Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

Нижегородский государственный архитектурно-

строительный университет (ННГАСУ)

*Факультет инженерно-экологических систем и сооружений  
Кафедра информационных систем и технологий*

**КУРСОВАЯ РАБОТА**  
по дисциплине: «Инфокоммуникационные системы и сети»

**На тему: «Создание онлайн-игры»**

Выполнил студент 3 курса гр.ИС-29: Голышкин П.А.

Проверил: старший преподаватель: Морозов Н.С.

**Содержание**

[**Цель Работы** 3](#_Toc124962870)

[**Задачи** 4](#_Toc124962871)

[**Теория** 5](#_Toc124962872)

[**IP-адресация** 5](#_Toc124962873)

[**Протокол связи TCP** 6](#_Toc124962874)

[**Заголовок сегмента TCP** 6](#_Toc124962875)

[**Механизм действия протокола** 7](#_Toc124962876)

[**Топология сети тестирования** 7](#_Toc124962877)

[**Листинг программы** 9](#_Toc124962878)

[**Серверная часть (server.py)** 9](#_Toc124962879)

[**Клиентская часть (client.py)** 11](#_Toc124962880)

[**Заключение** 15](#_Toc124962881)

# **Цель Работы**

Написать на языке программирования Python сетевой чат с интегрированной в него игрой Камень-Ножницы-Бумага.

Заготовка игры представлена по [ссылке](https://github.com/kit8nino/2022-BigData/blob/main/__/main.py) (<https://github.com/kit8nino/2022-BigData/blob/main/__/main.py>)

Необходимо написать сервер и клиент, игра не должна мешать работе основного функционала чата.

# **Задачи**

1. Изучение маршрутизации, стека TCP/IP
2. Настройка протокола транспортного уровня
3. Настройка сокетов серверной и клиентской части игры

# **Теория**

## **IP-адресация**

Каждый компьютер в сети TCP/IP имеет адреса трех уровней:

* Локальный адрес узла, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети - это МАС-адрес сетевого адаптера или порта маршрутизатора, например, 11-А0-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано. Для всех существующих технологий локальных сетей МАС-адрес имеет формат 6 байтов: старшие 3 байта - идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем. Для узлов, входящих в глобальные сети, такие как Х.25 или frame relay, локальный адрес назначается администратором глобальной сети.
* IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Network Information Center, NIC), если сеть должна работать как составная часть Internet. Обычно провайдеры услуг Internet получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла - гибкое, и граница между этими полями может устанавливаться весьма произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

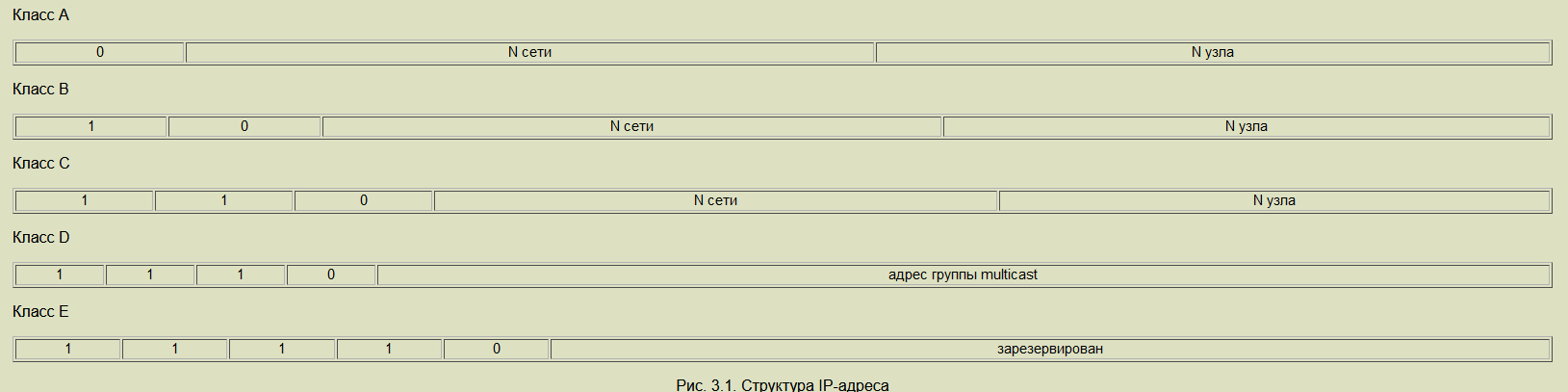
* Символьный идентификатор-имя, например, SERV1.IBM.COM. Этот адрес назначается администратором и состоит из нескольких частей, например, имени машины, имени организации, имени домена. Такой адрес, называемый также DNS-именем, используется на прикладном уровне, например, в протоколах FTP или telnet.

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например:

128.10.2.30 - традиционная десятичная форма представления адреса,

10000000 00001010 00000010 00011110 - двоичная форма представления этого же адреса.

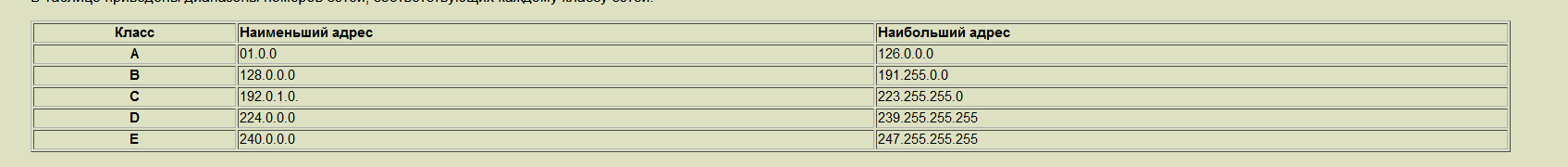
Структура IP-адреса.



Адрес состоит из двух логических частей - номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая к номеру узла, определяется значениями первых битов адреса:

* Если адрес начинается с 0, то сеть относят к классу А, и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса А имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей, о чем будет сказано ниже.) В сетях класса А количество узлов должно быть больше 216 , но не превышать 224.
* Если первые два бита адреса равны 10, то сеть относится к классу В и является сетью средних размеров с числом узлов 28 - 216. В сетях класса В под адрес сети и под адрес узла отводится по 16 битов, то есть по 2 байта.
* Если адрес начинается с последовательности 110, то это сеть класса С с числом узлов не больше 28. Под адрес сети отводится 24 бита, а под адрес узла - 8 битов.
* Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес - multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес.
* Если адрес начинается с последовательности 11110, то это адрес класса Е, он зарезервирован для будущих применений.

В таблице приведены диапазоны номеров сетей, соответствующих каждому классу сетей.



## **Протокол связи TCP**

### **TCP – Transfer Control Protocol**. Протокол управления передачей. Он служит для   обеспечения и установление надежного соединения между двумя устройствами и надежную передачу данных. При этом протокол TCP контролирует оптимальный размер передаваемого пакета данных, осуществляя новую посылку при сбое передачи.

Функциональность TCP/IP включает в себя стек протоколов, разделенный на четыре уровня:

1. Прикладной уровень – предоставляет приложениям стандартизированный обмен данными. Его протоколы включают HTTP, FTP, протокол POP3 для обмена электронной почтой и простой протокол управления сетью. На прикладном уровне полезной нагрузкой являются фактические данные приложения.
2. Транспортный уровень – отвечает за поддержание сквозной связи в сети. TCP управляет связью между хостами и обеспечивает управление потоком, передачу нескольких потоков и надежность. К транспортным протоколам относятся TCP и протокол пользовательских блоков информации, который иногда используется вместо TCP.
3. Сетевой уровень или интернет-уровень – управляет пакетами и соединяет независимые сети для передачи данных через границы сети. Протоколами сетевого уровня являются IP и протокол управляющих сообщений интернета, который используется для отчетов об ошибках.
4. Физический уровень или уровень сетевого интерфейса (канальный уровень) – состоит из протоколов, работающих на сетевом компоненте соединения узлов или хостов в сети. Протоколы этого уровня включают Ethernet для локальных сетей и протокол разрешения адресов.

4-уровневая модель позволяет использовать TCP/IP для широкого круга задач:

* удаленный вход в систему по сети;
* интерактивная передача файлов;
* доставка электронной почты;
* доступ к веб-страницам;
* удаленный доступ к файловой системе хост-сервера.

В TCP/IP подробно описаны основные протоколы или методы связи на каждом уровне по мере прохождения информации.

### **Заголовок сегмента TCP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бит** | **0 – 3** | **4 – 6** | **7 – 15** | **16 – 31** |
| **0** | Порт источника, **Source Port** | | | Порт назначения, **Destination Port** |
| **32** | Порядковый номер, **Sequence Number (SN)** | | | |
| **64** | Номер подтверждения, **Acknowledgement Number (ACK SN)** | | | |
| **96** | Длина заголовка, (**Data offset**) | Зарезервировано | Флаги | Размер окна, **Window Size** |
| **128** | Контрольная сумма, **Checksum** | | | Указатель важности, **Urgent Point** |
| **160** | Опции (необязательное, но используется почти всегда) | | | |
| **160/192+** | Данные | | | |

### **Механизм действия протокола**

В отличие от традиционной альтернативы — UDP, который может сразу же начать передачу пакетов, TCP устанавливает соединения, которые должны быть созданы перед передачей данных. TCP-соединение можно разделить на 3 стадии:

* Установка соединения
* Передача данных
* Завершение соединения

## **Топология сети тестирования**

**Сетевая топология** — это конфигурация графа, вершинам которого соответствуют конечные узлы сети (компьютеры и коммуникационное оборудование (маршрутизаторы), а рёбрам — физические или информационные связи между вершинами.

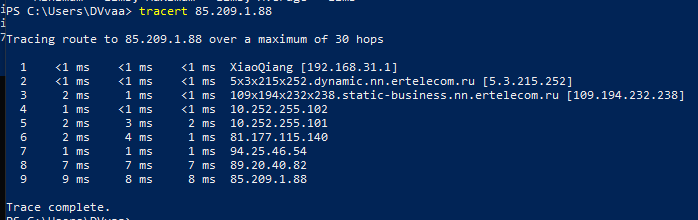


Рис 1. Трассирование устройства вне локальной сети

# **Листинг программы**

## **server.py**

import socket

from enum import IntEnum

class Action(IntEnum):

Rock = 0

Paper = 1

Scissors = 2

Lizard = 3

Spock = 4

victories = {

Action.Scissors: [Action.Lizard, Action.Paper],

Action.Paper: [Action.Spock, Action.Rock],

Action.Rock: [Action.Lizard, Action.Scissors],

Action.Lizard: [Action.Spock, Action.Paper],

Action.Spock: [Action.Scissors, Action.Rock]

}

def determine\_winner(user1\_action, user2\_action):

defeats = victories[user1\_action]

if user1\_action == user2\_action:

data = (f"Both players selected {user1\_action.name}. It's a tie!")

elif user2\_action in defeats:

data = (f"{user1\_action.name} beats {user2\_action.name}!")

else:

data = (f"{user2\_action.name} beats {user1\_action.name}!")

return data

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sock.bind(('localhost', 6925))

clients = {}

game = False

players = {}

choice = []

print('Start Server')

while True:

data, addres = sock.recvfrom(1024)

print(addres[0], addres[1])

data = data.decode('utf-8')

if addres not in clients.keys():

clients.update({addres: data})

print("new connected")

print(type(data))

data += " connected to this chat"

print(data)

try:

for client in clients.keys():

if client == addres:

continue

print(players)

print(choice)

if (data == "exit"):

print(clients.pop(addres))

print('deleted')

if (len(players) < 2):

if (game and data == "Y" and not (client in players.keys())):

players.update({client: False})

data = "Start Game \nChoose num: 0-Rock 1-Paper 2-Scissors 3-Lizard 4-Spock"

for player in players.keys():

data = 'Server: ' + data

sock.sendto(data.encode('utf-8'), player)

continue

elif (game and data == "N"):

data = "Game is over"

game = False

players.clear()

if (data == "/startRPS" and len(clients) > 1):

game = True

data = "Сhoose whether you will play Y/N"

players.update({client: False})

elif (len(players) == 2 and game and not players[addres]):

choice.append(data)

sock.sendto((f"Your choice is {data} - {Action(int(data)).name}").encode('utf-8'), addres)

players[addres] = True

data = "I made a choice "

elif (data == "/stop"):

game = False

players.clear()

choice.clear()

sock.sendto((str(clients[addres]) + ': ').encode('utf-8') + data.encode('utf-8'), client)

if (not False in players.keys() and len(choice) == 2):

data = determine\_winner(Action(int(choice[0])), Action(int(choice[1])))

for player in players.keys():

data = 'Server: ' + data + "\n Game is over"

sock.sendto(data.encode('utf-8'), player)

game = False

players.clear()

choice.clear()

except:

continue

Серверная часть

## **client.py**

import socket

import sys

import threading

def read\_sok():

while True:

data = sor.recv(1024)

print(data.decode('utf-8'))

server = 'localhost', 6925

alias: str = input() # Вводим наш псевдоним

sor = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sor.bind(('', 0)) # Задаем сокет как клиент

sor.sendto((alias).encode('utf-8'), server) # Уведомляем сервер о подключении

potok = threading.Thread(target=read\_sok)

potok.start()

while True:

mensahe = input()

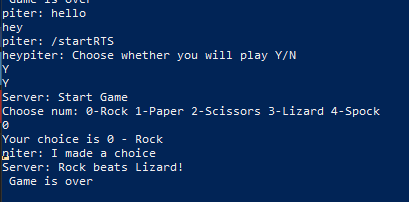
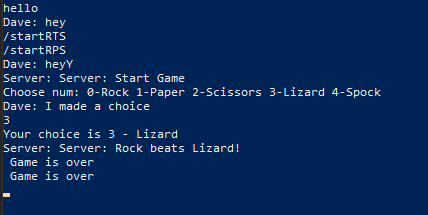
sor.sendto((mensahe).encode('utf-8'), server)

if (mensahe == "exit"):

sys.exit()

Клиентская часть

# **Пример работы**



# **Заключение**

В данной курсовой работе был разработан чат на сокетах на языке Python с интерпретированной игрой "Камень-Ножницы-Бумага". Целью работы было изучение маршрутизации, стека TCP/IP, протоколов транспортного уровня и настройка сокетов серверной и клиентской части игры.

В ходе работы были использованы сокеты для создания соединения между клиентом и сервером, а также были реализованы протоколы TCP и IP для обмена данными между клиентом и сервером. Была реализована игра "Камень-Ножницы-Бумага", которая позволяет двум игрокам играть друг с другом в реальном времени через чат.

В результате работы был разработан функционирующий чат на сокетах с интерпретированной игрой "Камень-Ножницы-Бумага", который позволяет двум игрокам играть друг с другом в реальном времени через чат, и который был использован для демонстрации изученных процессов маршрутизации, стека TCP/IP, протоколов транспортного уровня и настройки сокетов. Работа показала, что изучение этих технологий и их применение в практическом проекте может привести к разработке функциональных и полезных приложений.

# **Список литературы**

1. "Python. Создание интернет-приложений" А. Данилов и Ю. Карпенко
2. "Python. Сетевое программирование" Д. Бейли
3. "TCP/IP протоколы и сокеты" Д. Роули