Exercício 04 - Prog. Concorrente e Distribuída (IF711)

Luís Eduardo Martins Alves
Pedro Nogueira Coutinho
Zenio Angelo

Centro de Informática UFPE
Universidade Federal de Pernambuco

TCP Client - Conexão e envio de mensagem

```
for iteration := 0; iteration < total_iterations; iteration++ {
        conn, err := net.DialTCP(TYPE, nil, tcpServer)
        if err != nil {
                println("Dial failed:", err.Error())
                os.Exit(1)
        }
        num := strconv.Itoa(rand.Intn(80))

        start_time := time.Now()
        _, err = conn.Write([]byte(num))
        if err != nil {
                println("Write data failed:", err.Error())
                os.Exit(1)
        }
```

```
conn, err := net.DialTCP(TYPE, nil, tcpServer)
if err != nil {
        println("Dial failed:", err.Error())
        os.Exit(1)
}
```

```
_, err = conn.Write([]byte(num))
if err != nil {
        println("Write data failed:", err.Error())
        os.Exit(1)
}
```

TCP Server - Aceitação de comunicação e lidando com requisição

```
listen, err := net.Listen(TYPE, HOST+":"+PORT)
if err != nil {
        log.Fatal(err)
        os.Exit(1)
}
defer listen.Close()

println("Servidor pronto para receber mensagens TCP.")
for {
        conn, err := listen.Accept()
        if err != nil {
                log.Fatal(err)
                os.Exit(1)
        }
        go handleRequest(conn)
}
```

TCP Server - Lidando com requisição

```go
func handleRequest(conn net.Conn) {
        // incoming request
        buffer := make([]byte, 1024)
        bytesRead, err := conn.Read(buffer)
        if err != nil {
                log.Fatal(err)
        }
        n, err := strconv.Atoi(string(buffer[:bytesRead]))

        //fmt.Printf("Recebido de %s: %d\n", conn.LocalAddr().String(), n)
        // write data to response
        conn.Write([]byte(strconv.Itoa(fibo(n))))

        // close conn
        conn.Close()
}
```

```go
func fibo(n int) int {
        ans := 1
        prev := 0
        for i := 1; i < n; i++ {
                temp := ans
                ans = ans + prev
                prev = temp
        }
        return ans
}
```

TCP Server - Enviando resposta

```
        //fmt.Printf("Recebido de %s: %d\n", conn.LocalAddr().String(), n)
        // write data to response
        conn.Write([]byte(strconv.Itoa(fibo(n))))

        // close conn
        conn.Close()
}
```

UDP Client - Conexão

```
servAddr, err := net.ResolveUDPAddr(TYPE, HOST+":"+PORT)
if err != nil {
        fmt.Println("Erro ao resolver endereço do servidor:", err)
        os.Exit(1)
}
```

```
conn, err := net.DialUDP("udp", nil, servAddr)
if err != nil {
        fmt.Println("Erro ao conectar ao servidor:", err)
        os.Exit(1)
}
```

UDP Client - Envio de mensagem

```
for iteration := 0; iteration < total_iterations; iteration++ {
        num := strconv.Itoa(rand.Intn(80))
        //fmt.Printf("Fibo for %s\n", num)
        start_time := time.Now()
        _, err = conn.Write([]byte(num))
        if err != nil {
                fmt.Println("Erro ao enviar mensagem:", err)
                os.Exit(1)
        }
```

UDP Server - Aceitação de comunicação

```
servAddr, err := net.ResolveUDPAddr(TYPE, HOST+":"+PORT)
if err != nil {
        fmt.Println("Erro ao resolver endereço do servidor:", err)
        os.Exit(1)
}
```

```
conn, err := net.ListenUDP(TYPE, servAddr)
if err != nil {
        fmt.Println("Erro ao ouvir:", err)
        os.Exit(1)
}
defer conn.Close()
fmt.Println("Servidor pronto para receber mensagens UDP.")
```

UDP Server - Lidando com requisição

```
for {
        buffer := make([]byte, 1024)
        n, addr, err := conn.ReadFromUDP(buffer)
        if err != nil {
                fmt.Println("Erro ao receber dados:", err)
                os.Exit(1)
        }
        go handleRequest(*conn, n, addr, buffer)
}
```

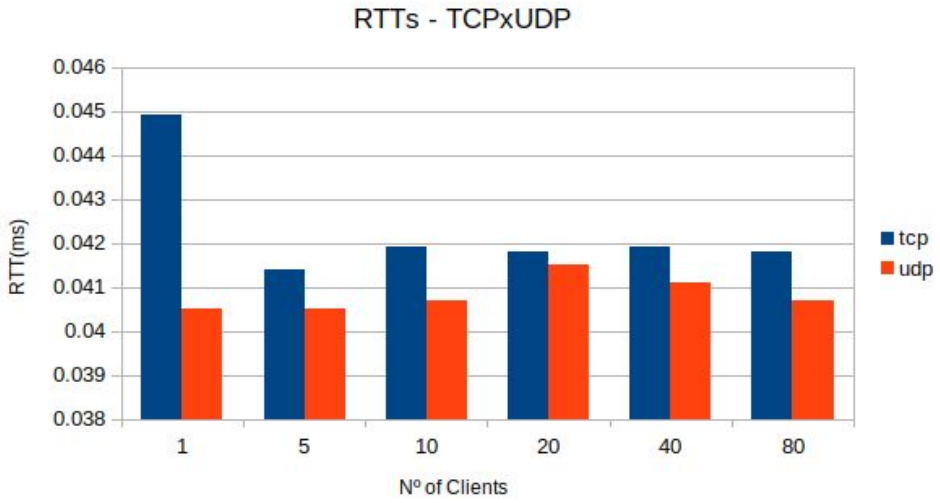
```
func handleRequest(conn net.UDPConn, n int, addr *net.UDPAddr, buffer []byte) {
        num, err := strconv.Atoi(string(buffer[:n]))
        //fmt.Printf("Recebido de %s: %d\n", addr.String(), num)

        // Enviar uma resposta ao cliente
        response := []byte(strconv.Itoa(fibo(num)))
        _, err = conn.WriteToUDP(response, addr)
        if err != nil {
                fmt.Println("Erro ao enviar resposta:", err)
                os.Exit(1)
        }
}
```

```
func fibo(n int) int {
        ans := 1
        prev := 0
        for i := 1; i < n; i++ {
                temp := ans
                ans = ans + prev
                prev = temp
        }
        return ans
}
```

UDP Server - Enviando resposta

```
// Enviar uma resposta ao cliente
response := []byte(strconv.Itoa(fibo(num)))
_, err = conn.WriteToUDP(response, addr)
if err != nil {
        fmt.Println("Erro ao enviar resposta:", err)
        os.Exit(1)
}
```

Analisando resultados e diferenças



nº de clientes	1	5	10	20	40	80
RTT - TCP(ms)	0.0449	0.0414	0.0419	0.0418	0.0419	0.0418
RTT – UDP(ms)	0.0405	0.0405	0.0407	0.0415	0.0411	0.0407