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 № 1

по курсу «Компьютерные сети»

«Простейший протокол прикладного уровня»

Студент группы ИУ9-32Б Лавров Р. Д.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

Целью работы является знакомство с принципами разработки протоколов прикладного уровня и их реализацией на языке Go.

2 Результаты

Исходный код простейший программы приведенный на лекциих 1- 2.

Листинг 1: Пример реализации класса

```
package main

import (
  "encoding/json"
  "fmt"
  "net"
)

type Message struct {
  X1, Y1, X2, Y2 float32
  X3, Y3, X4, Y4 float32
}

type Response struct {
  Intersect  bool
  X, Y       float32
}

func main() {
  var x1, y1, x2, y2, x3, y3, x4, y4 float32
  fmt.Print("Line 1: ")
  fmt.Scanf("%f %f %f %f", &x1, &y1, &x2, &y2)
  fmt.Print("Line 2: ")
  fmt.Scanf("%f %f %f %f", &x3, &y3, &x4, &y4)

  msg := Message{x1, y1, x2, y2, x3, y3, x4, y4}
  msgEncode, err := json.Marshal(msg)

  conn, err := net.Dial("tcp", "185.102.139.161:3000")
  if err != nil {
    fmt.Println("Error:", err)
    return
  }

```

```
    defer conn.Close()
    _, err = conn.Write(msgEncode)
    if err != nil {
        fmt.Println("Error:", err)
        return
    }

    buffer := make([]byte, 1024)
    n, err := conn.Read(buffer)

    var response Response
    json.Unmarshal(buffer[:n], &response)
    if response.Intersect {
        fmt.Printf("(%.2f, %.2f)\n", response.X, response.Y)
    } else {
        fmt.Println("Lines do not intersect")
    }
}
```

Листинг 2: Пример реализации класса

```
package main

import (
    "encoding/json"
    "fmt"
    "net"
)

type Message struct {
    X1, Y1, X2, Y2 float32
    X3, Y3, X4, Y4 float32
}

type Response struct {
    Intersect bool
    X, Y      float32
}

func main() {
    l, err := net.Listen("tcp4", ":3000")
    if err != nil {
        fmt.Println(err)
    }
    defer l.Close()

```

```
26    for {
27      c, err := l.Accept()
28      if err != nil {
29        fmt.Println(err)
30        return
31      }
32      go handleClient(c)
33    }
34  }
35
36  func handleClient(conn net.Conn) {
37    defer conn.Close()
38    buffer := make([]byte, 1024)
39    msg := Message{}
40
41    for {
42      n, _ := conn.Read(buffer)
43      json.Unmarshal(buffer[:n], &msg)
44      fmt.Println("Get data")
45      fmt.Println(msg)
46      intersect, x, y := calculateIntersection(msg.X1, msg.Y1, msg.X2, msg.Y2, msg.X3, msg.Y3,
          msg.X4, msg.Y4)
47      response := Response{intersect, x, y}
48      responseData, _ := json.Marshal(response)
49      conn.Write(responseData)
50      fmt.Println("Sent response")
51      break
52    }
53  }
54
55  func calculateIntersection(x1, y1, x2, y2, x3, y3, x4, y4 float32) (bool, float32, float32) {
56    denom := (y4-y3)*(x2-x1) - (x4-x3)*(y2-y1)
57    if denom == 0 {
58      return false, 0, 0
59    }
60    ua := ((x4-x3)*(y1-y3) - (y4-y3)*(x1-x3)) / denom
61    return true, x1 + ua*(x2-x1), y1 + ua*(y2-y1)
62  }
```

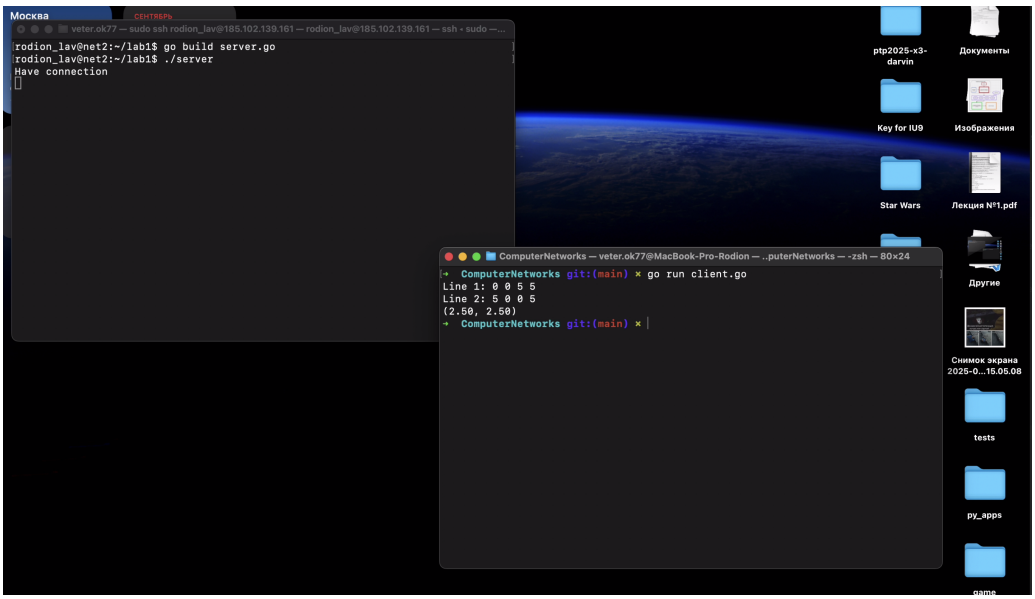

Рис. 1 — Результат работы

3 Вывод

Я научился реализовывать TCP соединение между клиентом и сервером