Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Петрозаводский государственный университет»

Физико-технический институт

Кафедра информационно-измерительных систем и физической электроники

**СЕТЕВАЯ ИГРА НА БАЗЕ ПРОТОКОЛА UDP**

Курсовой проект по дисциплине «Сети ЭВМ и телекоммуникации»

Автор работы:

Студент группы 21312

С. Лапин

Преподаватель:

Д. И. Трофимов 

г. Петрозаводск 2023

**Содержание**

1. [Введение 3](#_Toc136796257)
2. [Описание файла Protocol 4](#_Toc136796258)
3. [Описание файла конфигурации (настроек) игры 5](#_Toc136796259)
4. [Описание класса Server 5](#_Toc136796260)
5. [Описание класса Client 6](#_Toc136796261)
6. [Описание функции main 7](#_Toc136796262)
7. [Описание класса Pong 7](#_Toc136796263)
8. [Игровой процесс 11](#_Toc136796264)
9. [Заключение 12](#_Toc136796265)
10. [Приложение 1. Ссылка на git-репозиторий 13](#_Toc136796266)
11. [Приложение 2. Листинг файла settings.py 13](#_Toc136796267)
12. [Приложение 3. Листинг файла protocol.py 13](#_Toc136796268)
13. [Приложение 4. Листинг файла server.py 13](#_Toc136796269)
14. [Приложение 5. Листинг файла client.py 14](#_Toc136796270)
15. [Приложение 6. Листинг основных методов класса Pong 15](#_Toc136796271)
16. [Приложение 7. Листинг файла main.py 17](#_Toc136796272)

# Введение

Цель работы.

Создание сетевого приложения на языке программирования Python.   
В качестве сетевого приложения была выбрана сетевая игра «Пинг-понг».

Описание приложения

Приложение имеет архитектуру P2P, для игры требуется 2 пользователя: хост и игрок.

Передача данных осуществляется в пределах локальной сети при помощи протокола UDP и специального протокола, который в UDP-заголовке данных отправляет необходимую для игры информацию.

Используемые порты: 21312, 21313, 21314.

# Описание файла Protocol

Класс Protocol отвечает за представление данных в необходимом виде.

В зависимости от роли пользователя, данные могут принимать следующий вид:

* Хост

Таблица 1. Заголовки специального протокола хоста

|  |  |  |  |
| --- | --- | --- | --- |
| Позиция мяча по оси Х (2 байта) | Позиция мяча по оси Y (2 байта) | Позиция игрока по оси Y (2 байта) | Указатель конца игры (1 байт) |

* Игрок

Таблица 2. Заголовки специального протокола игрока.

|  |  |  |
| --- | --- | --- |
| Резерв (4 байта) | Позиция игрока по оси Y (2 байта) | Указатель конца игры (1 байт) |

Хост, помимо своей позиции на игровом поле, также отвечает за учет позиции мяча, в отличие от игрока.

**Функции:**

* serialize

Функция возвращает строку в необходимом порядке расположения байт.

Принимает следующие параметры: роль пользователя и информацию, которую необходимо передать. Возвращает строку в необходимом порядке расположения байт.

* deserialize

Функция возвращает кортеж данных в необходимых типах данных.

Принимает следующие параметры: роль пользователя и поток байт.

# Описание файла конфигурации (настроек) игры

Файл settings.py хранит в себе константы, конфигурирующие игру:

* GAME\_WIDTH и GAME\_HEIGHT – размеры игрового поля;
* PORT, SERVER\_PORT, CLIENT\_PORT – номера портов, используемые при сетевом взаимодействии;
* IP – IP-адрес устройства пользователя.

# Описание класса Server

Класс server (хост) содержит методы для обмена пакетами с игроком.

**Поля класса:**

* ip

Данное поле принимает значение IP-адреса компьютера, принадлежащего хосту.

* client\_ip

Данное поле принимает значение IP-адреса компьютера, принадлежащего игроку.

**Методы класса:**

* send\_broadcast

Метод отправляет пакет на широковещательный канал, порт 21312.

Данный пакет необходим для того, чтобы игрок мог подключиться к хосту.

* get\_client\_ip

Метод прослушивает порт 21313 и принимает пакет.

Проанализировав пакет, полю client\_ip присваивается значение IP-адреса отправителя.

* handshake\_with\_client

Метод отправляет пакет игроку, содержащий следующее сообщение: «Hello, I`m using port 21314. Server».

Данное сообщение является своеобразным рукопожатием, который необходим для подтверждения соединения между пользователями и определения порта, на который будут отправляться пакеты хостом.

* send\_packet

Принимаемые параметры: данные в необходимом формате, порт клиента.

Метод отправляет пакет (на установленный порт) игроку, в котором содержится поток байт с информацией.

* receive\_packet

Метод прослушивает установленный игроком порт (21313) и возвращает поток данных.

# Описание класса Client

Класс client (игрок) содержит методы для обмена пакетами с хостом.

**Поля класса:**

* server\_ip

Данное поле принимает значение IP-адреса компьютера, принадлежащего хосту.

**Методы класса:**

* connect\_to\_server

Метод прослушивает порт 21312 и принимает пакет.

Проанализировав пакет, полю server\_ip присваивается значение IP-адреса отправителя.

* handshake\_with\_server

Метод отправляет пакет хосту, содержащий следующее сообщение: «Hello, I`m using port 21313. Client».

Данное сообщение является своеобразным рукопожатием, который необходим для определения порта, на который будут отправляться пакеты игроком.

* connection\_ack

Метод прослушивает порт 21314 и принимает пакет.

Проанализировав пакет, при совпадении значения IP-адреса отправителя с полем server\_ip, возвращается True.

* send\_packet

Принимаемые параметры: данные в необходимом формате, порт хоста.

Метод отправляет пакет (на установленный порт) хосту, в котором содержится поток байт с информацией.

* receive\_packet

Метод прослушивает установленный игроком порт (21314) и возвращает поток данных.

# Описание функции main

При запуске приложение, предлагается выбор: присоединиться к игре, или подключиться к уже существующей (в дальнейшем – это указатель на роль игрока). Далее, происходит запуск игры.

# Описание класса Pong

Класс Pong содержит методы геймплея и взаимодействия пользователя с игрой.

**Поля класса:**

* field

Данное поле является двумерным массивом заданного в настройках размера и служит для визуализации игрового поля в консоли.

* Ball

Данное поле является структурой, в которой в свойстве «x» хранится координата мяча по оси Х, в свойстве «y» хранится координата мяча по оси Y, а в свойстве «dir» хранится направление движения мяча.

* gameover

Данное поле является указателем конца игры. При запуске игры принимает значение False.

* is\_server

Данное поле является указателем роли пользователя.

* socket / client\_socket

Данное поле является экземпляром класса в соответствии с ролью пользователя.

* connection

Данное поле является указателем соединения.

* player

Данное поле является структурой, в которой в свойстве «x» хранится координата игрока по оси Х, а в свойстве «y» хранится координата игрока по оси Y.

* opponent

Данное поле является структурой, в которой в свойстве «x» хранится координата оппонента по оси Х, а в свойстве «y» хранится координата оппонента по оси Y.

* MODE

Данное поле является указателем игрового режима (бесконечный/до первого пропуска).

**Методы класса:**

* \_\_init\_connection

Приватный метод. Принимаемый параметр: указатель роли пользователя.

Метод устанавливает соединение между пользователями:

* + Роль хоста

Инициализируются необходимые для игры поля (is\_server = True, socket = Server(), connection = False, player, opponent). Далее, раз в 5 секунд выполняется метод «Server» send\_broadcast и одновременно выполняется метод get\_client\_ip. При успешной установке соединения, отправляется рукопожатие и метод завершает свою работу.

* + Роль игрока

Инициализируются необходимые для игры поля (is\_server = False, client\_socket = Client()). Далее, выполняется метод connect\_to\_server и затем отправляется рукопожатие. При успешной установке соединения, выводится IP-адрес хоста и метод завершает свою работу.

* \_\_init\_\_

Конструктор класса. Первым делом вызывается метод \_\_init\_connection. Далее выполняется заполнение игрового поля (field, player, opponent). Если роль игрока – хост, также предоставляется выбор игрового режима.

* show

Метод отвечает за вывод игрового поля на экран.

* update

Метод отвечает за обновление игрового поля на экране и сетевое взаимодействие.

В зависимости от роли пользователя имеет следующее поведение:

* + Роль хоста

Первым делом, вызывается метод \_\_update\_ball. Далее, происходит обмен пакетами с игроком (хост отправляет свою позицию и позицию мяча, а принимает лишь позицию игрока). Затем, происходит обновление игрового поля.

* + Роль игрока

Игрок принимает и обрабатывает пакет хоста и, при условии, что еще не конец игры, отправляет свои координаты и обновляет игровое поле.

* \_\_update\_ball

Приватный метод. Отвечает за логику обновление позиции мяча в зависимости от направления его движения.

Первым делом, вызывается метод logic. Далее, раз в 0,25 секунд + пинг между хостом и игроком, происходит обновление позиции мяча.

* logic

Приватный метод. Отвечает за логику отскока мяча, коллизии мяча с игроками и режим игры.

Также, выставляет указатель конца игры, при пропущенном мяче.

* movement

Данный метод отвечает за взаимодействие пользователя с игрой с помощью клавиатуры в бесконечном цикле.

При выставленном указателе конца игры, прекращает свою работу.

* start

Данный метод отвечает за весь игровой процесс.

# Игровой процесс

При запуске приложения, нас приветствует окно, в котором необходимо выбрать, создать игру, или подключится к уже существующей. Выбор происходит с помощью ввода значений с клавиатуры, согласно подсказкам.



Рисунок 1. Окно приветствия

Далее, необходимо дождаться установления соединения, и, в случае создания игры, выбрать игровой режим.

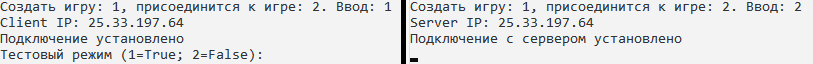


Рисунок 2. Окна выбора роли пользователя

Затем, на экран выводится игровое поле и пользователь может с ним взаимодействовать, путем нажатия клавиш «W» - двигаться вверх и «S» - двигаться вниз. В зависимости от роли пользователя, его положение, будет в разных частях экрана. Роль хоста – в левой части, роль игрока – в правой.

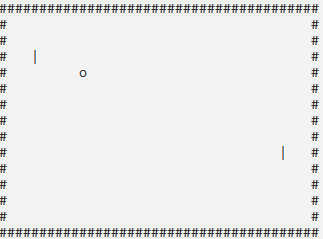


Рисунок 3. Игровое поле.

# Заключение

Результатом курсового проекта стало написание консольной P2P сетевой игры. Для обмена данными, использовался протокол UDP и специальный протокол на его основе. Реализованный функционал позволяет играть сессию игры с двух различных компьютеров в пределах одной локальной сети.

# Приложение 1. Ссылка на git-репозиторий

<https://github.com/Farlom/NetsCourseProject>

В репозитории можно найти видео-демонстрацию работы приложения, а так же исходные файлы.

# Приложение 2. Листинг файла settings.py

import socket  
from msvcrt import getch

GAME\_WIDTH = 40  
GAME\_HEIGHT = 15  
PORT = 21312  
SERVER\_PORT = 21313  
CLIENT\_PORT = 21314  
IP = socket.gethostbyname(socket.gethostname())

# Приложение 3. Листинг файла protocol.py

def serialize(player: bool, data: tuple) -> str:  
 if player:  
 return f'{data[0]:02d}{data[1]:02d}{data[2]:02d}0'  
 else:  
 return f'0000{data[0]:02d}{data[1]}'

def deserialize(player: bool, data):  
 if player:  
 return int(data[4:6])  
 else:  
 return int(data[0:2]), int(data[2:4]), int(data[4:6]), bool(int(data[6:7]))

# Приложение 4. Листинг файла server.py

import socket  
import settings

class Server:  
 ip = settings.IP  
 client\_ip = None

def send\_broadcast(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1)  
 destination\_address = ('<broadcast>', settings.PORT)  
 sock.sendto(settings.IP.encode(), destination\_address)  
 sock.close()

def send\_packet(self, message, port=settings.CLIENT\_PORT):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 destination\_address = (self.client\_ip, port)  
 sock.sendto(message.encode(), destination\_address)  
 sock.close()

def get\_client\_ip(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.bind(('', settings.SERVER\_PORT))  
 data = 0  
 while data == 0:  
 m, addr = sock.recvfrom(1024)  
 data = addr[0]  
 self.client\_ip = data  
 sock.close()

def handshake\_with\_client(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 destination\_address = (self.client\_ip, settings.CLIENT\_PORT)  
 sock.sendto(f'Hello, I`m using port {settings.CLIENT\_PORT}. Server'.encode(), destination\_address)  
 sock.close()

def recieve\_packet(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.bind(('', settings.SERVER\_PORT))  
 data = 0  
 while data == 0:  
 m = sock.recvfrom(7)  
 data = m[0].decode()  
 sock.close()  
 return data

# Приложение 5. Листинг файла client.py

import socket  
import settings

class Client:  
 server\_ip = None

def send\_packet(self, message, port=settings.SERVER\_PORT):  
 destination\_address = (self.server\_ip, port)  
 sending\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, socket.IPPROTO\_UDP)  
 sending\_socket.sendto(message.encode(), destination\_address)  
 sending\_socket.close()

def connect\_to\_server(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.bind(('', settings.PORT))  
 data = 0  
 while data == 0:  
 m = sock.recvfrom(14)  
 data = m[0].decode()  
 self.server\_ip = data  
 sock.close()

def handshake\_with\_server(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 destination\_address = (self.server\_ip, settings.SERVER\_PORT)  
 sock.sendto(f'Hello, I`m using port {settings.SERVER\_PORT}. Client'.encode(), destination\_address)  
 sock.close()

def connection\_ack(self) -> bool:  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.bind(('', settings.CLIENT\_PORT))  
 data = 0  
 while data == 0:  
 m, addr = sock.recvfrom(35)  
 data = m.decode()  
 if addr[0] == self.server\_ip:  
 sock.close()  
 return True

def recieve\_packet(self):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 sock.bind(('', settings.CLIENT\_PORT))  
 data = 0  
 while data == 0:  
 m = sock.recvfrom(7)  
 data = m[0].decode()  
 sock.close()  
 return data

# Приложение 6. Листинг основных методов класса Pong

def \_\_init\_conn(self, choice=2):  
 if choice == 1:  
 self.is\_server = True  
 self.socket = Server()  
 self.connection = False  
 self.player = Struct(4, int(settings.GAME\_HEIGHT / 2))  
 self.opponent = Struct(settings.GAME\_WIDTH - 5, int(settings.GAME\_HEIGHT / 2))

def send\_broadcast():  
 while self.socket.client\_ip is None:  
 self.socket.send\_broadcast()  
 time.sleep(5)  
 self.connection = True  
 print('Подключение установлено')

def get\_client\_ip():  
 self.socket.get\_client\_ip()  
 print(f'Client IP: {self.socket.client\_ip}')

thread\_broad = Thread(target=send\_broadcast)  
 thread\_client = Thread(target=get\_client\_ip)  
 thread\_broad.start()  
 thread\_client.start()

while True:  
 if thread\_broad.is\_alive():  
 time.sleep(1)  
 else:  
 break

self.socket.handshake\_with\_client()  
 elif choice == 2:  
 self.is\_server = False  
 self.opponent = Struct(4, int(settings.GAME\_HEIGHT / 2))  
 self.player = Struct(settings.GAME\_WIDTH - 5, int(settings.GAME\_HEIGHT / 2))  
 self.client\_socket = Client()  
 self.client\_socket.connect\_to\_server()  
 print(f'Server IP: {self.client\_socket.server\_ip}')  
 self.client\_socket.handshake\_with\_server()  
 if self.client\_socket.connection\_ack():  
 print('Подключение с сервером установлено')

def update(self):  
 if self.is\_server:  
 self.\_\_update\_ball()  
 self.socket.send\_packet(protocol.serialize(self.is\_server, (self.ball.x, self.ball.y, self.player.y)))  
 self.field[self.opponent.y][self.opponent.x] = ' '  
 self.opponent.y = protocol.deserialize(self.is\_server, self.socket.recieve\_packet())  
 self.field[self.opponent.y][self.opponent.x] = '|'  
 self.show()  
 else:  
 self.field[self.ball.y][self.ball.x] = ' '  
 self.field[self.opponent.y][self.opponent.x] = ' '  
 self.ball.x, self.ball.y, self.opponent.y, self.gameover = protocol.deserialize(self.is\_server, self.client\_socket.recieve\_packet())

if not self.gameover:  
 self.field[self.ball.y][self.ball.x] = 'o'  
 self.field[self.opponent.y][self.opponent.x] = '|'  
 self.show()  
 self.client\_socket.send\_packet(protocol.serialize(self.is\_server, (self.player.y, 0)))

# Приложение 7. Листинг файла main.py

from game import Pong

if \_\_name\_\_ == '\_\_main\_\_':  
 choice = int(input('Создать игру: 1, присоединится к игре: 2. Ввод: '))  
 game = Pong(choice)  
 game.start()