемы могут быть решены путем доработки программы и введения дополнительных механизмов защиты. Например, можно использовать технологию SSL/TLS для шифрования данных, а также реализовать механизмы защиты от DDoS-атак.

В целом, выполнение курсовой работы позволило углубить знания в области сетевых технологий и разработки программного обеспечения, а также получить практические навыки работы с сокетами и протоколом TCP IPv4. Разработанный чат может быть использован как основа для создания более сложных систем удаленного общения, которые будут учитывать требования безопасности и защиты от атак.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Таненбаум Э. С. Компьютерные сети. 5-е издание. – СПб.: Питер, 2013.

2. Сергеев Ю. В., Швецова Е. В. Программирование сокетов в Linux. – М.: ДМК Пресс, 2016.

3. Розенблатт М. Язык программирования Python. – М.: ДМК Пресс, 2015.

4. Герасимов С. В. Сетевое программирование на языке C#. – СПб.: БХВ-Петербург, 2014.

5. Красильщиков А. С., Крылов А. В. Программирование сокетов на языке Java. – М.: Издательство "Лань", 2016.

6. Лав М., Нил П., Садье Р. PHP и MySQL. Создание интернет-магазинов, блогов и социальных сетей. – М.: ДМК Пресс, 2017.

7. Рождественский И. А., Шевчук А. С. Программирование на языке С++. – М.: Издательство "Лань", 2018.

8. Терещенко А. В., Лебедева Е. В. Основы сетевых технологий. – М.: Издательство "Экзамен", 2014.

9. Барсегян А. А. Сетевые технологии. – М.: Издательство "Бином", 2016.  
10. Кант Р. А., Макдональд Дж. Программирование на языке Perl. – М.: ДМК Пресс, 2017.

11. Серебряков В. А., Кузнецова Л. В. Сетевое программирование на языке Python. – СПб.: Питер, 2015.

12. Каргин В. В., Королев Д. А., Левашов В. А. Программирование на языке Ruby. – М.: Издательство "Лань", 2015.

13. Литвиненко Е. А., Колесникова А. В. Сетевые технологии в современном бизнесе. – М.: Издательство "Экономика", 2016.

14. Столяров Д. А., Максимов С. Ю. Сетевое программирование на языке Delphi. – М.: ДМК Пресс, 2016.

15. Гордеев Ю. В., Лисицын А. И., Рогожин А. В. Программирование сокетов на языке Python 3. – М.: Издательство "Лань", 2017.