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»

Тема: «Разработка онлайн-игры»

Выполнил студент 3 курса гр. ИС-30: Лоскутов А. В.

Проверил старший преподаватель: Морозов Н.С.

Нижний Новгород

2023 г.

**Содержание**

[Введение 3](#_heading=h.gjdgxs)

[Цель работы 3](#_heading=h.30j0zll)

[Задачи 3](#_heading=h.1fob9te)

[Теоретическая часть 4](#_heading=h.3znysh7)

[I. Протокол связи 4](#_heading=h.2et92p0)

[1. Модель OSI — Open Systems Interconnection 4](#_heading=h.tyjcwt)

[2. TCP/IP — Transmission Control Protocol/Internet Protocol 5](#_heading=h.3dy6vkm)

[3. UDP — User Datagram Protocol 5](#_heading=h.1t3h5sf)

[4. FTP — File Transfer Protocol 5](#_heading=h.4d34og8)

[5. DNS 6](#_heading=h.2s8eyo1)

[6. HTTP — HyperText Transfer Protocol 6](#_heading=h.17dp8vu)

[II. IP-адресация 6](#_heading=h.3rdcrjn)

[III. Маршрутизация TCP/IP 7](#_heading=h.26in1rg)

[Интерфейс приложения 9](#_heading=h.lnxbz9)

[Код программы 10](#_heading=h.35nkun2)

[I. Серверная часть приложения 10](#_heading=h.1ksv4uv)

[I. Клиентская часть приложения 12](#_heading=h.44sinio)

[Вывод 16](#_heading=h.2jxsxqh)

[Список литературы 17](#_heading=h.3j2qqm3)

# Введение

В настоящее время существует большое число так называемых «offline» игр. Все они характеризуются простотой правил, пошаговым выполнением ходов, малым числом игроков (обычно двое). С развитием сети Интернет, у данных игр появились сетевые аналоги.

Вся инфраструктура завязана на интернет-соединении, поэтому важным навыком, для специалиста, является умение взаимодействовать при помощи кода с другой вычислительной машиной. В данной работе представлен пример разработки небольшой онлайн игры.

# Цель работы

Целью данной курсовой работы является проектирование и реализация игры «Камень, ножницы, бумага» между двумя произвольными пользователями локальной сети. Должна быть реализована поддержка коммуникации по локальной сети (общий чат).

## Задачи

* Изучить систему маршрутизации;
* Изучить систему стека TCP/IP;
* Произвести настройку протокола транспортного уровня;
* Произвести настройку сокетов серверной и клиентской части игры;

## Теоретическая часть

## Протокол связи

Сетевой протокол — это набор правил и соглашений, используемых для связи устройств на определенном сетевом уровне. Протоколы обеспечивают и определяют формат обмена информацией между участниками компьютерных сетей. В работе сетей задействуется большое количество протоколов. Например, загрузка страницы в браузере — это результат работы, организованной согласно нескольким протоколам:

* согласно HTTP браузер формирует сообщение для сервера;
* согласно DNS браузер узнает IP-адрес сайта по его доменному имени;
* согласно TCP устанавливается соединение и гарантируется целостность передачи данных;
* согласно IP осуществляется адресация в сети;
* согласно Ethernet осуществляется физическая передача данных между устройствами в сети.

Множество протоколов можно классифицировать по сетевым уровням, на котором они работают. Наиболее распространенные сетевые модели — это OSI и TCP/IP. В рамках этого материала мы расскажем об этих моделях и опишем часто используемые протоколы

### Модель OSI — Open Systems Interconnection

На момент зарождения компьютерных сетей не существовало моделей, которые бы определяли общие стандарты работы сетей и подходы к их проектированию. Каждая компания, работавшая над созданием сетей, реализовывала собственные задумки, которые не могли работать с решениями от других создателей компьютерных сетей.

Более важным является то, что такое положение дел было проблемой. Сети, которые должны были объединять компьютеры, из-за архитектурных отличий создавали себе преграды для расширения. В 1977 году задачу по решению этой проблемы взяла на себя организация ISO (International Organization for Standardization) — международная организация по стандартизации. В течение 7 лет эта организация изучала реализации сетей того периода и в 1984 году представила модель OSI.

OSI — это аббревиатура Open Systems Interconnection, которая буквально переводится как «Взаимодействие открытых систем». Не следует путать «открытость» систем с понятием Open Source: система считается открытой, если она построена на основе общедоступных спецификаций,которые позволяет общаться двум системам вне зависимости от их архитектуры.

Модель состоит из 7 сетевых уровней, каждому из которых отведена своя роль и задачи. Это Физический, Канальный, Сетевой, Транспортный, Сеансовый, Представительный, Прикладной.

### TCP/IP — Transmission Control Protocol/Internet Protocol

TCP/IP (в некоторых источниках называется как «модель DoD»), как и OSI, реализует деление архитектуры на уровни. В RFC 1122 утверждена четырехуровневая архитектура указанного стека:

* прикладной уровень;
* транспортный уровень;
* межсетевой уровень, также иногда его называют просто сетевым или интернет;
* канальный уровень/сетевых интерфейсов/сетевого доступа.

### UDP — User Datagram Protocol

UDP-дейтаграмма состоит из 5 составляющих:

* Порт источника;
* Порт назначения;
* Длина заголовка;
* Контрольная сумма;
* Данные с вышестоящего уровня.

Задачи UDP — это проверка кадра и работа с портами. Из-за своей простоты протокол не может обеспечить контроль доставки данных. В отличие от TCP, UDP не запрашивает повторно побитые данные у отправителя.

### FTP — File Transfer Protocol

Протокол передачи файлов. Его использовали ещё в 1971 году — задолго до появления протокола IP. На текущий момент этим протоколом пользуются при удалённом доступе к хостингам. FTP является надёжным протоколом, поэтому гарантирует передачу данных.

Этот протокол работает по принципу клиент-серверной архитектуры. Пользователь проходит аутентификацию (хотя в отдельных случаях может подключаться анонимно) и получает доступ к файловой системе сервера.

### DNS

* DNS – система доменных имен, которая сообщает IP-адрес посредством вызова запроса с доменом;

### HTTP — HyperText Transfer Protocol

* HTTP – отвечает за клиент-серверное взаимодействие, передает гипертексты внутри WWW (доставляет документы через сервер браузеру); HTTP имеет расширение HTTPS, которое поддерживает шифрование. Данные в нём передаются поверх криптографического протокола TLS.

## IP-адресация

Сетевой протокол IP (Internet Protocol) появился в 80-х годах для объединения компьютерных сетей. Сейчас существуют 2 версии — IPv4 и IPv6:

* IPv4 — регулирует межсетевое взаимодействие посредством 32-битного поля адресов, где адресное пространство ограничено;
* IPv6 — усовершенствованный стандарт, где для адресной маршрутизации используется 128-битное поле, определяющее IPv6-адрес принимающего узла. В IPv6 ARP не используется.

Обе версии протокола предназначены для выполнения одних и тех же целей. IPv6 появился как альтернатива IPv4: в определенный момент IP-адреса стали заканчиваться. Сейчас эта проблема решается с помощью технологии NAT, и введение IPv6 как общего стандарта несколько затянулось. В этом материале будет разобрана первая версия IPv4. Протокол IP работает с блоком информации, который называется IP-пакет.

* IHL (Internet Header Length): многие поля заголовка не фиксированы и в этом поле указан его общий размер;
* Версия: IPv6 или IPv4;
* Тип обслуживания — данные для QoS;
* Длина пакета — размер пакета вместе с заголовком и данными;
* Идентификатор — число из 4 байтов, показывающее, что части разделенного пакета — это единое целое;
* Флаги — указывает на то, что пакет фрагментированный;
* Смещение фрагмента — сдвиг относительно первого фрагмента;
* Время жизни — максимальное количество прохождений через маршрутизаторы;
* Протокол — протокол транспортного уровня;
* Контрольная сумма заголовка — контрольная сумма заголовка;
* IP-адрес отправителя;
* IP-адрес получателя;
* Опции — поле для расширения стандартного заголовка, например, для специфичного оборудования;
* Смещение — к какой части принадлежит фрагмент;
* Данные.

## Маршрутизация TCP/IP

Маршрутизация — это процедура определения пути следования пакета из одной сети в другую. Такой механизм доставки становится возможным благодаря реализации во всех узлах сети протокола межсетевого обмена IP. Любое сообщение, которое отправляется по сети, должно быть при отправке разделено на фрагменты. Каждый из фрагментов должен быть снабжен адресами отправителя и получателя, а также номером этого пакета в последовательности пакетов, оставляющих все сообщение в целом.

Такая система позволяет на каждом шлюзе выбирать маршрут, основываясь на текущей информации о состоянии сети, что повышает надежность системы в целом. При этом каждый пакет может пройти от отправителя к получателю по своему собственному маршруту. Порядок получения пакетов получателем не имеет значения. Однако, существует особый тип оборудования, называемый маршрутизаторами (routегs), который применяется в сетях со сложной конфигурацией для связи ее участков с различными сетевыми протоколами, а также для более эффективного разделения трафика и использования альтернативных путей между узлами сети. Основная цель применения маршрутизаторов —объединение разнородных сетей и обслуживание альтернативных путей.

Маршрутизаторы не просто осуществляют связь разных типов сетей и обеспечивают доступ к глобальной сети, но и могут управлять трафиком на основе протокола сетевого уровня (третьего в модели OSI), то есть на более высоком уровне по сравнению с коммутаторами. Необходимость в таком управлении возникает при усложнении топологии сети и росте числа ее узлов, если в сети появляются избыточные пути, когда нужно решать задачу максимально эффективной и быстрой доставки отправленного пакета по назначению. При этом существует два основных алгоритма

Wide Area Network — досл. Сеть с широкой областью доступа определения наиболее выгодного пути и способа доставки данных: RIP и OSPF. При использовании протокола маршрутизации RIР, основным критерием выбора наиболее эффективного пути является минимальное число «хопов» (hops), т.е. сетевых устройств между узлами. Этот протокол минимально загружает процессор маршрутизатора и предельно упрощает процесс конфигурирования, но он не рационально управляет трафиком.

При использовании OSPF наилучший путь выбирается не только с точки зрения минимизации числа хопов, но и с учетом других критериев: производительности сети, задержки при передаче пакета и т.д. Сети большого размера, чувствительные к перегрузке трафика и базирующиеся на сложной маршрутизирующей аппаратуре, требуют использования протокола ОSРF. Реализации этого протокола возможна только на маршрутизаторах с достаточно мощным процессором, т.к. его реализация требует существенных процессинговых затрат.

Маршрутизация в сетях, как правило, осуществляться с применением пяти популярных сетевых протоколов — ТСР/IР, Nоvеll IРХ, АррlеТаlk II, DECnеt Phase IV и Хегох ХNS.

# Интерфейс приложения

Рис.1 Интерфейс игры «Камень, ножницы, бумага».

# Код программы

## Серверная часть приложения

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("127.0.0.1", 1020).listen()

## Клиентская часть приложения

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):

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\_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: {self.lose}\nНичей:" f" {self.draw}"

def send\_button(self):

input\_text = self.entry.get()

if not input\_text:

return

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

self.txt.insert(END, f"catherina => {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=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.master.bind("<Return>", self.send\_button)

self.game\_data\_label = Label(

self.root,

justify="left",

font=self.mini\_button\_font,

text=self.game\_data\_text(),

bg="#fff5ee",

)

self.opponent\_label = Label(

self.root,

justify="right",

font=self.mini\_button\_font,

text=f"Оппонент:\nНе найден",

bg="#ccffcc",

)

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

self.opponent\_label.place(x=300, y=0)

self.txt = tk.Text(

self.root, font=self.mini\_button\_font, width=45, height=8, bg="#fff5ee"

)

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

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

scrollbar = tk.Scrollbar(self.txt)

scrollbar.place(relheight=1, relx=0.958)

self.entry = tk.Entry(

self.root, font=self.mini\_button\_font, width=35, bg="#fff5ee"

)

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

send = Button(

self.root,

text="Отправить",

font=self.mini\_button\_font,

command=self.send\_button,

width=8,

height=1,

)

send.place(x=339, y=325)

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"]

self.game.opponent\_label.configure(text=f"Оппонент:\n {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("430x360+200+200")

main\_root.title("К\_Н\_Б")

main\_root.resizable(False, False)

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

nick = f"loskutov\_andrey"

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=("127.0.0.1", 1020)

)

socket\_thread.start()

game\_thread.run()

# Вывод

Цель курсового проекта была достигнута - была разработана несложная игра «Камень, ножницы, бумага», путем связывания двух вычислительных машин, для игры по локальной сети и коммуникации игроков с помощью чата. Полученные знания и навыки, в ходе курсового проекта, будут полезны для дальнейшей работы.

# Список литературы

1. Курс «Компьютерные сети» [Электронный ресурс] URL: [http://math.gsu.by/wp-content/uploads/courses/networks/построение\_больших\_сетей](http://math.gsu.by/wp-content/uploads/courses/networks/r5.2.html#:~:text=IP%2D%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%20%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%20%D1%81%D0%BE%D0%B1%D0%BE%D0%B9%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D0%BE%D0%B9,%D0%B2%D1%80%D0%B5%D0%BC%D1%8F%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BE%D0%B2%20%D0%B8%20%D0%BC%D0%B0%D1%80%D1%88%D1%80%D1%83%D1%82%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2.)
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. Статья «Вычислительная сеть» [Электронный ресурс] URL: [https://ru.wikipedia.org/wiki/Вычислительная\_сеть](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C)
6. Статья «Основные принципы работы сетевой маршрутизации»[Электронный ресурс] URL: https://pc.ru/docs/network/routing