МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Нижегородский государственный архитектурно-строительный университет»

*Кафедра информационных систем и технологий*

**Курсовая работа по дисциплине**

**«Инфокоммуникационные системы и сети»**

Выполнил студент гр. ИС-29: Куликов А.А.

Проверил старший преподаватель: Морозов Н. С.

Нижний Новгород

2022 г.

**Содержание**

1. Задание 3
2. Теоретическая часть 4
3. Листинг программы 11
4. Пример работы приложения 19
5. Заключение 20
6. Список литературы 21

**Задание**

Изучить маршрутизацию, стек TCP/IP, настройку протокола транспортного уровня, настройку сокетов серверной и клиентской части игры.

**Теоретическая часть**

**Определение IP-адреса**

IP-адрес – это уникальный адрес, идентифицирующий устройство в интернете или локальной сети. IP означает «Интернет-протокол» – набор правил, регулирующих формат данных, отправляемых через интернет или локальную сеть.

По сути, IP-адрес – это идентификатор, позволяющий передавать информацию между устройствами в сети: он содержит информацию о местоположении устройства и обеспечивает его доступность для связи. IP-адреса позволяют различать компьютеры, маршрутизаторы и веб-сайты в интернете и являются важным компонентом работы интернета.

**Что такое IP-адрес?**

IP-адрес – это строка чисел, разделенных точками. IP-адреса представляют собой набор из четырех чисел, например, 192.158.1.38. 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.

IP-адреса не случайны. Они рассчитываются математически и распределяются Администрацией адресного пространства Интернета (Internet Assigned Numbers Authority, IANA), подразделением Корпорации по присвоению имен и номеров в Интернете (Internet Corporation for Assigned Names and Numbers, ICANN). ICANN – это некоммерческая организация, основанная в США в 1998 году с целью поддержки безопасности интернета и обеспечения его доступности для всех пользователей. Каждый раз, когда кто-либо регистрирует домен в интернете, он пользуется услугами регистратора доменных имен, который платит ICANN небольшой сбор за регистрацию домена.

**Как работают IP-адреса**

Понимание того, как работают IP-адреса, поможет разобраться, почему определенное устройство не подключается так, как ожидалось, и устранить неполадки в работе сети.

Интернет-протокол работает так же, как и любой другой язык: передает информацию с использованием установленных правил. Устройства обнаруживают другие подключенные устройства и обмениваются с ними информацией, используя этот протокол. Проще говоря, все компьютеры, где бы они не находились, могут общаться друг с другом.

Использование IP-адресов обычно происходит незаметно. Процесс работает следующим образом:

* Устройство подключается к интернету не напрямую: сначала оно подключается к сети, подключенной к интернету, а сеть, в свою очередь, предоставляет устройству доступ к интернету.
* Если вы находитесь дома, скорее всего, этой сетью является сеть вашего интернет-провайдера. В офисе это будет сеть вашей компании.
* IP-адрес назначается устройству вашим интернет-провайдером.
* Ваша интернет-активность проходит через интернет-провайдера, а он перенаправляет вам ответы на запросы, используя ваш IP-адрес. Поскольку провайдер предоставляет доступ в Интернет, его роль заключается в назначении IP-адрес вашему устройству.
* Однако ваш IP-адрес может измениться, например, при включение или выключение модема или маршрутизатора. Можно также связаться с интернет-провайдером, чтобы он изменил IP-адрес.
* Если вы находитесь вне дома, например, путешествуете, и берете с собой устройство, домашний IP-адрес не закрепляется за устройством. Это связано с тем, что устройство будет использовать другую сеть (Wi-Fi в отеле, аэропорту, кафе) для доступа в интернет и другой временный IP-адрес, назначенный интернет-провайдером в отеле, аэропорту или кафе.

Как следует из этого процесса, существуют различные типы IP-адресов, которые будут описаны ниже.

**Типы IP-адресов**

Существуют разные категории IP-адресов, и в каждой категории имеются разные типы.

**Клиентские IP-адреса**

У каждого человека или компании с тарифным планом на получение интернет-услуг есть два типа IP-адресов: частный и общедоступный. Термины частный и общедоступный относятся к сетевому расположению: частный IP-адрес используется внутри сети, а общедоступный – за пределами сети.

**Частные IP-адреса**

Каждое устройство, которое подключается к вашей интернет-сети, имеет частный IP-адрес. Это могут быть компьютеры, смартфоны, планшеты, а также любые устройства с поддержкой Bluetooth, такие как динамики, принтеры, смарт-телевизоры. С развитием интернета вещей растет и количество частных IP-адресов в домашней сети. Маршрутизатору необходимо идентифицировать каждое из этих устройств, а многие устройства также должны идентифицировать друг друга. Поэтому маршрутизатор генерирует частные IP-адреса, которые являются уникальными идентификаторами каждого устройства и позволяют различать их в сети.

**Общедоступные IP-адреса**

Общедоступный IP-адрес – это основной адрес, связанный со всей сетью. Каждое подключенное устройство имеет собственный IP-адрес, но они также включены в состав основного IP-адреса сети. Как было описано выше, общедоступный IP-адрес предоставляется маршрутизатору интернет-провайдером. Обычно у интернет-провайдеров есть большой пул IP-адресов, которые они присваивают клиентам. Общедоступный IP-адрес – это адрес, который устройства за пределами интернет-сети будут использовать для распознавания этой сети.

**Общедоступные IP-адреса**

Общедоступные IP-адреса бывают двух видов: динамические и статические.

**Динамические IP-адреса**

Динамические IP-адреса меняются автоматически и регулярно. Интернет-провайдеры покупают большой пул IP-адресов и автоматически присваивают их своим клиентам. Периодически они меняют присвоенные IP-адреса и помещают старые IP-адреса обратно в пул для использования другими клиентами. Обоснованием этого подхода служит экономия средств провайдера. Автоматизация регулярного изменения IP-адресов позволяет им не выполнять никаких действий для восстановления IP-адреса клиента, например, если он переезжает. Также имеются преимущества с точки зрения безопасности, поскольку изменение IP-адреса затрудняет взлом сетевого интерфейса злоумышленниками.

**Статические IP-адреса**

В отличие от динамических IP-адресов, статические IP- адреса остаются неизменными. После того, как сеть назначает IP-адрес, он остается неизменным. Большинству частных лиц и организаций не нужны статические IP-адреса, но для организаций, планирующих размещать собственные серверы, наличие статического IP-адреса крайне важно. Это связано с тем, что статический IP-адрес гарантирует, что привязанные к нему веб-сайты и адреса электронной почты будут иметь постоянные IP-адреса. Это очень важно, если требуется, чтобы другие устройства могли находить их в интернете.

В результате возникла классификация по типам IP-адресов веб-сайтов.

**Два типа IP-адресов веб-сайтов**

Для владельцев веб-сайтов, использующих пакет веб-хостинга (что характерно для большинства веб-сайтов), а не собственный сервер, существует два типа IP-адресов веб-сайтов: общие и выделенные.

**Общие IP-адреса**

Веб-сайты, использующие общие хостинговые планы от провайдеров веб-хостинга, обычно являются одним из многих веб-сайтов, размещенных на одном сервере. Это, как правило, веб-сайты физических лиц или компаний малого и среднего бизнеса, с ограниченным объемом трафика, количеством страниц и т. д. Такие веб-сайты имеют общие IP-адреса.

**Выделенные IP-адреса**

В некоторых тарифных планах веб-хостинга есть возможность приобрести выделенный IP-адрес (или адреса). Это может упростить получение SSL-сертификата и позволяет использовать собственный FTP-сервер (сервер протокола передачи файлов). Кроме того, это упрощает организацию общего доступа и передачу файлов в рамках организации и позволяет использовать анонимный FTP-доступ. Выделенный IP-адрес также позволяет получить доступ к веб-сайту, используя только IP-адрес, а не доменное имя. Это полезно, если требуется создать и протестировать его перед регистрацией домена.

**Протокол TCP/IP**

Протокол TCP/IP – это целая сетевая модель, описывающая способ передачи данных в цифровом виде. На правилах, включенных в нее, базируется работа интернета и локальных сетей независимо от их назначения и структуры.

Произошло наименование протокола от сокращения двух английских понятий – Transmission Control Protocol и Internet Protocol. Набор правил, входящий в него, позволяет обрабатывать как сквозную передачу данных, так и другие детали этого механизма. Сюда входит формирование пакетов, способ их отправки, получения, маршрутизации, распаковки для передачи программному обеспечению.

Стек протоколов TCP/IP был создан в 1972 году на базе NCP (Network Control Protocol), в январе 1983 года он стал официальным стандартом для всего интернета. Техническая спецификация уровней взаимодействия описана в документе RFC 1122.

В составе стека есть и другие известные протоколы передачи данных – UDP, FTP, ICMP, IGMP, SMTP. Они представляют собой частные случаи применения технологии: например, у SMTP единственное предназначение заключается в отправке электронных писем.

**Уровни модели TCP/IP**

Протокол TCP/IP основан на OSI и так же, как предшественник, имеет несколько уровней, которые и составляют его архитектуру. Всего выделяют 4 уровня – канальный (интерфейсный), межсетевой, транспортный и прикладной.

**Канальный (сетевой интерфейс)**

Аппаратный уровень обеспечивает взаимодействие сетевого оборудования Ethernet и Wi-Fi. Он соответствует физическому из предыдущего стандарта OSI. Здесь задача состоит в кодировании информации, ее делению на пакеты и отправке по нужному каналу. Также измеряются параметры сигнала вроде задержки ответа и расстояния между хостами.

**Межсетевой (Internet Layer)**

Интернет состоит из множества локальных сетей, объединенных между собой как раз за счет протокола связи TCP/IP. Межсетевой уровень регламентирует взаимодействие между отдельными подсетями. Маршрутизация осуществляется путем обращения к определенному IP-адресу с использованием маски.

Если хосты находятся в одной подсети, маркируемой одной маской, данные передаются напрямую. В противном случае информация «путешествует» по целой цепочке промежуточных звеньев, пока не достигнет нужной точки. Назначение IP-адреса проводится по стандарту IPv4 или IPv6 (они не совместимы между собой).

**Транспортный уровень (Transport Layer)**

Следующий уровень отвечает за контроль доставки, чтобы не возникало дублей пакетов данных. В случае обнаружения потерь или ошибок информация запрашивается повторно. Такой подход дает возможность полностью автоматизировать процессы независимо от скорости и качества связи между отдельными участками интернета или внутри конкретной подсети.

Протокол TCP отличается большей достоверностью передачи данных по сравнению с тем же UDP, который подходит только для передачи потокового видео и игровой графики. Там некритичны потери части пакетов, чего нельзя сказать о копировании программных файлов и документов. На этом уровне данные не интерпретируются.

**Прикладной уровень (Application Layer)**

Здесь объединены 3 уровня модели OSI – сеансовый, представления и прикладной. На него ложатся задачи по поддержанию сеанса связи, преобразованию данных, взаимодействию с пользователем и сетью. На этом уровне применяются стандарты интерфейса API, позволяющего передавать команды на выполнение определенных задач.

Возможно и использование «производных» протоколов. Например, для открытия сайтов используется HTTPS, при отправке электронной почты – SMTP, для назначения IP-адресов – DHCP. Такой подход упрощает программирование, снижает нагрузку на сеть, увеличивает скорость обработки команд и передачи данных.

**Порты и сокеты – что это и зачем они нужны**

Процессы, работающие на прикладном уровне, «общаются» с транспортным, но они видны ему как «черные ящики» с зашифрованной информацией. Зато он понимает, на какой IP-адрес адресованы данные и через какой порт надо их принимать. Этого достаточно для точного распределения пакетов по сети независимо от месторасположения хостов. Порты с 0 до 1023 зарезервированы операционными системами, остальные, в диапазоне от 1024 до 49151, условно свободны и могут использоваться сторонними приложениями.

Комбинация IP-адреса и порта называется сокетом и используется при идентификации компьютера. Если первый критерий уникален для каждого хоста, второй обычно фиксирован для определенного типа приложений. Так, получение электронной почты проходит через 110 порт, передача данных по протоколу FTP – по 21, открытие сайтов – по 80.

**Преобразование IP-адресов в символьные адреса**

Технология активно используется для назначения буквенно-цифровых названий веб-ресурсов. При вводе домена в адресной строке браузера сначала происходит обращение к специальному серверу DNS. Он всегда прослушивает порт 53 у всех компьютеров, которые подключены к интернету, и по запросу преобразует введенное название в стандартный IP-адрес.

После определения точного местонахождения файлов сайта включается обычная схема работы – от прикладного уровня с кодированием данных до обращения к физическому оборудованию на уровне сетевых интерфейсов. Процесс называется инкапсуляцией информации. На принимающей стороне происходит обратная процедура – декапсуляция.

**Листинг программы**

**server.py:**

import json

import socket

import threading

from typing import Optional, Union

class Result:

Win = "Победа"

Lose = "Поражение"

Draw = "Ничья"

class Servak:

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)

print("Сервак запущен")

while True:

client, address = self.sock.accept()

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

self.clients.append(client)

print(f"Зашёл новый игрок")

def spread(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"]

print(f"Nickname: {nickname} Command: {command} Message: {message}")

if command == "chat":

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

continue

if command == "action":

self.actions[client] = {

"action": int(message),

"nickname": nickname,

}

if len(self.actions) == 2:

self.out\_result()

self.actions = {}

except Exception as e:

print(f"Ошибка клиента: {e}")

self.clients.remove(client)

client.close()

print("Клиент закрыт")

return False

def out\_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 == 0 and opponent\_choice == 1 or opponent\_choice == 3:

user\_result = Result.Win

opponent\_result = Result.Lose

if user\_choice == 1 and opponent\_choice == 2 or opponent\_choice == 3:

user\_result = Result.Win

opponent\_result = Result.Lose

if user\_choice == 2 and opponent\_choice == 0 or opponent\_choice == 4:

user\_result = Result.Win

opponent\_result = Result.Lose

if user\_choice == 3 and opponent\_choice == 2 or opponent\_choice == 4:

user\_result = Result.Win

opponent\_result = Result.Lose

if user\_choice == 4 and opponent\_choice == 0 or opponent\_choice == 1:

user\_result = Result.Win

opponent\_result = Result.Lose

if opponent\_choice == 0 and user\_choice == 1 or user\_choice == 3:

opponent\_result = Result.Win

user\_result = Result.Lose

if opponent\_choice == 1 and user\_result == 2 or user\_result == 3:

opponent\_result = Result.Win

user\_result = Result.Lose

if opponent\_choice == 2 and user\_result == 0 or user\_result == 4:

opponent\_result = Result.Win

user\_result = Result.Lose

if opponent\_choice == 3 and user\_result == 2 or user\_result == 4:

opponent\_result = Result.Win

user\_result = Result.Lose

if opponent\_choice == 4 and user\_result == 0 or user\_result == 1:

opponent\_result = Result.Win

user\_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\_\_":

Servak("0.0.0.0", 8081).listen()

**client.py:**

import json

import random

from enum import IntEnum

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

import tkinter as tk

import threading

import socket

from typing import Optional

class action(IntEnum):

Stone = 0

Scissors = 1

Paper = 2

Lizard = 3

Spock = 4

class game\_commands:

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, client\_: "SocketClient"):

main\_root = Tk()

main\_root.geometry("720x360")

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

main\_root.resizable(False, False)

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

super(Main, self).\_\_init\_\_(main\_root)

self.client = client\_

client\_.game = self

self.root = main\_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"Я -> {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=game\_commands(self, action.Stone),

),

Button(

self.root,

text="Ножницы",

font=self.\_button\_font,

command=game\_commands(self, action.Scissors),

),

Button(

self.root,

text="Бумага",

font=self.\_button\_font,

command=game\_commands(self, action.Paper),

),

Button(

self.root,

text="Ящерица",

font=self.\_button\_font,

command=game\_commands(self, action.Lizard),

),

Button(

self.root,

text="Спок",

font=self.\_button\_font,

command=game\_commands(self, action.Spock),

),

]

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

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

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

self.game\_buttons[3].place(x=445, y=100, width=120, height=50)

self.game\_buttons[4].place(x=590, y=100, width=120, height=50)

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

self.game\_start\_label = Label(

self.root,

justify="center",

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=300, y=5)

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

self.opponent\_label.place(x=325, y=55)

self.txt = tk.Text(

self.root, font=self.\_mini\_button\_font, width=77, height=8, bg="#c1b7c9"

)

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

self.txt.place(x=10, y=160)

scrollbar = tk.Scrollbar(self.txt)

scrollbar.place(relheight=1, relx=1)

self.entry = tk.Entry(

self.root, font=self.\_mini\_button\_font, width=66, bg="#556066"

)

self.entry.place(x=10, y=335)

send = Button(

self.root,

text="Отправить",

font=self.\_mini\_button\_font,

command=self.send\_button,

width=9,

height=1,

)

send.place(x=615, y=325)

def normalize\_buttons(self):

for button in self.game\_buttons:

button["state"] = tk.NORMAL

class SocketClient:

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

self.client = None

self.name = name

self.game: Main | None = None

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

if message == "Ничья":

self.game.draw += 1

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

if message == "Победа":

self.game.win += 1

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

if message == "Поражение":

self.game.lose += 1

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

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

self.game.normalize\_buttons()

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"]

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\_\_":

nick = f"Игрок №{random.randint(0, 9999)}"

print(f"{nick=}")

client = SocketClient(name=nick)

app = Main(client)

app.pack()

game\_thread = threading.Thread(target=app.mainloop)

socket\_thread = threading.Thread(

target=client.socket\_start, args=("192.168.118.80", 8081)

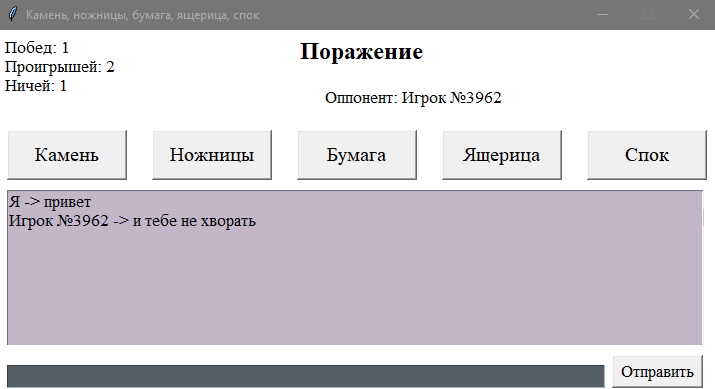
)

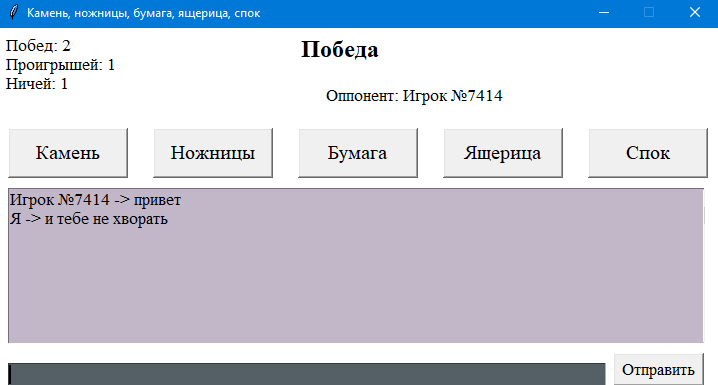
socket\_thread.start()

game\_thread.run()

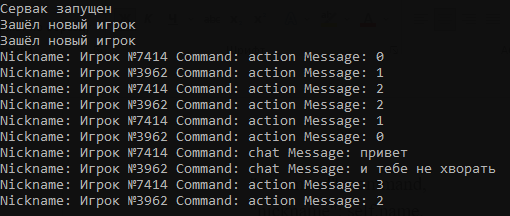
**Пример работы приложения**

Клиент:

****

****

Сервер:



**Заключение**

Мы научились связывать, при помощи программных средств, две вычислительных машины в локальной сети на примере игры «Камень, ножницы, бумага, ящерица, спок». Это поможет нам в дальнейшем использовать полученные, в ходе данной работы, навыки для дальнейшей работы.

**Список литературы**

1. Курс «Компьютерные сети» [Электронный ресурс] URL: http://math.gsu.by/wp-content/uploads/courses/networks/построение\_больших\_сетей

2. Статья «Протоколы передачи данных» [Электронный ресурс] URL: https://tproger.ru/explain/protokoly-peredachi-dannyh-chto-jeto-kakie-byvajut-i-v-chjom-razlichija/

3. Документация IBM «Маршрутизация TCP/IP» [Электронный ресурс] URL: https://www.ibm.com/docs/ru/aix/7.1?topic=protocol-tcpip-routing

4. Таблица «безопасных» цветов от студии Артемия Лебедева [Электронный ресурс] URL: https://www.artlebedev.ru/colors/

5. Конвертер цветов из одной цветовой модели в ряд других [Электронный ресурс] URL: https://colorscheme.ru/color-converter.html