ФГБОУ ВО

ННГАСУ

Кафедра информационных систем и технологий

Курсовая работа

Инфокоммуникационные системы и сети

**Разработка приложения с реализацией передачи данных по транспортному уровню**

Выполнила студентка 3 курса группы ИС-29

Хазова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Проверил преподаватель

Морозов Н.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Нижний Новгород

2022 год

Оглавление

[Цель работы 3](#_Toc124956040)

[Задачи 3](#_Toc124956041)

[Теоретическая часть 4](#_Toc124956042)

[Описание пользовательского интерфейса 8](#_Toc124956043)

[Заключение 11](#_Toc124956044)

[Список использованной литературы 12](#_Toc124956045)

[Приложение: листинг кода программы 13](#_Toc124956046)

# Цельработы

Разработка приложения, которое реализует передачу пакетов данных с использованием протокола транспортного уровня.

# ****Задачи****

1. изучение маршрутизации стека TCP/IP;
2. настройка протокола транспортного уровня;
3. настройка сокетов серверной и клиентской части игры.

# Теоретическая часть

*Протокол связи, используемый в программе*: Transmission Control Protocol (TCP, протокол управления передачей)  — один из основных [протоколов передачи данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Механизм TCP предоставляет [поток данных](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с предварительной установкой соединения. В рамках соединения осуществляется обязательное подтверждение правильности приема для всех переданных сообщений, и при необходимости выполняется повторная передача. Соединение в TCP позволяет вести передачу данных одновременно в обе стороны, то есть полнодуплексную передачу.

Согласно спецификации протокола TCP/IP, каждому узлу, подсоединенному к IP-сети, присваивается уникальный номер. Узел может представлять собой компьютер, маршрутизатор, межсетевой экран и др. Если один узел имеет несколько физических подключений к сети, то каждому подключению должен быть присвоен свой уникальный номер.

Этот номер, или по-другому IP-адрес, имеет длину в четыре октета, и состоит из двух частей. Первая часть определяет сеть, к которой принадлежит узел, а вторая - уникальный адрес самого узла внутри сети.

Таблица 1 - Представление IP-адреса в двоичном виде

|  |  |  |  |
| --- | --- | --- | --- |
| Номер сети | | | Номер узла |
| 11011100 | 11010111 | 00001110 | 00010110 |

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например, 192.168.2.35. Каждое число в этом наборе принадлежит интервалу от 0 до 255. Таким образом, полный диапазон IP-адресации – это адреса от 0.0.0.0 до 255.255.255.255.

Соединение в протоколе TCP идентифицируется парой полных адресов обоих взаимодействующих процессов (оконечных точек). Адрес каждой из оконечных точек включает IP-адрес (номер сети и номер компьютера) и номер порта.

Установление соединения выполняется в следующей последовательности:

* При установлении соединения одна из сторон является инициатором. Она посылает запрос к протоколу TCP на открытие порта для передачи (active open).
* После открытия порта протокол TCP на стороне процесса-инициатора посылает запрос процессу, с которым требуется установить соединение.
* Протокол TCP на приемной стороне открывает порт для приема данных (passive open) и возвращает квитанцию, подтверждающую прием запроса.
* Для того чтобы передача могла вестись в обе стороны, протокол на приемной стороне также открывает порт для передачи (active port) и также передает запрос к противоположной стороне.
* Сторона-инициатор открывает порт для приема и возвращает квитанцию. Соединение считается установленным. Далее происходит обмен данными в рамках данного соединения.

Для корректной работы сокетов нам необходимо знать IP-адрес компьютера. Для этого, в операционной системе Windows, заходим в командную строку, где вводим команду *ipconfig.* В разделе «Беспроводная сеть» находим строку IPv4-адрес, где и будет находиться IP вашего компьютера в данной(локальной) сети.

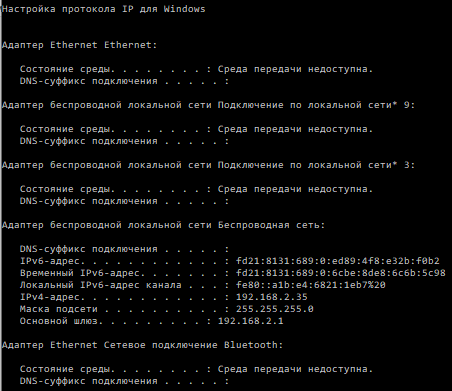


Рисунок 1 - Результат работы команды ipconfig

Чтобы узнать свой глобальный (внешний) IP-адрес (выдается провайдером и помогает устройству подключаться к сети Интернет) достаточно перейти на сайт  [2ip.ru](https://2ip.ru/).

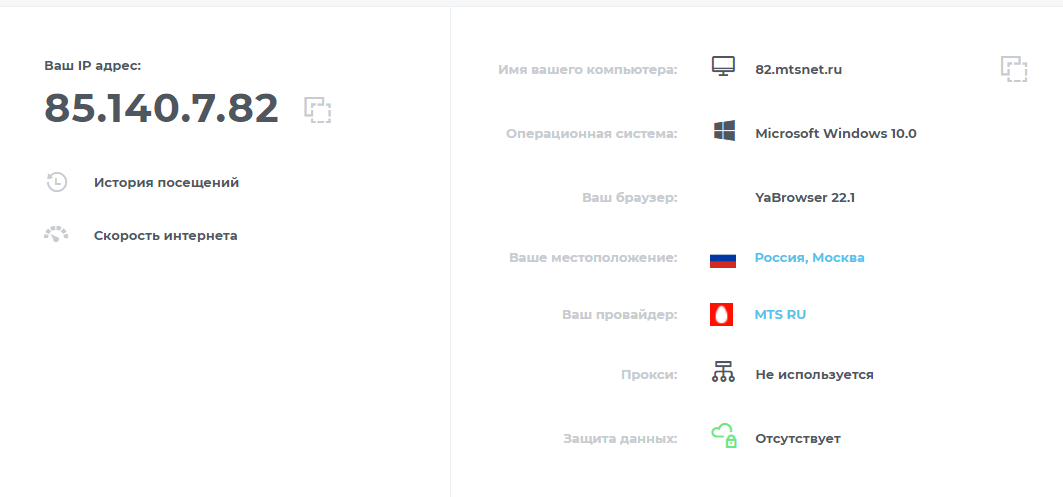


Рисунок 2 – Сервис 2ip.ru

*Топология сети тестирования:*

Разрабатываемая программа будет соединять пару компьютеров посредством связи клиент-сервер. С помощью команды tracert рассмотрим маршрут, через который проходят пакеты данных:

1. при подключении к другому компьютеру, который использует кабельное подключение к сети Интернет:

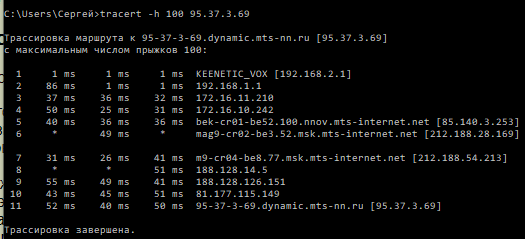


Рисунок 3 – Результат команды tracert для домашней сети

Следовательно, при передаче данных от одного компьютера к другому, пакеты проходят через 10 узлов.

1. при подключении к устройству, используемому мобильную точку доступа:

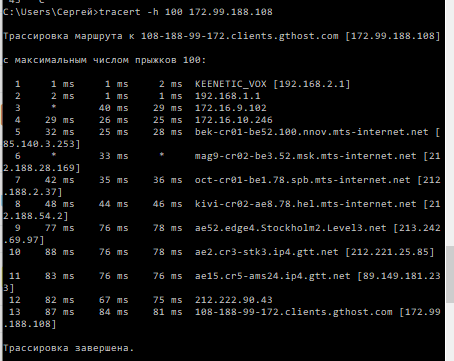


Рисунок 4 – Результаты команды tracert для

мобильной точки доступа

При передаче данных на мобильное устройство пакеты проходят через 12 узлов.

# Описание пользовательского интерфейса

1. Запуск программы.

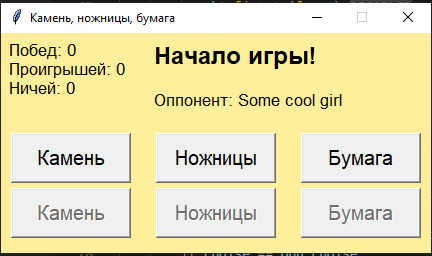


Рисунок 5 – вид интерфейса при запуске игры

1. Для того чтобы начать играть нажимаем на любую из верхних кнопок(«Камень», «Ножницы» или «Бумага»). Так реализуется выбор первого игрока.

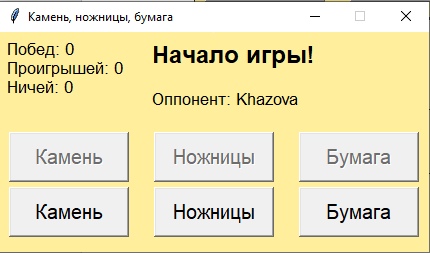


Рисунок 6 – вид интерфейса после нажатия верхних кнопок

1. После этого происходит выбор оппонента, с помощью нажатия на нижние кнопки.

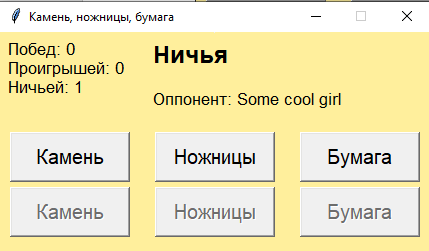


Рисунок 7 – вид интерфейса после нажатия нижних кнопок

1. Далее программа обрабатывает полученные результаты по условиям игры «Камень, ножницы, бумага» и выводит ответ на форму.

До закрытия сеанса игры все победы, проигрыши и ничьи суммируются.

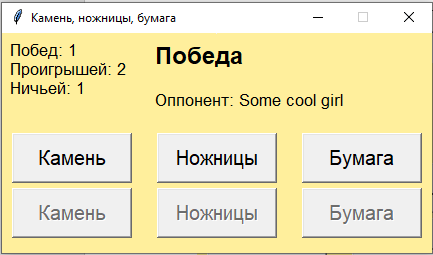


Рисунок 8 – вид интерфейса при «победе»

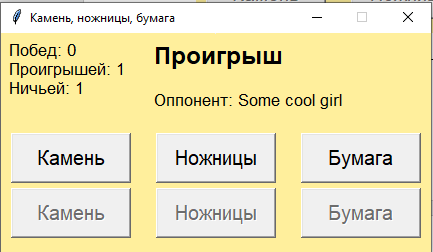


Рисунок 9 – вид интерфейса при «проигрыше»

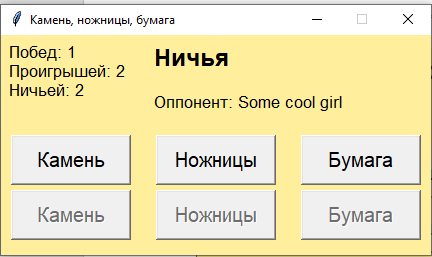


Рисунок 10 – вид интерфейса при «ничьей»

# Заключение

В ходе выполнения курсовой работы была рассмотрена реализация передачи данных через протокол TCP. Разработка приложения потребовала изучения таких аспектов, как: IP-адресация, маршрутизация стека TCP/IP, работа с сокетами клиентской и серверной части.

В результате было разработано клиент-серверное приложение на языке программирования Python 3, интерфейс которого был реализован с помощью библиотеки tkinter (кроссплатформенная библиотека для разработки графического интерфейса на языке Python). Программа реализует клиент-серверное подключение через сокеты. Реализация сокетов обеспечивает упаковку данных в единые пакеты для их передачи на транспортном уровне в нужной последовательности и без ошибок. Передача через сокеты надежна и удобна в использовании, так как сокет гарантирует исправление ошибок, обработку доставки и сохраняет последовательность отправляемых данных.

Таким образом, использование сокетов автоматизирует передачу данных по сети. Протокол TCP полностью берет на себя разбиение передаваемых данных на пакеты соответствующего размера, отправку их в сеть и обработку на другой стороне, а сокет гарантирует исправление ошибок, обрабатывает доставку и сохраняет последовательность данных. Приложение знает только, что отправляет на транспортный уровень определенное число байтов и другая подключенная сторона получает эти байты.

# Список использованной литературы

1. [Socket](https://docs.python.org/3/library/socket.html#module-socket) — Low-level networking interface / docs.python.org: [сайт]. URL: <https://docs.python.org/3/library/socket.html> (дата обращения: 17.12.2022).
2. [Tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) — Python interface to Tcl/Tk / docs.python.org: [сайт]. URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 16.12.2022).
3. Кручинин Сергей Владимирович Межуровневые протоколы сетевых стеков [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/mezhurovnevye-protokoly-setevyh-stekov/viewer (дата обращения: 14.12.2022).
4. Морковкин Егор Андреевич, Новичихина Алёна Александровна, Замулин Иван Сергеевич IP-адресация и информационная безопасность [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/ip-adresatsiya-i-informatsionnaya-bezopasnost/viewer (дата обращения: 14.12.2022).
5. Рубашенков А.М., Бобров А.В. Протокол TCP [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/protokol-tcp/viewer (дата обращения: 14.12.2022).
6. Соколов А.С. Моделирование сегмента вычислительной сети и выявление проблемных участков в процессе мониторинга. [Электронный ресурс]. URL: https://habr.com/ru/company/skillfactory/blog/690186/ (дата обращения: 15.12.2022).
7. Таненбаум Э., Уэзеролл Д. Т18 Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.: ил [Электронный ресурс]. URL: https://habr.com/ru/company/skillfactory/blog/690186/ (дата обращения: 14.12.2022).
8. Федорук В.Г. Протоколы сетевого взаимодействия TCP/IP [Электронный ресурс]. URL: https://www.opennet.ru/docs/RUS/tcpip/ (дата обращения: 16.12.2022).
9. Всё об IP-адресах и о том, как с ними работать / Habr.com: [сайт]. URL: https://cyberleninka.ru/article/n/ip-adresatsiya-i-informatsionnaya-bezopasnost/viewer (дата обращения: 15.12.2022).
10. Руководство по программированию сокетов на Python / Habr.com: [сайт]. URL: https://habr.com/ru/company/skillfactory/blog/690186/ (дата обращения: 17.12.2022).

# Приложение: листинг кода программы

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

import tkinter as tk

import threading

import socket

class Main(Frame):

def \_\_init\_\_(self, root):

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

self.opponent\_name = 'Some cool girl'

self.set\_my\_name('Khazova')

self.startUI()

self.opponent\_choise = IntVar()

def startUI(self):

self.game\_btns = [Button(root, text="Камень", font=("TkTextFont", 15),

command=lambda x=1: self.btn\_click(x)),

Button(root, text="Ножницы", font=("TkTextFont", 15),

command=lambda x=2: self.btn\_click(x)),

Button(root, text="Бумага", font=("TkTextFont", 15),

command=lambda x=3: self.btn\_click(x))]

self.game\_btns[0].place(x=10, y=155, width=120, height=50)

self.game\_btns[1].place(x=155, y=155, width=120, height=50)

self.game\_btns[2].place(x=300, y=155, width=120, height=50)

self.game\_btns1 = [Button(root, text="Камень", font=("TkTextFont", 15),

command=lambda x=1: self.btn\_click1(x)),

Button(root, text="Ножницы", font=("TkTextFont", 15),

command=lambda x=2: self.btn\_click1(x)),

Button(root, text="Бумага", font=("TkTextFont", 15),

command=lambda x=3: self.btn\_click1(x))]

self.game\_btns1[0].place(x=10, y=100, width=120, height=50)

self.game\_btns1[1].place(x=155, y=100, width=120, height=50)

self.game\_btns1[2].place(x=300, y=100, width=120, height=50)

self.lbl = Label(root, text="Начало игры!", bg="#ffef9c",

font=("TkTextFont", 18, "bold"))

self.lbl.place(x=150, y=5)

self.win = self.drow = self.lose = 0

self.lbl2 = Label(root, justify="left", font=("TkTextFont", 13),

text=f"Побед: {self.win}\nПроигрышей:"

f" {self.lose}\nНичей: {self.drow}",

bg="#ffef9c")

self.lbl3 = Label(root, justify="right", font=("TkTextFont", 13),

text=f" Оппонент: {self.opponent\_name}",

bg="#ffef9c")

self.lbl2.place(x=5, y=5)

self.lbl3.place(x=145, y=55)

for btn in self.game\_btns:

btn['state'] = tk.DISABLED

def btn\_click(self, choise):

self.choise = choise

self.lbl3.configure(text=f" Оппонент: {self.opponent\_name}")

self.calc\_result(choise, self.opponent\_choise)

for btn in self.game\_btns:

btn['state'] = tk.DISABLED

for btn in self.game\_btns1:

btn['state'] = tk.NORMAL

def btn\_click1(self, choise1):

self.opponent\_choise = choise1

for btn in self.game\_btns1:

btn['state'] = tk.DISABLED

for btn in self.game\_btns:

btn['state'] = tk.NORMAL

self.lbl3.configure(text=f" Оппонент: {self.my\_name}")

def calc\_result(self, choise, opp\_choise):

if choise == opp\_choise:

self.drow += 1

self.lbl.configure(text="Ничья")

elif choise == 1 and opp\_choise == 2 \

or choise == 2 and opp\_choise == 3 \

or choise == 3 and opp\_choise == 1:

self.win += 1

self.lbl.configure(text="Победа")

else:

self.lose += 1

self.lbl.configure(text="Проигрыш")

print(f'My choise: {choise}')

print(f'Opp choise: {opp\_choise}')

self.lbl2.configure(text=f"Побед: {self.win}\nПроигрышей:"

f" {self.lose}\nНичьей: {self.drow}")

self.set\_opponent\_choise = IntVar()

def set\_my\_name(self, name):

self.my\_name = name

def get\_my\_name(self):

return self.my\_name

def set\_opponent\_name(self, name):

self.opponent\_name = name

def is\_opponent\_chosen(self):

return self.opponent\_choise != 'None'

def set\_opponent\_choise(self, opp\_choise):

root.after(20, self.opponent\_choise.set, opp\_choise)

def get\_opponent\_choise(self):

return self.opponent\_choise.get()

def socket\_start():

global app

def client(sock):

while True:

message = app.opponent\_choise

print(message)

sock.send(message)

def server(conn):

while True:

data = conn.recv(1024)

if not data:

break

Main.set\_opponent\_choise(app, data)

print("Полученно!")

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

sock1 = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = "192.168.38.19"

port = 12345

sock.bind((host, port))

sock.listen(1)

sock1.connect((host, port))

conn, addr = sock.accept()

print('connected: ', addr)

socket1\_thread = threading.Thread(target=server, args=(conn, ))

socket2\_thread = threading.Thread(target=client, args=(sock1, ))

socket1\_thread.start()

socket2\_thread.start()

if \_\_name\_\_ == '\_\_main\_\_':

root = Tk()

root.geometry("430x220+200+200")

root.title("Камень, ножницы, бумага")

root.resizable(False, False)

root["bg"] = "#ffef9c"

app = Main(root)

app.pack()

socket\_thread = threading.Thread(target=socket\_start, args=())

socket\_thread.start()

root.mainloop()