Amrita School of Engineering

Department of Computer Science and Engineering

19CSE312: Distributed Systems

Lab-Evaluation-2

Date: 30/03/2022

RAVELLA ABHINAV

CB.EN.U4CSE19453

1. Write a Go RPC client-server system in which the server maintains information/ metadata about the research papers (Paper title, Author, Journal, volume and year)
   The server supports two different types of searches on the system based on
   (a) paper title and
   (b) author name and year.
   The client will request for paper details based upon the above two methods and the server will send the information back to the client. If data is not found you must print "Error".

   Code:
   Server:

```go
package main

import (
    "log"
    "net"
    "net/http"
    "net/rpc"
    "time"
)

type Args struct{}

type TimeServer int64

func main() {
    timeserver := new(TimeServer)

    rpc.Register(timeserver)
```

```
    rpc.HandleHTTP()
    listener, err := net.Listen("tcp", ":1234")
    if err != nil {
        log.Fatal("Listener error: ", err)
    }
    http.Serve(listener, nil)
}

func (t *TimeServer) GiveServerTime(args *Args, reply *int64) error {

    *reply = time.Now().Unix()
    return nil
}
```

Client:

```
package main

import (
    "fmt"
    "net/rpc"
)

type Paper struct {
    Title   string
    Author  string
    Journal string
    Volume  int
    Year    int
}

type PaperList struct {
    List []Paper
}

func main() {

    client, err := rpc.DialHTTP("tcp", "localhost:1234")
    if err != nil {
        fmt.Println("Error in connecting to the server")
        return
    }


    var p1 = Paper{"The Go Programming Language", "Alan A. A. Donovan",
"Journal of Go", 1, 2009}
```

```
    var p2 = Paper{"Go Concurrency Patterns", "Brian W. Kernighan", "Go
Concurrency Cookbook", 2, 2012}
    var p3 = Paper{"Go in Action", "Brian W. Kernighan", "Go in Action",
3, 2016}

    var pl = PaperList{make([]Paper, 3)}
    pl.List[0] = p1
    pl.List[1] = p2
    pl.List[2] = p3

    var reply bool
    err = client.Call("PaperServer.Add", pl, &reply)
    if err != nil {
        fmt.Println("Error in calling the server")
        return
    }
    fmt.Println("Server reply: ", reply)
}
```

Output:

```
PS C:\Users\Administrator\Documents\SEM-6\19CSE312 - Distributed Systems\Lab\go> go run .\q1client.go
Error in calling the server
PS C:\Users\Administrator\Documents\SEM-6\19CSE312 - Distributed Systems\Lab\go>
```

2. Write a program that makes use of go routines. This program takes as input a text message and the first go routine encodes this message based on any of the coding strategy of your choice. Once encoded completely, it then sends this encoded message to the other routine that computes the hash value by simply computing the given formula (no. of alphabets in message * number of vowels in the encoded message). The main routine finally prints the combination of the two condings as the final output in the following manner: ENCODING1_HASH.

.

Code:

```
package main

import (
    "fmt"
    "time"
)

func main() {
```

```
    var msg string
    fmt.Print("Enter a message = ")
    fmt.Scanln(&msg)

    go encode(msg)
    go hash(msg)

    time.Sleep(time.Second * 1)
    fmt.Println("Encoded message = ", encode(msg))
    fmt.Println("Hash value = ", hash(msg))
    fmt.Println("Encoding1_Hash = ", msg+"_"+msg)
}

func encode(msg string) string {
    str := ""
    for i := 0; i < len(msg); i++ {
        if msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] ==
'o' || msg[i] == 'u' {
            str = str + "*"
        } else {
            str = str + string(msg[i])
        }
    }
    return str
}

func hash(msg string) int {
    var count int
    var vowel int
    for i := 0; i < len(msg); i++ {
        if msg[i] == 'a' || msg[i] == 'e' || msg[i] == 'i' || msg[i] ==
'o' || msg[i] == 'u' {
            vowel++
        }
    }
    count = len(msg) * vowel
    return count
}
```

Output:

```
PS C:\Users\Administrator\Documents\SEM-6\19CSE312 - Distributed Systems\Lab\go> go run .\q2.go
Enter a message = Amrita
Encoded message =  Amr*t*
Hash value =  12
Encoding1_Hash =  Amrita_Amrita
```

```
PS C:\Users\Administrator\Documents\SEM-6\19CSE312 - Distributed Systems\Lab\go> go run .\q2.go
Enter a message = Distributed Systems
Encoded message =  D*str*b*t*d
Hash value =  44
Encoding1_Hash =  Distributed_Distributed
PS C:\Users\Administrator\Documents\SEM-6\19CSE312 - Distributed Systems\Lab\go>
```