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 №3

Пиринговые сети

Вариант 29

Выполнил
студент группы ИУ9-31Б
Лисов Алексей

Москва, 2023

1 Условие

Целью данной работы является создание пиринговой сети на языке Go. Условие задачи, исходный код и пример работы программы необходимо предоставить в формате LaTeX.

2 Код решения

Файл main.go

```
package main

import (
        "encoding/json"
        "flag"
        "fmt"
        "io/ioutil"
        "iu9-networks/lab3/models"
        "iu9-networks/lab3/pkg/peer"
        "os"
)

func getDataForStart(nodeName string) (peer.Peer, error) {
        file, err := os.Open("../a.json")
        if err != nil {
                fmt.Println(err)
                return peer.Peer{}, err
        }
        defer file.Close()

        data, err := ioutil.ReadAll(file)
        if err != nil {
                fmt.Println(err)
                return peer.Peer{}, err
        }

        var nodes map[string]models.Node
        err = json.Unmarshal(data, &nodes)
        if err != nil {
                fmt.Println(err)
                return peer.Peer{}, err
```

```
        }

        neighbours := make(map[string]models.Node)

        for name, node := range nodes {
                if name != nodeName {
                        neighbours[name] = node
                }
        }

        return peer.Peer{
                Name:      nodeName,
                Info:      nodes[nodeName],
                Neighbors: neighbours,
        }, nil
}

func main() {

        name := flag.String("name", "", "Node name")

        flag.Parse()

        peer, err := getDataForStart(*name)
        if err != nil {
                fmt.Println(err)
                return
        }

        fmt.Println(peer)

        go peer.StartWebServer()
        go peer.StartSocket()

        // Keep the main goroutine alive
        select {}
}
```

Файл peer.go

```
package peer
```

```
import (
        "encoding/json"
        "fmt"
        "github.com/gorilla/websocket"
        "html/template"
        "iu9-networks/lab3/models"
        "log"
        "net"
        "net/http"
)

var upgrader = websocket.Upgrader{
        ReadBufferSize:  1024,
        WriteBufferSize: 1024,
}

type Peer struct {
        Name      string
        Info      models.Node
        Neighbors map[string]models.Node
        conn      *websocket.Conn
}

var tpl *template.Template

// StartWebServer - метод "поднятия" веб сервера, отвечающего за интерфейс
func (p *Peer) StartWebServer() {
        tpl, _ = template.ParseFiles("../index.html")

        http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
                tpl.Execute(w, struct{ NodeName string }{NodeName: p.Name})
        })

        http.HandleFunc("/ws/"+p.Name, func(w http.ResponseWriter, r *http.Request)
                conn, err := upgrader.Upgrade(w, r, nil)
                if err != nil {
                        log.Println(err)
                        return
                }
```

```
                p.conn = conn

                for {
                        // Чтение сообщений из веб-сокета
                        _, bytes, err := conn.ReadMessage()
                        if err != nil {
                                fmt.Println("Ошибка чтения сообщения из веб-сокета
                                return
                        }

                        msg := models.Line{}
                        err = json.Unmarshal(bytes, &msg)
                        if err != nil {
                                fmt.Println("Ошибка десериализации JSON:", err)
                                continue
                        }

                        p.drawLine(msg)
                }
        })

        fmt.Println("Starting web server_2 on port ", p.Info.HtmlServerPort)
        http.ListenAndServe(":"+p.Info.HtmlServerPort, nil)
}

// drawLine - функция "отрисовки" линии
// передает информацию о нарисовании линии всем соседям
func (p *Peer) drawLine(msg models.Line) {
        // Отправка полученного сообщения всем узлам
        p.SendMessage(msg)
        // и себе не забыть нарисовать
        p.conn.WriteJSON(msg)
}

// StartSocket - стартуем сокет
func (p *Peer) StartSocket() {
        ln, _ := net.Listen("tcp", ":"+p.Info.Port)
        defer ln.Close()
```

```
        for {
                conn, _ := ln.Accept()
                go p.handleConnection(conn)
        }
}

// handleConnection - обработка входящих соединений
func (p *Peer) handleConnection(conn net.Conn) {
        defer conn.Close()

        buf := make([]byte, 1024)
        n, _ := conn.Read(buf)

        var message models.Line
        json.Unmarshal(buf[:n], &message)

        // теперь просто по вебсокету отправляем

        if p.conn != nil {
                err := p.conn.WriteJSON(message)
                if err != nil {
                        fmt.Println(err)
                        return
                }
        }

        fmt.Printf("Received message: %+v\n", message)
}

// SendMessage - посылка сообщения всем соседям
func (p *Peer) SendMessage(message models.Line) {
        for _, neighbor := range p.Neighbors {
                conn, err := net.Dial("tcp", ":"+neighbor.Port)
                if err != nil {
                        fmt.Println(err)
                        return
                }

                defer conn.Close()
```

```
                jsonMessage, _ := json.Marshal(message)
                _, err = conn.Write(jsonMessage)
                if err != nil {
                        fmt.Println(err)
                        return
                }
        }
}
```

3 Скриншоты

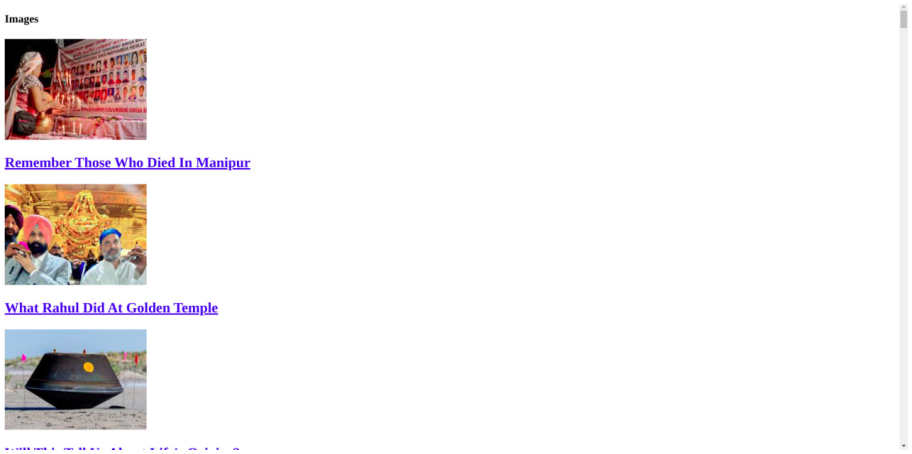
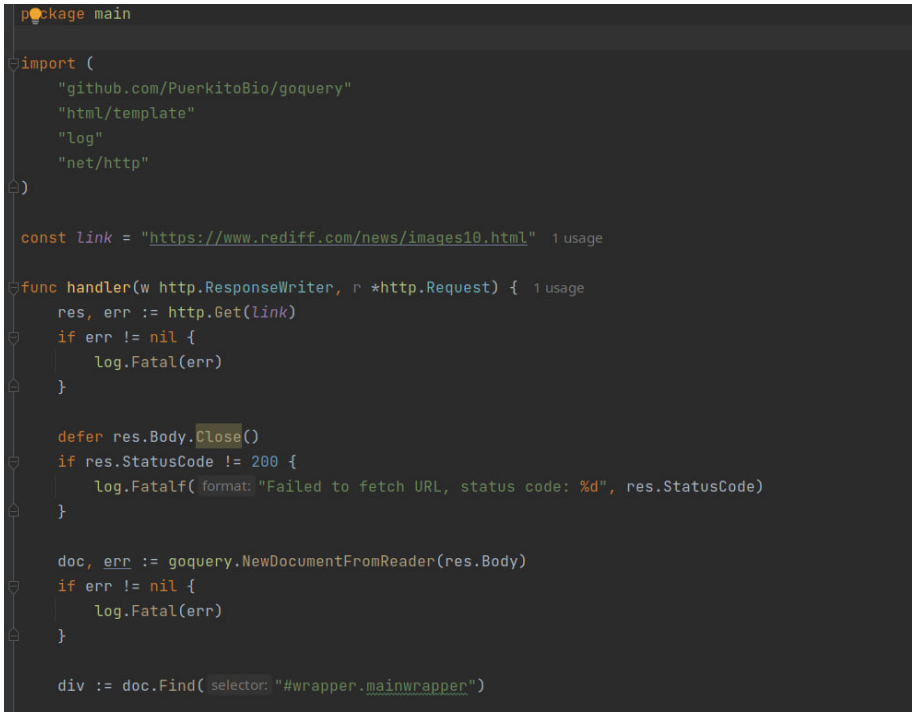


Рис. 1: Результат



```go
package main

import (
    "github.com/PuerkitoBio/goquery"
    "html/template"
    "log"
    "net/http"
)

const link = "https://www.rediff.com/news/images10.html"  1 usage

func handler(w http.ResponseWriter, r *http.Request) {  1 usage
    res, err := http.Get(link)
    if err != nil {
        log.Fatal(err)
    }

    defer res.Body.Close()
    if res.StatusCode != 200 {
        log.Fatalf( format: "Failed to fetch URL, status code: %d", res.StatusCode)
    }

    doc, err := goquery.NewDocumentFromReader(res.Body)
    if err != nil {
        log.Fatal(err)
    }

    div := doc.Find( selector: "#wrapper.mainwrapper")
```

Рис. 2: Код


    </head>
    <body>
        {{.}}
    </body>
    </html>
    `
    pageTmpl, err := template.New(name: "page").Parse(tmpl)
    if err != nil {
        log.Fatal(err)
    }
    w.Header().Set(key: "Content-Type", value: "text/html; charset=utf-8")
    err = pageTmpl.Execute(w, template.HTML(html))
    if err != nil {
        log.Fatal(err)
    }
}
func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(addr: ":8080", handler: nil)
}

Рис. 3: Код