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: «Разработка онлайн-игры»

Выполнил студент 3 курса гр. ИС-29: Уманский А.А.

Проверил старший преподаватель: Морозов Н.С.

**Содержание**

[Введение 3](#_Toc124985841)

[Задание 4](#_Toc124985842)

[Теоретическая часть 5](#_Toc124985843)

[Протокол связи 5](#_Toc124985844)

[1. IP — Internet Protocol 5](#_Toc124985845)

[2. TCP/IP — Transmission Control Protocol/Internet Protocol 5](#_Toc124985846)

[3. UDP — User Datagram Protocol 6](#_Toc124985847)

[4. FTP — File Transfer Protocol 6](#_Toc124985848)

[5. DNS — domain name system 6](#_Toc124985849)

[6. HTTP — HyperText Transfer Protocol 7](#_Toc124985850)

[IP-адресация 7](#_Toc124985851)

[Маршрутизация TCP/IP 9](#_Toc124985852)

[Маршрут до определенного IP 11](#_Toc124985853)

[Интерфейс приложения 12](#_Toc124985854)

[Код программы 13](#_Toc124985855)

[Вывод 19](#_Toc124985856)

[Список литературы 20](#_Toc124985857)

# Введение

В настоящее время почти все сферы жизнедеятельности основаны на интернет-соединении, поэтому один из важнейших навыков – это знание принципов и умение взаимодействовать при помощи кода с другой вычислительной машиной. Для развития и укрепления этого навыка напишем несложную игру для двух человек с использованием сокетов.

# Задание

* 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

### IP — Internet Protocol

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). Он классифицируется как протокол сетевого уровня по сетевой модели OSI. IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня — транспортного уровня сетевой модели OSI — например, TCP, которые используют IP в качестве транспорта.

Основная задача этого протокола — маршрутизация датаграмм, т. е. определение пути следования данных по узлам сети.

Наиболее популярная версия на текущий момент — IPv4 с 32-битными адресами. Это значит, что в интернете могут хранится 4.29 млрд адресов IPv4. Также существует версия IPv6, которая имеет 2 ^ 128 уникальных адресов.

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

Это стек протоколов TCP и IP. Первый обеспечивает и контролирует надёжную передачу данных и следит за её целостностью. Второй же отвечает за маршрутизацию для отправки данных. Протокол TCP часто используется более комплексными протоколами.

### UDP — User Datagram Protocol

один из ключевых элементов набора сетевых протоколов для Интернета. С UDP компьютерные приложения могут посылать сообщения (в данном случае называемые датаграммами) другим хостам по IP-сети без необходимости предварительного сообщения для установки специальных каналов передачи или путей данных.

Основное преимущество UDP протокола заключается в скорости доставки данных. Именно поэтому чувствительные к сетевым задержкам приложения часто используют этот тип передачи данных.

### 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, так как 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 — domain name system

Служба доменных имен — это стандартный протокол, который позволяет пользователям получать доступ к веб-сайтам, используя удобочитаемые адреса. Как телефонная книга позволяет найти имя контакта и узнать его телефонный номер, так и DNS позволяет ввести адрес веб-сайта и автоматически определить его IP-адрес, то есть уникальный идентификатор конкретного устройства (сервера) в компьютерной сети.

### HTTP — HyperText Transfer Protocol

HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в Интернете. HTTP является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (web-browser). Полученный итоговый документ будет (может) состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видео-файлов, скриптов и многого другого.

HTTP имеет расширение HTTPS, которое поддерживает шифрование. Данные в нём передаются поверх криптографического протокола TLS.

## IP-адресация

Стек протоколов TCP/IP предназначен для соединения отдельных

подсетей, построенных по разным технологиям канального и физического

уровней (Ethernet, Token Ring, FDDI, ATM, X.25 и т. д.), в единую сеть.

Каждая из технологий нижнего уровня предполагает свою схему адресации. Поэтому на межсетевом уровне требуется единый способ адресации,

позволяющий уникально идентифицировать каждый узел, входящий в со

ставную сеть. Таким способом в TCP/IP-сетях является IP-адресация. Узел

составной сети, имеющий IP-адрес, называется хост (host).

Хороший пример, иллюстрирующий составную сеть, - международная почтовая система адресации. Информация сетевого уровня - это индекс страны, добавленный к адресу письма, написанному на одном из тысяч языков земного шара, например, на китайском. И даже если это письмо

должно пройти через множество стран, почтовые работники которых не

знают китайского, понятный им индекс страны-адресата подскажет, через

какие промежуточные страны лучше передать письмо, чтобы оно кратчайшим путем попало в Китай. А уже там работники местных почтовых

отделений смогут прочитать точный адрес, указывающий город, улицу,

дом и человека, и доставить письмо адресату, так как адрес написан на

языке и в форме, принятой в данной стране.

В стеке TCP/IP используются три типа адресов: локальные (называемые также аппаратными), IP-адреса и символьные доменные имена.

В терминологии TCP/IP под локальным адресом понимается такой тип адреса, который используется средствами базовой технологии для доставки данных в пределах подсети, являющейся элементом составной интерсети. В разных подсетях допустимы разные сетевые технологии, разные стеки протоколов, поэтому при создании стека TCP/IP предполагалось наличие разных типов локальных адресов. Если подсетью интерсети является локальная сеть, то локальный адрес — это МАС - адрес. МАС - адрес назначается сетевым адаптерам и сетевым интерфейсам маршрутизаторов. МАС - адреса назначаются производителями оборудования и являются уникальными, так как управляются централизованно. Для всех существующих технологий локальных сетей МАС - адрес имеет формат 6 байт, например 11-AO-17-3D-BC-01.

Однако протокол IP может работать и над протоколами более высокого уровня, например над протоколом IPX или Х.25. В этом случае локальными адресами для протокола IP соответственно будут адреса IPX и Х.25. Следует учесть, что компьютер в локальной сети может иметь несколько локальных адресов даже при одном сетевом адаптере. Некоторые сетевые устройства не имеют локальных адресов. Например, к таким устройствам относятся глобальные порты маршрутизаторов, предназначенные для соединений типа «точка-точка».

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень передает пакеты между сетями. Эти адреса состоят из 4 байт, например 109.26.17.100. IP-адрес 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 (Internet Network Information Center, InterNIC), если сеть должна работать как составная часть Internet.

Обычно поставщики услуг Internet получают диапазоны адресов у подразделений InterNIC, а затем распределяют их между своими абонентами. 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.

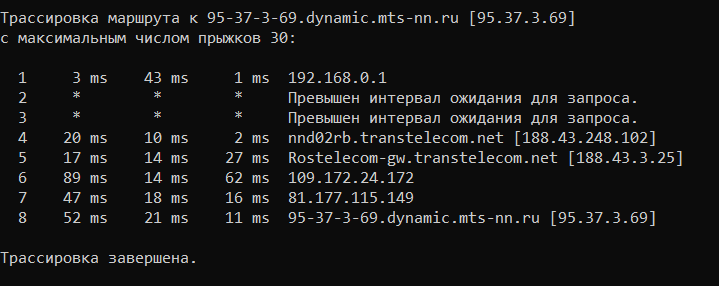
## Маршрутизация TCP/IP

Маршрутом называется путь, по которому пакеты пересылаются от отправителя к получателю. Маршрут определяет не полный путь, а только сегмент пути от хоста до шлюза (или от шлюза до шлюза), который может переслать пакеты целевому хосту. Описание маршрута содержит такую информацию, как список сетей, достижимых локальным хостом, и список шлюзов для отправки пакетов в удаленные сети. При получении дейтаграммы шлюз ищет в таблицах маршрутизации следующий узел ее маршрута до целевого хоста и отправляет дейтаграмму этому узлу.

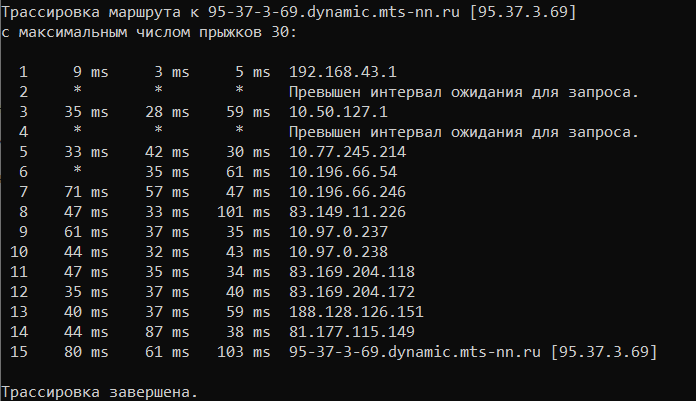
В таблицу маршрутизации ядра можно добавлять несколько маршрутов к одному и тому же хосту. Процедура выбора маршрута сначала находит все маршруты, соответствующие запросу, а потом выбирает маршрут с минимальной метрикой расстояния. При наличии нескольких маршрутов одинаковой длины выбирается тот маршрут, который задан наиболее точно. Если несколько маршрутов совпадают по обоим критериям, то эти маршруты применяются по очереди.

# Маршрут до определенного IP

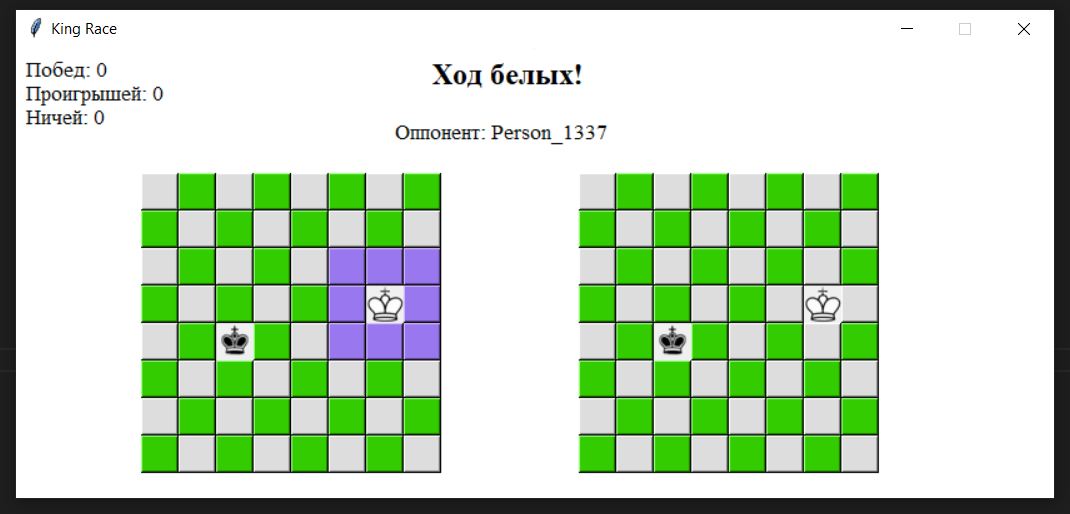
1) Через беспроводную сеть



2) Через мобильный интернет



# Интерфейс приложения



# Код программы

from tkinter import \*

import tkinter as tk

import threading

import socket as sk

import time

class Main(Frame):

    def \_\_init\_\_(self, root):

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

        self.opponent\_name = 'Person\_1337'

        self.set\_my\_name('andrey\_ciferki')

        self.startUI()

        self.opponent\_choise = IntVar()

    def startUI(self):

        self.game\_btns = [Button(root, text="")]

        self.game\_btns\_opp = [Button(root, text="")]

        self.movies=0

        #доска

        self.game\_btns.clear()

        self.game\_btns\_opp.clear()

        for i in range(64):

            if i//8%2==0 and i%2==0:

                self.game\_btns.append(Button(root, bg='#ddd', command=lambda x=i: self.btn\_click(x)))

            elif i//8%2==0 and i%2==1:

                self.game\_btns.append(Button(root, bg='#3c0', command=lambda x=i: self.btn\_click(x)))

            elif i//8%2==1 and i%2==0:

                self.game\_btns.append(Button(root, bg='#3c0', command=lambda x=i: self.btn\_click(x)))

            elif i//8%2==1 and i%2==1:

                self.game\_btns.append(Button(root, bg='#ddd', command=lambda x=i: self.btn\_click(x)))

            self.game\_btns[i].place(x=100+(i%8)\*30, y=100+(i//8)\*30, width=30, height=30)

        #доска 2

        for i in range(64):

            if i//8%2==0 and i%2==0:

                self.game\_btns\_opp.append(Button(root, bg='#ddd', command=lambda x=i: self.btn\_click\_opp(x)))

            elif i//8%2==0 and i%2==1:

                self.game\_btns\_opp.append(Button(root, bg='#3c0', command=lambda x=i: self.btn\_click\_opp(x)))

            elif i//8%2==1 and i%2==0:

                self.game\_btns\_opp.append(Button(root, bg='#3c0', command=lambda x=i: self.btn\_click\_opp(x)))

            elif i//8%2==1 and i%2==1:

                self.game\_btns\_opp.append(Button(root, bg='#ddd', command=lambda x=i: self.btn\_click\_opp(x)))

            self.game\_btns\_opp[i].place(x=350+100+(i%8)\*30, y=100+(i//8)\*30, width=30, height=30)

        #короли

        self.img\_w\_king = PhotoImage(file='C:/Users/User/Desktop/веб и другое/bigD/kurs\_rab/white\_king.png')

        self.white\_king=Label(root, image=self.img\_w\_king)

        self.img\_b\_king = PhotoImage(file='C:/Users/User/Desktop/веб и другое/bigD/kurs\_rab/black\_king.png')

        self.black\_king=Label(root, image=self.img\_b\_king)

        self.w\_king\_pos=59

        self.b\_king\_pos=3

        #короли 2

        self.white\_king\_opp=Label(root, image=self.img\_w\_king)

        self.black\_king\_opp=Label(root, image=self.img\_b\_king)

        #остальное

        self.lbl = Label(root, text="Ход белых!", bg="#FFF",

                         font=("Times New Roman", 18, "bold"))

        self.lbl.place(x=330, y=5)

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

        self.lbl2 = Label(root, justify="left", font=("Times New Roman", 13),

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

                              f" {self.lose}\nНичей: {self.drow}",

                         bg="#FFF")

        self.lbl3 = Label(root, justify="right", font=("Times New Roman", 13),

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

                         bg="#FFF")

        self.lbl2.place(x=5, y=5)

        self.lbl3.place(x=300, y=55)

        self.new()

    def btn\_unblock(self,pos,pos\_opp):

        i=0

        for btn in self.game\_btns:

            if (i==pos-9)or(i==pos-8)or(i==pos-7)or(i==pos-1)or(i==pos+1)or(i==pos+7)or(i==pos+8)or(i==pos+9):

                if (i!=pos\_opp-9)and(i!=pos\_opp-8)and(i!=pos\_opp-7)and(i!=pos\_opp-1)and(i!=pos\_opp+1)and(i!=pos\_opp+7)and(i!=pos\_opp+8)and(i!=pos\_opp+9):

                    btn['state'] = tk.NORMAL

                    btn['bg']='#97e'

            else:

                btn['state'] = tk.DISABLED

            if (abs(pos%8-(i)%8)>1):

                btn['state'] = tk.DISABLED

                btn['bg']=self.game\_btns\_opp[i]['bg']

            i+=1

    def btn\_unblock1(self,pos,pos\_opp):

        i=0

        for btn in self.game\_btns\_opp:

            if (i==pos-9)or(i==pos-8)or(i==pos-7)or(i==pos-1)or(i==pos+1)or(i==pos+7)or(i==pos+8)or(i==pos+9):

                if (i!=pos\_opp-9)and(i!=pos\_opp-8)and(i!=pos\_opp-7)and(i!=pos\_opp-1)and(i!=pos\_opp+1)and(i!=pos\_opp+7)and(i!=pos\_opp+8)and(i!=pos\_opp+9):

                    btn['state'] = tk.NORMAL

                    btn['bg']='#97e'

            else:

                btn['state'] = tk.DISABLED

            if (abs(pos%8-(i)%8)>1):

                btn['state'] = tk.DISABLED

                btn['bg']=self.game\_btns[i]['bg']

            i+=1

    #для себя

    def btn\_click(self, choise):

        self.lbl.configure(text="Ход черных!")

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

        #выделение полей

        i=0

        for btn in self.game\_btns:

            btn['bg']=self.game\_btns\_opp[i]['bg']

            i+=1

        #конец выделения

        self.choise = choise

        self.w\_king\_pos=choise

        self.white\_king.place(x=100+(choise%8)\*30, y=100+(choise//8)\*30, width=30, height=30)

        self.white\_king\_opp.place(x=350+100+(choise%8)\*30, y=100+(choise//8)\*30, width=30, height=30)

        if (self.w\_king\_pos<8):

            self.lbl.configure(text="Победа белых!")

            self.win+=1

            self.new()

        else:

            self.btn\_unblock1(self.b\_king\_pos,self.w\_king\_pos)

    #для противника

    def btn\_click\_opp(self, choise):

        self.lbl.configure(text="Ход белых!")

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

        #выделение полей

        i=0

        for btn in self.game\_btns\_opp:

            btn['bg']=self.game\_btns[i]['bg']

            i+=1

        #конец выделения

        self.opponent\_choise = choise

        self.b\_king\_pos=choise

        self.black\_king.place(x=100+(choise%8)\*30, y=100+(choise//8)\*30, width=30, height=30)

        self.black\_king\_opp.place(x=350+100+(choise%8)\*30, y=100+(choise//8)\*30, width=30, height=30)

        self.block\_opp()

        self.movies+=1

        if (self.b\_king\_pos>55):

            self.lbl.configure(text="Победа черных!")

            self.lose+=1

            self.new()

        elif (self.movies>12):

            self.lbl.configure(text="Ничья!")

            self.drow+=1

            self.new()

        else:

            self.btn\_unblock(self.w\_king\_pos,self.b\_king\_pos)

    #новая игра

    def new(self):

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

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

        self.set\_opponent\_choise = IntVar()

        self.w\_king\_pos=59

        self.b\_king\_pos=3

        self.white\_king.place(x=100+(3%8)\*30, y=100+(7)\*30, width=30, height=30)

        self.white\_king\_opp.place(x=350+100+(3%8)\*30, y=100+(7)\*30, width=30, height=30)

        self.black\_king.place(x=100+(3%8)\*30, y=100+(0)\*30, width=30, height=30)

        self.black\_king\_opp.place(x=350+100+(3%8)\*30, y=100+(0)\*30, width=30, height=30)

        self.btn\_unblock(self.w\_king\_pos,self.b\_king\_pos)

        self.block\_opp()

        self.movies=0

    def block\_me(self):

        for btn in self.game\_btns:

            btn['state']=DISABLED

    def block\_opp(self):

        for btn in self.game\_btns\_opp:

            btn['state']=DISABLED

    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):

        self.opponent\_choise=opp\_choise

    def get\_opponent\_choise(self):

        return self.opponent\_choise.get()

def socket\_start():

    global app

    def client():

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

        sock.connect(('127.0.0.1', 32145))

        data2="no"

        while True:

            daat=str(app.opponent\_choise)

            sock.send(daat.encode())

            data = sock.recv(140)

            if data2!=data:

                print(data)

            data2=data

            time.sleep(0.5)

    def server():

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

        sock1.bind(('127.0.0.1',32145))

        sock1.listen(10)

        print('Server is running')

        conn, addr = sock1.accept()

        print('connected:', addr)

        while True:

            data = conn.recv(140)

            data=data.decode()

            conn.send("opp\_choise:".encode()+data.encode())

            if not data:

                break

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

    server\_thread = threading.Thread(target=server)

    client\_thread = threading.Thread(target=client)

    server\_thread.daemon=True

    client\_thread.daemon=True

    server\_thread.start()

    client\_thread.start()

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

    root = Tk()

    root.geometry("830x360+200+200")

    root.title("King Race")

    root.resizable(False, False)

    root["bg"] = "#FFF"

    app = Main(root)

    app.pack()

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

    socket\_thread.start()

    game\_thread = threading.Thread(target=root.mainloop(), args=(root,))

    game\_thread.start()

# Вывод

В результате выполнения данного задания мы научились связывать при помощи программных средств несколько вычислительных машины в локальной сети. Для примера была выбрана игра «King Race» (гонка королей). Этот полезный опыт обязательно пригодится в будущем для более сложных и необходимых задач.

# Список литературы

1. Документация IBM «Маршрутизация TCP/IP» [Электронный ресурс] URL: <https://www.ibm.com/docs/ru/aix/7.1?topic=protocol-tcpip-routing>
2. Конвертер цветов из одной цветовой модели в ряд других [Электронный ресурс] URL: <https://colorscheme.ru/color-converter.html>
3. 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" \l ":~: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.)
4. Статья «Протоколы передачи данных» [Электронный ресурс] URL: [https://tproger.ru/explain/protokoly-peredachi-dannyh-chto-jeto-kakie-byvajut-i-v-chjom-razlichija/](https://tproger.ru/explain/protokoly-peredachi-dannyh-chto-jeto-kakie-byvajut-i-v-chjom-razlichija/Д)
5. Таблица «безопасных» цветов от студии Артемия Лебедева [Электронный ресурс] URL: <https://www.artlebedev.ru/colors/>
6. Что такое DNS: гайд по технологии» [Электронный ресурс] URL: <https://mcs.mail.ru/blog/chto-takoe-dns-tri-bukvy-na-kotoryh-derzhitsya-internet>