



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»

**Летучка № 2**  
**по курсу «Компьютерные сети»**  
**«Реализация сервера на UDP и TCP протоколах»**

Студент группы ИУ9-32Б Лавров Р. Д.

Преподаватель Посевин Д. П.

*Москва 2025*

# 1 Задание

Реализовать сервера на UDP и TCP протоколах.

## 2 Результаты

??- ??.

Листинг 1: TCP сервер

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "math/rand/v2"
7     "net"
8     "strings"
9 )
10
11 var NUM string
12 var collect map[int]bool
13
14 func main() {
15     listener, err := net.Listen("tcp", "10.37.196.182:4000")
16     if err != nil {
17         fmt.Println("Ошибка запуска сервера:", err)
18         return
19     }
20     defer listener.Close()
21
22     fmt.Println("Сервер запущен на порту 4000")
23     fmt.Println("Ожидание подключений ... ")
24
25     collect = make(map[int]bool)
26     NUM = guessNumber()
27
28     for {
29         conn, err := listener.Accept()
30         if err != nil {
31             fmt.Println("Ошибка подключения:", err)
32             continue
33         }
34         go handleClient(conn)
35     }
```

```

36 }
37
38 func handleClient(conn net.Conn) {
39     defer conn.Close()
40
41     fmt.Println("Новое подключение")
42     reader := bufio.NewReader(conn)
43
44     for {
45         message, err := reader.ReadString('\n')
46         if err != nil {
47             fmt.Println("Клиент отключился")
48             return
49         }
50         fmt.Printf("Клиент ввёл: %s", message)
51         cow, bull := analyzeMessage(strings.TrimSpace(message), NUM)
52         if bull == 4 {
53             conn.Write([]byte("Поздравляем! Вы угадали число!\n"))
54             NUM = guessNumber()
55         } else {
56             response := fmt.Sprintf("Коровы: %d, Быки: %d\n", cow, bull)
57             conn.Write([]byte(response))
58         }
59     }
60 }
61
62 func analyzeMessage(message string, number string) (int, int) {
63     cow, bull := 0, 0
64     for i := 0; i < 4; i++ {
65         if strings.Contains(message, string(number[i])) {
66             if string(number[i]) == string(message[i]) {
67                 bull++
68             } else {
69                 cow++
70             }
71         }
72     }
73     return cow, bull
74 }
75
76 func guessNumber() string {
77     collect = make(map[int]bool)
78     ans := ""
79     for i := 0; i < 4; i++ {
80         d := rand.IntN(10)
81         for {

```

```

82     if _, ok := collect[d]; !ok {
83         break
84     }
85     d = rand.IntN(10)
86 }
87 ans += string('0' + d)
88 collect[d] = true
89 }
90 fmt.Println(ans)
91 return ans
92 }

```

## Листинг 2: UDP сервер

```

1 package main
2
3 import (
4     "fmt"
5     "math/rand/v2"
6     "net"
7     "strings"
8 )
9
10 var NUM string
11 var collect map[int]bool
12
13 func main() {
14     addr, err := net.ResolveUDPAddr("udp", "10.37.196.182:4000")
15     if err != nil {
16         fmt.Println("Ошибка разрешения адреса:", err)
17         return
18     }
19
20     conn, err := net.ListenUDP("udp", addr)
21     if err != nil {
22         fmt.Println("Ошибка запуска сервера:", err)
23         return
24     }
25     defer conn.Close()
26
27     fmt.Println("UDP сервер запущен на порту 4000")
28     fmt.Println("Ожидание сообщений ... ")
29
30     collect = make(map[int]bool)
31     NUM = guessNumber()
32

```

```

33  buffer := make([]byte, 1024)
34
35  for {
36      n, clientAddr, err := conn.ReadFromUDP(buffer)
37      if err != nil {
38          fmt.Println("Ошибка чтения:", err)
39          continue
40      }
41
42      message := strings.TrimSpace(string(buffer[:n]))
43      fmt.Printf("Получено сообщение от %s: %s\n", clientAddr, message)
44
45      go handleClient(conn, clientAddr, message)
46  }
47 }
48
49 func handleClient(conn *net.UDPConn, clientAddr *net.UDPAddr, message string) {
50     cow, bull := analyzeMessage(message, NUM)
51
52     var response string
53     if bull == 4 {
54         response = "Поздравляем! Вы угадали число!\n"
55         NUM = guessNumber()
56     } else {
57         response = fmt.Sprintf("Коровы: %d, Быки: %d\n", cow, bull)
58     }
59
60     _, err := conn.WriteToUDP([]byte(response), clientAddr)
61     if err != nil {
62         fmt.Println("Ошибка отправки:", err)
63     }
64 }
65
66 func analyzeMessage(message string, number string) (int, int) {
67     cow, bull := 0, 0
68     for i := 0; i < 4; i++ {
69         if strings.Contains(message, string(number[i])) {
70             if string(number[i]) == string(message[i]) {
71                 bull++
72             } else {
73                 cow++
74             }
75         }
76     }
77     return cow, bull
78 }

```

```

79
80 func guessNumber() string {
81     ans := ""
82     collect = make(map[int]bool)
83     for i := 0; i < 4; i++ {
84         d := rand.IntN(10)
85         for {
86             if _, ok := collect[d]; !ok {
87                 break
88             }
89             d = rand.IntN(10)
90         }
91         ans += string('0' + d)
92         collect[d] = true
93     }
94     fmt.Println("Новое число:", ans)
95     return ans
96 }

```

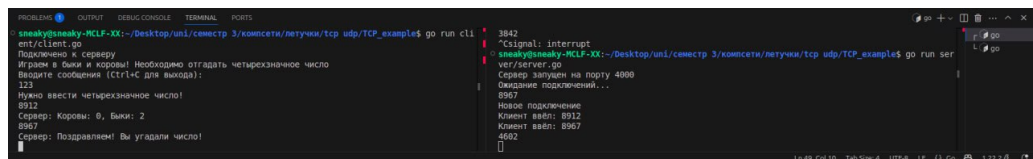


Рис. 1 — Результат работы

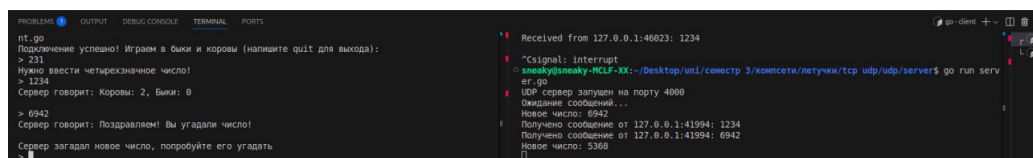


Рис. 2 — Результат работы

### 3 Вывод

В ходе выполнения летучки был реализован сервер для игры "Быки и коровы" на UDP и TCP протоколах. Сервер успешно обрабатывает подключения клиентов, принимает их догадки и возвращает количество коров и быков. При угадывании числа сервер генерирует новое число для продолжения игры.