Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 5

по курсу «Компьютерные сети»

«Реализация WebSocket клиента и сервера на языке Golang»

Студент группы ИУ9-31Б Горбунов А. Д.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

Реализовать сетевую службу на языке программирования Golang взаимодействующую по протоколу связи WebSocket по вариантам. Клиентское приложение получает через стандартный поток ввода данные и в формате JSON передает их на сервер, сервер выполняет вычисления и возвращает результат обратно клиенту, который в свою очередь выводит полученный результат в стандартный поток вывода.

Для тестирования можно использовать исходные коды программ на языке Python продемонстрированные на лекции.

Входные данные: Результат эксперимента представляющий из себя последовательность действительных чисел произвольной длинны.

Функционал сервера: Среднее значение, среднеквадратичное отклонение и относительную погрешность эксперимента в процентах.

2 Результаты

Исходный код программы представлен в листинге 1, 2

Листинг 1 — client.go

```
1  package main
2  import (
3    "encoding/json"
4    "fmt"
5    "log"
6    "github.com/gorilla/websocket"
7    "github.com/skorobogatov/input"
8  )
9  var addr = ""
10 func sendMessageToServer(numbers []float64) ([]byte, error) {
11   c, _, err := websocket.DefaultDialer.Dial(addr, nil)
12   if err != nil {     return nil, err  }
13   defer c.Close()
14   requestBody := map[string]interface{}{"numbers": numbers}
15   jsonBody, err := json.Marshal(requestBody)
16   if err != nil {     return nil, err  }
17   err = c.WriteMessage(websocket.TextMessage, jsonBody)
18   if err != nil {     return nil, err  }
19   _, response, err := c.ReadMessage()
20   if err != nil {     return nil, err  }
21   return response, nil
22 }
23 func main() {
24     fmt.Print("IP: ")
25     ip := ""
26     fmt.Scan(&ip)
27     fmt.Print("Port: ")
28     port := ""
29     fmt.Scan(&port)
30     addr = "ws://" + ip + ":" + port
31   for {
32     fmt.Println("Enter the command ('quit' to exit, 'calc' to send data)
       :")
33     command := input.Gets()
34         switch command {
35     case "quit":
36       return
37     case "calc":
38       var numbers []float64
39       var n int
40       fmt.Print("Enter the number of numbers: ")
41       fmt.Scanln(&n)
42       fmt.Println("Enter numbers:")
43       for i := 0; i < n; i++ {
44         var num float64
45         fmt.Scanln(&num)
46         numbers = append(numbers, num)
47       }
48       response, err := sendMessageToServer(numbers)
49       if err != nil {
50         log.Fatal("Error sending message to server:", err)
51       }
52       fmt.Println("Response from the server:", string(response))
53     default:
54       fmt.Println("Error: unknown command")
55     }
56   }
57 }
```

Листинг 2 — server.py

```python
import asyncio
import math
import websockets
import json

async def calculate_experiment_results(data):
    numbers = data['numbers']
    result = []
    average = sum(numbers) / len(numbers)
    result.append(f"Average: {average}")

    sum_squares = sum((num - average) ** 2 for num in numbers)
    mean_square_deviation = math.sqrt(sum_squares / len(numbers))
    result.append(f"Standard deviation: {mean_square_deviation}")

    relative_error = (mean_square_deviation / average) * 100
    result.append(f"Relative error: {relative_error}")

    return result

async def server(websocket, path):
    while True:
        try:
            data = await websocket.recv()
            print(f"Received data: {data}")

            data = json.loads(data)

            result = await calculate_experiment_results(data)

            await websocket.send(json.dumps(result, ensure_ascii=False).encode('utf-8'))
        except websockets.exceptions.ConnectionClosedError:
            break
print("IP:", end=" ")
ip = input()
print("Port:", end=" ")
port = input()
start_server = websockets.serve(server, ip, port)

asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Результат запуска представлен на рисунке 1, 2

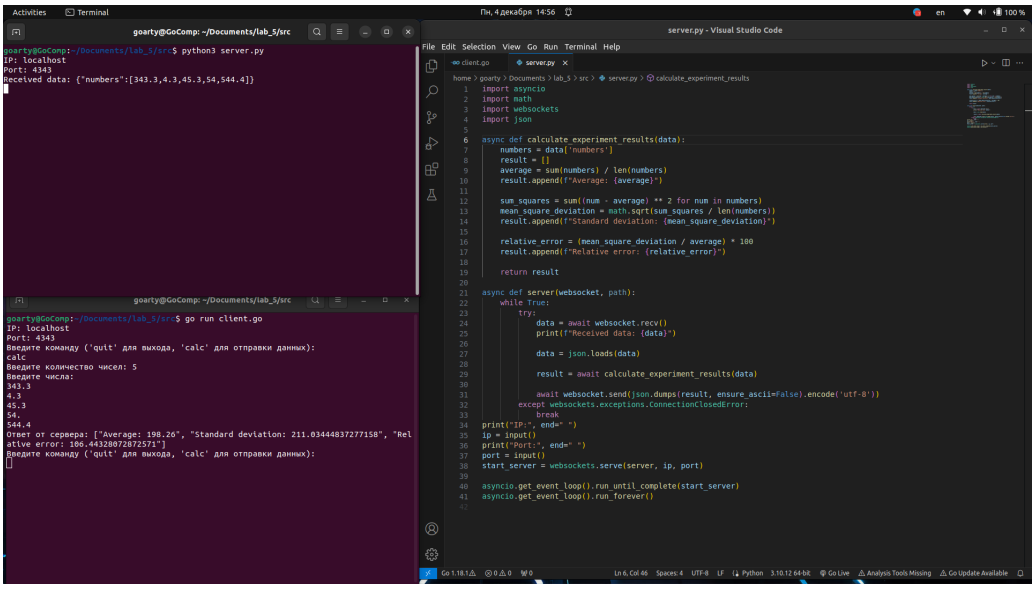

Рис. 1 — Пример на localhost

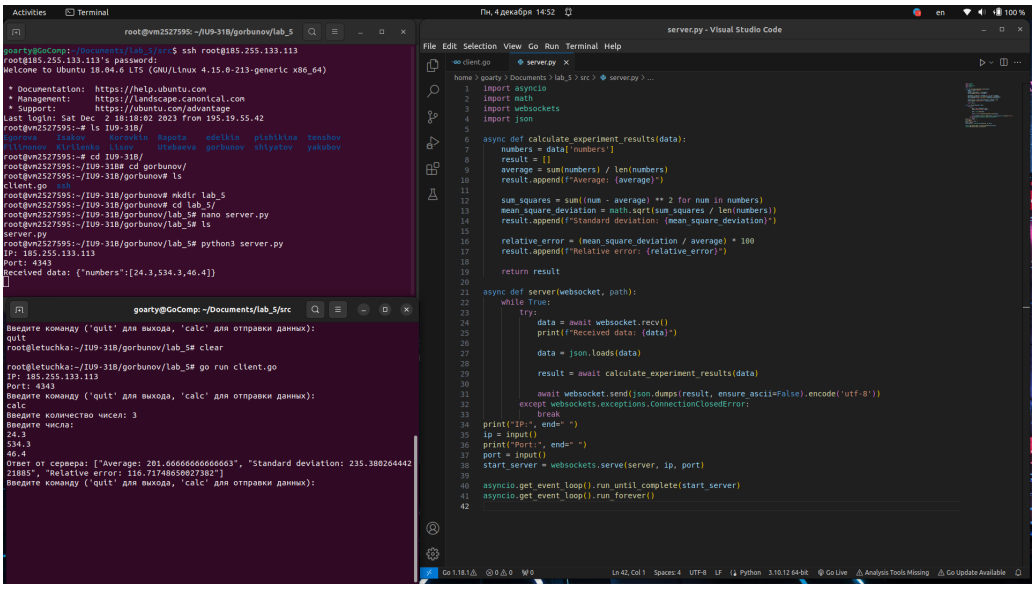

Рис. 2 — Пример на серверах