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: Крупин В. Н.

Проверил: старший преподаватель: Морозов Н. С.

Нижний Новгород

2023 год

**Содержание**

[Цель 3](#_Toc125072164)

[Задачи 4](#_Toc125072165)

[Теория 5](#_Toc125072166)

[IP – адресация 5](#_Toc125072167)

[Классы IP-адресов 6](#_Toc125072168)

[Бесклассовая адресация 6](#_Toc125072169)

[Особые IP-адреса 6](#_Toc125072170)

[Динамические IP-адреса 7](#_Toc125072171)

[Протокол связи TCP 8](#_Toc125072172)

[Протокол TCP и его особенности 8](#_Toc125072173)

[Алгоритм работы TCP 9](#_Toc125072174)

[Заголовок TCP пакета 9](#_Toc125072175)

[Топология сети тестирования 10](#_Toc125072176)

[Интерфейс приложения 11](#_Toc125072177)

[Листинг программы 12](#_Toc125072178)

[Серверная часть 12](#_Toc125072179)

[Клиентская часть 13](#_Toc125072180)

[Заключение 17](#_Toc125072181)

[Список литературы 18](#_Toc125072182)

# **Цель**

Написать на языке программирования Python сетевой чат с интегрированной в него игрой Камень-Ножницы-Бумага. В игре должен быть чат, который не мешает остальной работе программы.

# **Задачи**

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

# **Теория**

## **IP – адресация**

IP-адрес (aй-пи адрес, сокращение от англ. Internet Protocol Address) — уникальный идентификатор (адрес) устройства (обычно компьютера), подключённого к локальной сети или интернету.

IP-адрес представляет собой 32-битовое (по версии IPv4) или 128-битовое (по версии IPv6) двоичное число. Удобной формой записи IP-адреса (IPv4) является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками, например, 192.168.0.1. (или 128.10.2.30 — традиционная десятичная форма представления адреса, а 10000000 00001010 00000010 00011110 — двоичная форма представления этого же адреса).

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень протокола IP передаёт пакеты между сетями. IP-адрес назначается администратором во время конфигурирования компьютеров и маршрутизаторов.

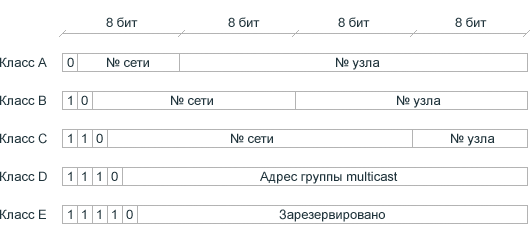
IP-адрес состоит из двух частей: номера сети и номера узла. В случае изолированной сети её адрес может быть выбран администратором из специально зарезервированных для таких сетей блоков адресов (192.168.0.0/16, 172.16.0.0/12 или 10.0.0.0/8). Если же сеть должна работать как составная часть Интернета, то адрес сети выдаётся провайдером либо pегиональным интернет-регистратором (Regional Internet Registry, RIR). Всего существует пять RIR: ARIN, обслуживающий Северную Америку; APNIC, обслуживающий страны Юго-Восточной Азии; AfriNIC, обслуживающий страны Африки; LACNIC, обслуживающий страны Южной Америки и бассейна Карибского моря; и RIPE NCC, обслуживающий Европу, Центральную Азию, Ближний Восток. Региональные регистраторы получают номера автономных систем и большие блоки адресов у ICANN, а затем выдают номера автономных систем и блоки адресов меньшего размера локальным интернет-регистраторам (Local Internet Registries, LIR), обычно являющимся крупными провайдерами.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Маршрутизатор по определению входит сразу в несколько сетей. Поэтому каждый порт маршрутизатора имеет собственный IP-адрес. Конечный узел также может входить в несколько IP-сетей. В этом случае компьютер должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

### **Классы IP-адресов**

Какая часть адреса относится к номеру сети, а какая к номеру узла, определяется значениями первых бит адреса. Значения этих бит являются также признаками того, к какому классу относится тот или иной IP-адрес.

На рисунке показана структура IP-адреса разных классов.



### **Бесклассовая адресация**

Со второй половины 90-х годов XX века классовая маршрутизация повсеместно вытеснена бесклассовой маршрутизацией, при которой количество адресов в сети определяется только и исключительно маской подсети.

### **Особые IP-адреса**

В протоколе IP существует несколько соглашений об особой интерпретации IP-адресов:

* eсли весь IP-адрес состоит только из двоичных нулей, то он обозначает адрес того узла, который сгенерировал этот пакет; этот режим используется только в некоторых сообщениях ICMP;
* eсли в поле номера сети стоят только нули, то по умолчанию считается, что узел назначения принадлежит той же самой сети, что и узел, который отправил пакет;
* eсли все двоичные разряды IP-адреса равны 1, то пакет с таким адресом назначения должен рассылаться всем узлам, находящимся в той же сети, что и источник этого пакета. Такая рассылка называется ограниченным широковещательным сообщением (limited broadcast);
* eсли в поле номера узла назначения стоят только единицы, то пакет, имеющий такой адрес, рассылается всем узлам сети с заданным номером сети. Например, в сети 192.190.21.0 с маской 255.255.255.0 пакет с адресом 192.190.21.255 доставляется всем узлам сети этой сети. Такая рассылка называется широковещательным сообщением (broadcast).

### **Динамические IP-адреса**

IP-адрес называют динамическим, если он назначается автоматически при подключении устройства к сети и используется в течение ограниченного промежутка времени, как правило, до завершения сеанса подключения.

## **Протокол связи TCP**

**Транспортный протокол (TCP)** используется для передачи информации. Информация передаётся маленькими частями – сетевыми пакетами. То есть поток информации разбивается на много маленьких пакетов.

Каждый пакет состоит из заголовка и самих данных. Заголовок содержит служебную информацию, например порт источника и назначения.

Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым (в отличие от UDP) целостность передаваемых данных и уведомление отправителя о результатах передачи.

### **Протокол TCP и его особенности**

Главной особенностью TCP (Transmission Control Protocol) является то, что этот протокол гарантирует доставку всех отправленных пакетов. При этом проверяется целостность пакетов и их порядок. Если пакет потерялся или испортился, то получатель запросит эти пакеты у отправителя снова. Если пакеты пришли не в том порядке, то они на принимающей стороне всё равно обработаются в правильном. Этот механизм контроля доставки накладывает дополнительную нагрузку в виде увеличения служебной информации, которую нужно передать вместе с полезными данными.

Протокол TCP делит поток информации на сегменты. В одном сегменте может быть несколько пакетов. Каждый сегмент проверяется на целостность, и если все хорошо, отправляется подтверждение передающей стороне. Таким образом подтверждается не каждый пакет, а каждый сегмент, но в сегменте может оказаться и всего лишь один пакет.

Поверх протокола TCP работают многие прикладные протоколы:

* сайты (http, https);
* электронная почта (imap, pop, smtp);
* передача файлов (cifs, nfs, ftp);
* удаленные подключения (rdp, ssh).

TCP пакеты передаются не просто так, а в рамках установленного соединения – которое называют TCP сессией.

Подключение можно выполнить только если вторая сторона прослушивает порт, к которому будет выполняться подключение.

### **Алгоритм работы TCP**

Алгоритм работы TCP следующий:

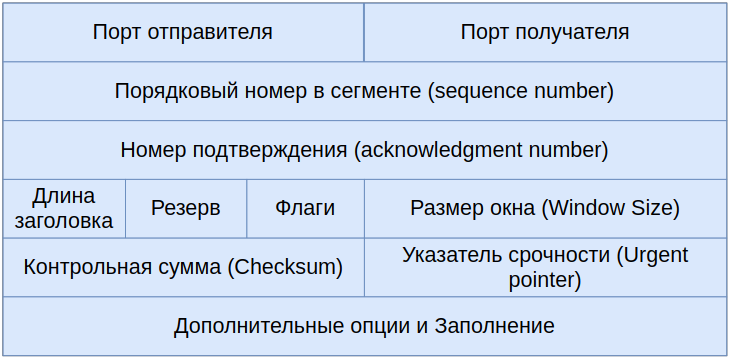
1. Используя трехкратное рукопожатие, между двумя узлами создаётся сеанс связи.
2. При отправке пакетов узлы последовательно нумеруют их и рассчитывают контрольную сумму.
3. Поскольку все пакеты имеют последовательные номера, то становится видно если какие-то из них отсутствуют. В этом случае отправляется запрос на повторную отправку пакета.
4. Если для какого-то пакета не совпала контрольная сумма, то отправляется запрос на повторную отправку пакета.

При открытии даже одной веб странички создаются несколько TCP соединений для:

* html страницы;
* каждого CSS и JavaScript файлов;
* каждого изображения.

И для каждого такого соединения вначале устанавливается сеанс, что замедляет передачу данных.

### **Заголовок TCP пакета**



## **Топология сети тестирования**

**Топология сети** — это расположение элементов (связей, узлов и т.д.) коммуникационной сети. Топология сети может использоваться для определения или описания расположения различных типов телекоммуникационных сетей, включая командные и управляющие радиосети, промышленные полевые шины и компьютерные сети.

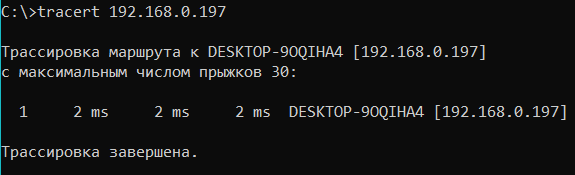


Рис 1. Трассирование устройства в локальной сети

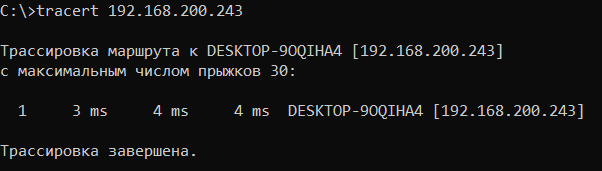
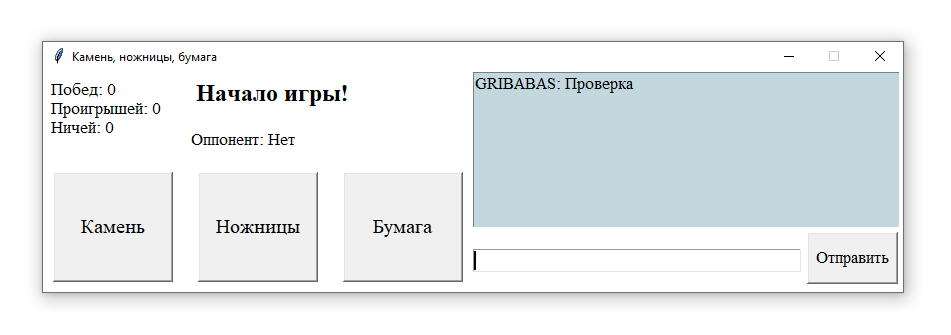


Рис 2. Трассирование устройства в моб. сети

# **Интерфейс приложения**



# **Листинг программы**

## **Серверная часть**

import json

import socket

import threading

from typing import Optional, Union

class Result:

WIN = "win"

LOSE = "lose"

DRAW = "draw"

class RPSServer:

def \_\_init\_\_(self, host, port):

self.host = host

self.port = port

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.sock.bind((self.host, self.port))

self.clients: list[socket.socket] = []

self.actions: dict[socket.socket, dict[str, Union[int, str]]] = {}

def listen(self):

self.sock.listen(2)

while True:

client, address = self.sock.accept()

threading.Thread(target=self.client\_handler, args=(client,)).start()

self.clients.append(client)

def distribute(self, data: str, author: Optional[socket.socket] = None):

for client in self.clients:

if client == author:

continue

client.send(data.encode())

def client\_handler(self, client: socket.socket):

while True:

try:

data = json.loads(client.recv(1024).decode())

if not data:

continue

command = data["command"]

nickname = data["nickname"]

message = data["message"]

if command == "chat":

self.distribute(json.dumps(data), client)

continue

if command == "action":

self.actions[client] = {

"action": int(message),

"nickname": nickname,

}

if len(self.actions) == 2:

self.send\_result()

self.actions = {}

except Exception as e:

print(e)

self.clients.remove(client)

client.close()

return False

def send\_result(self):

user = list(self.actions.keys())[0]

opponent = list(self.actions.keys())[1]

user\_choice = self.actions[user]["action"]

opponent\_choice = self.actions[opponent]["action"]

user\_result = Result.LOSE

opponent\_result = Result.WIN

if user\_choice == opponent\_choice:

user\_result = Result.DRAW

opponent\_result = Result.DRAW

if (user\_choice + 1) % 3 == opponent\_choice:

user\_result = Result.WIN

opponent\_result = Result.LOSE

user.send(

json.dumps(

{

"command": "result",

"message": user\_result,

"nickname": self.actions[opponent]["nickname"],

}

).encode()

)

opponent.send(

json.dumps(

{

"command": "result",

"message": opponent\_result,

"nickname": self.actions[user]["nickname"],

}

).encode()

)

if \_\_name\_\_ == "\_\_main\_\_":

RPSServer("192.168.0.197", 9090).listen()

## **Клиентская часть**

import json

import random

import threading

import socket

from typing import Optional

from enum import IntEnum

from tkinter import Tk, Frame, Button, Label, END

import tkinter as tk

class Action(IntEnum):

Rock = 0

Scissors = 1

Paper = 2

class GameCommand:

def \_\_init\_\_(self, game: "Main", choice: Action):

self.choice = choice

self.game = game

def process\_button(self):

self.game.client.send("action", str(self.choice.value))

def \_\_call\_\_(self):

for btn in self.game.game\_buttons:

if btn["state"] == tk.DISABLED:

return

btn["state"] = tk.DISABLED

threading.Thread(target=self.process\_button).start()

class Main(Frame):

def \_\_init\_\_(self, root, client\_: "SocketClient"):

super(Main, self).\_\_init\_\_(root)

self.client = client\_

client\_.game = self

self.root = root

self.opponent\_name = ""

self.game\_buttons = []

self.game\_start\_label: Optional[Label] = None

self.game\_data\_label: Optional[Label] = None

self.opponent\_label: Optional[Label] = None

self.entry: Optional[tk.Entry] = None

self.txt: Optional[tk.Text] = None

self.button\_font = ("Times New Roman", 15)

self.mini\_button\_font = ("Times New Roman", 13)

self.win = self.draw = self.lose = 0

self.start\_iu()

def game\_data\_text(self):

return f"Побед: {self.win}\nПроигрышей:" f" {self.lose}\nНичей: {self.draw}"

def send\_button(self, \*args):

input\_text = self.entry.get()

if not input\_text:

return

self.txt.configure(state="normal")

self.txt.insert(END, f"GRIBABAS : {input\_text}\n")

self.txt.see("end")

self.txt.configure(state="disabled")

self.entry.delete(0, END)

self.client.send("chat", input\_text)

def start\_iu(self):

self.game\_buttons = [

Button(self.root,text="Камень",font=self.button\_font,command=GameCommand(self, Action.Rock),),

Button(self.root,text="Ножницы",font=self.button\_font,command=GameCommand(self, Action.Scissors),),

Button(self.root,text="Бумага",font=self.button\_font,command=GameCommand(self, Action.Paper),),

]

self.game\_buttons[0].place(x=10, y=100, width=120, height=110)

self.game\_buttons[1].place(x=155, y=100, width=120, height=110)

self.game\_buttons[2].place(x=300, y=100, width=120, height=110)

self.master.bind("<Return>", self.send\_button)

self.game\_start\_label = Label(self.root,text="Начало игры!",bg="#FFF",font=("Times New Roman", 18, "bold"),)

self.game\_data\_label = Label(self.root,justify="left",font=self.mini\_button\_font,text=self.game\_data\_text(),bg="#FFF",)

self.opponent\_label = Label(self.root,justify="right",font=self.mini\_button\_font,text=f"Оппонент: Нет",bg="#FFF",)

self.game\_start\_label.place(x=150, y=5)

self.game\_data\_label.place(x=5, y=5)

self.opponent\_label.place(x=145, y=55)

self.txt = tk.Text(self.root, font=self.mini\_button\_font, width=47, height=8, bg="#c3d7df")

self.txt.configure(state="disabled")

self.txt.place(x=430, y=0)

self.entry = tk.Entry(self.root, font=self.mini\_button\_font, width=36, bg="white")

self.entry.place(x=430, y=177)

send = Button(self.root,text="Отправить",font=self.mini\_button\_font,command=self.send\_button,width=9,height=2,)

send.place(x=764, y=160)

class SocketClient:

def \_\_init\_\_(self, name: str):

self.client = None

self.name = name

self.game = None

def result\_handler(self, message: str):

if message == "draw":

self.game.draw += 1

self.game.game\_start\_label.configure(text="Ничья")

if message == "win":

self.game.win += 1

self.game.game\_start\_label.configure(text="Победа")

if message == "lose":

self.game.lose += 1

self.game.game\_start\_label.configure(text="Проигрыш")

self.game.game\_data\_label.configure(text=self.game.game\_data\_text())

for btn in self.game.game\_buttons:

btn["state"] = tk.NORMAL

def socket\_start(self, host: str, port: int):

self.client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.client.connect((host, port))

while True:

data = self.client.recv(1024)

if not data:

continue

data = json.loads(data.decode())

command = data["command"]

nickname = data["nickname"]

message = data["message"]

# todo: handler

self.game.opponent\_label.configure(text=f"Оппонент: {nickname}")

if command == "result":

self.result\_handler(message)

if command == "chat":

self.game.txt.configure(state="normal")

self.game.txt.insert(END, f"{nickname} : {message}\n")

self.game.txt.see("end")

self.game.txt.configure(state="disabled")

def send(self, command: str, message: str):

data = json.dumps(

{"command": command, "nickname": self.name, "message": message}

)

self.client.sendall(data.encode())

if \_\_name\_\_ == "\_\_main\_\_":

main\_root = Tk()

main\_root.geometry("860x220+200+200")

main\_root.title("Камень, ножницы, бумага")

main\_root.resizable(False, False)

main\_root["bg"] = "#FFF"

nick = "GRIBABAS"

print(nick)

client = SocketClient(name=nick)

app = Main(main\_root, client)

app.pack()

game\_thread = threading.Thread(target=main\_root.mainloop)

socket\_thread = threading.Thread(target=client.socket\_start, args=("192.168.0.197", 9090))

socket\_thread.start()

game\_thread.run()

# **Заключение**

В результате курсовой работы была разработана игра "Камень-Ножницы-Бумага" с клиент-серверным взаимодействием. Одновременно с этим была проведена настройка устройств ЭВМ для взаимодействия в локальной сети.

Были изучены основы маршрутизации, стека TCP/IP. Была произведена настройка протокола транспортного уровня, а также сокетов серверной и клиентской части игры

# **Список литературы**

1. Статья «Что такое IP-адресация?» [Электронный ресурс] URL: <https://2ip.ru/article/ip/>
2. Статья «Что такое транспортные протоколы» [Электронный ресурс] URL: <https://sysadminium.ru/tcp/>
3. Википедия «Transmission Control Protocol» [Электронный ресурс] URL:   
   <https://ru.wikipedia.org/wiki/Transmission_Control_Protocol>
4. Википедия « Network topology» [Электронный ресурс] URL:   
   <https://en.wikipedia.org/wiki/Network_topology>