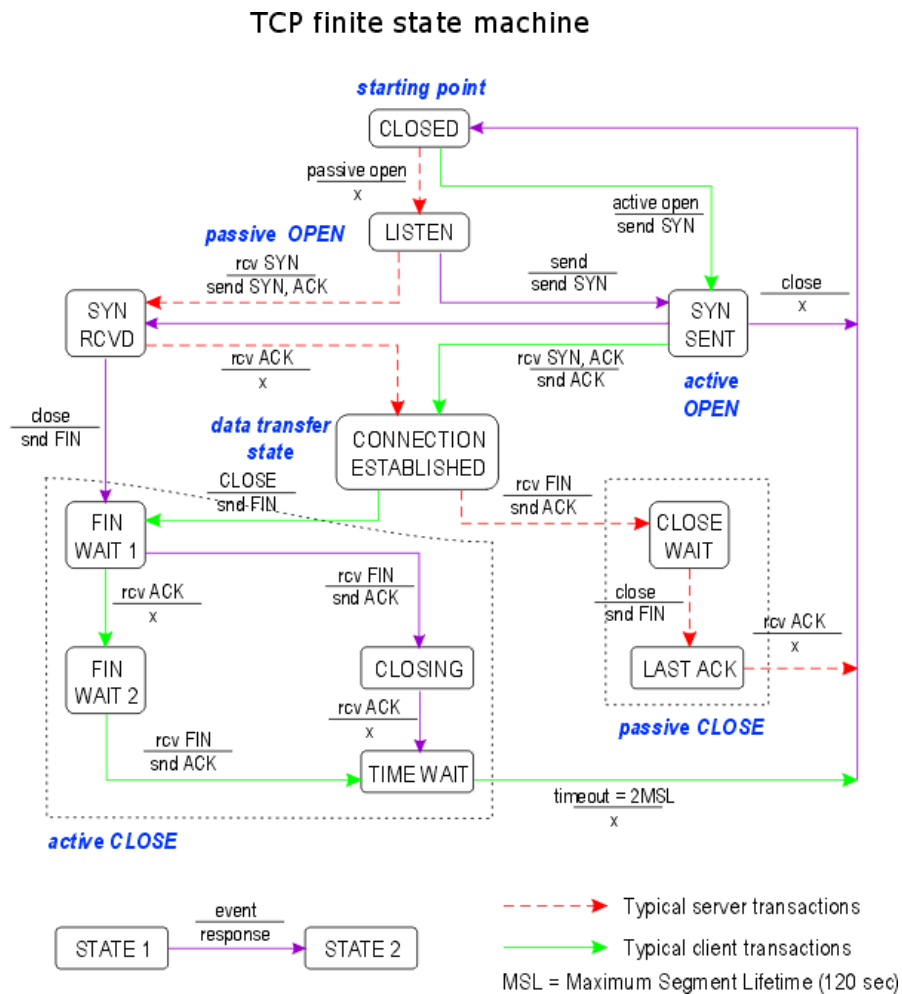


Tugas 1 Pemrograman Jaringan

Meilyand Evriyan Timor 1301161769

Soal No 1



Jelaskan maksud diagram finite state machine dari TCP Connection diatas!

Jawaban:

Diagram di atas merupakan proses dari TCP three-way handshake, di mana dimulai dari belum terdapat koneksi hingga melewati beberapa states sampai koneksi dibuat. Dengan menggunakan tiga jenis pesan yang sebagai penentu transisi antar state, yaitu:

1. SYN: merupakan *synchronize message*, digunakan untuk memulai dan membuat koneksi
2. ACK: merupakan *acknowledgment message*, sebagai penanda bahwa menerima pesan SYN atau FIN
3. FIN: merupakan *finish message*, sebagai penanda bahwa server ingin memutuskan koneksi

Penjelasan state pada lembar selanjutnya

Penjelasan setiap state sebagai berikut:

State	Penjelasan State	Event dan Transisi
CLOSED	Awal mula state, menandakan belum terdapat koneksi	Passive Open: server memulai proses koneksi dengan membuka TCP port lalu transisi ke state LISTEN
		Active Open, Send SYN: client memulai proses koneksi dengan mengirim pesan SYN ke server lalu transisi ke state SYN-SENT
LISTEN	Server menunggu pesan SYN dari client	Receive SYN, Send SYN ACK: server menerima pesan SYN dari client dan merespon dengan mengirim pesan SYN ACK ke client lalu transisi ke state SYN-RECEIVED
SYN-SENT	Client telah mengirim pesan SYN ke server dan menunggu respon dari server	Receive SYN, Send ACK: jika client menerima SYN dari server lain dan bukan ACK untuk SYN yg awal, SYN tersebut diakui lalu dan mengirim ACK-nya lalu melakukan transisi ke SYN-RECEIVED
		Receive SYN ACK, Send ACK: jika client menerima respon ACK dari SYN-nya (berupa SYN ACK), menandakan koneksi diterima lalu transisi ke state CONNECTION ESTABLISHED
SYN-RECEIVED	Server sudah merespon SYN dari client dan menunggu respon ACK dari client juga	Receive ACK: ketika server menerima ACK dari SYN client, maka lanjut transisi ke CONNECTION ESTABLISHED
CONNECTION ESTABLISHED	Koneksi telah dibentuk, pertukaran data dilakukan sampai koneksi ditutup	Close, Send FIN: client dapat menutup koneksi dengan mengirimkan pesan FIN lalu transisi ke FIN-WAIT 1
		Receive FIN, Send ACK: server menerima pesan FIN dari client dan melakukan konfirmasi dengan membalas pesan ACK ke client
CLOSE WAIT	Server telah menerima permintaan FIN (close connection) dari client dan	Close, Send FIN: aplikasi yg menggunakan TCP, setelah menerima informasi bahwa

	menunggu ACK dari client	koneksi ingin dimatikan, mengirimkan permintaan close connection lalu mengirimkan pesan FIN ke client, kemudian transisi ke LAST ACK
LAST ACK	Server sudah menerima request close connection dan mengkonfirmasinya, telah mengirimkan pesan FIN ke client dan menunggu ACK-nya	Receive ACK: Server menerima respon ACK dari client lalu transisi ke state CLOSED
FIN-WAIT 1	Client menunggu ACK dari FIN yang dikirimkan sebelumnya	Receive ACK: client menerima ACK dari FIN sebelumnya lalu transisi ke FIN-WAIT 2 Receive FIN, Send ACK: client menerima FIN dari server lain. Lalu mengirim ACK ke server tsb., kemudian lanjut ke state CLOSING
FIN-WAIT 2	Client menerima ACK dari FIN sebelumnya dan menunggu FIN dari server lain	Receive FIN, Send ACK: client menerima pesan FIN dari server lain dan mengirim ACK lalu transisi ke state TIME-WAIT
CLOSING	Client menunggu ACK dari FIN server lain pada state FIN-WAIT 1	Receive ACK: client menerima ACK dari FIN server lain lalu transisi ke state TIME-WAIT
TIME WAIT	Menunggu untuk koneksi ditutup	Timeout: Setelah 2 kali waktu MSL, maka transisi ke state CLOSED

Soal No 2 (for dan if/else)

```
package main
import "fmt"
func main() {
    i := 1
    for i <= 3 {
        fmt.Println(i)
        i = i + 1
    }

    for j := 7; j <= 9; j++ {
        fmt.Println(j)
    }

    for {
        fmt.Println("loop")
        break
    }

    for n := 0; n <= 5; n++ {
        if n%2 == 0 {
            continue
        }
        fmt.Println(n)
    }
}
```

```
package main
import "fmt"
func main() {
    if 7%2 == 0 {
        fmt.Println("7 is even")
    } else {
        fmt.Println("7 is odd")
    }

    if 8%4 == 0 {
        fmt.Println("8 is divisible by 4")
    }

    if num := 9; num < 0 {
        fmt.Println(num, "is negative")
    } else if num < 10 {
        fmt.Println(num, "has 1 digit")
    } else {
        fmt.Println(num, "has multiple digits")
    }
}
```

Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

PROGRAM FOR

```
[timor@localhost nomor2]$ go run for.go
1
2
3
7
8
9
loop
1
3
5
```

program for melakukan loop sesuai dengan keadaan yang ditentukan dengan nilai awalan $i = 1$

for pertama mencetak nilai i , selama $i \leq 3$. Didapat hasil 1 2 3

for ke dua mencetak nilai j selama $j \leq 9$, dengan awalan $j = 7$. Didapat hasil 7 8 9

for ke tiga merupakan loop forever dengan mencetak string "loop", namun terdapat break sehingga loop

hanya dilakukan sekali saja. Didapat hasil berupa string "loop"

for ke empat mencetak nilai n dengan awalan $n = 0$, selama $n \leq 5$ dengan kondisi jika $n \bmod 2 = 0$ maka nilai n tidak akan dicetak, melainkan mencetak angka ganjil. Didapat hasil 1 3 5

PROGRAM IF/ELSE

```
[timor@localhost nomor2]$ go run ifElse.go  
7 is odd  
8 is divisible by 4  
9 has 1 digit
```

program if/else melakukan pengecekan dari suatu kondisi jika memenuhi maka melakukan aksi dari kondisi tersebut

kondisi pertama, jika $7 \bmod 2 = 0$, maka mencetak string "7 is even", jika bukan maka mencetak string "7 is odd". Didapat hasil 7 is odd

kondisi ke dua, jika $8 \bmod 2 = 0$, maka mencetak string "8 is divisible by 4", jika bukan maka lanjut ke kondisi ke tiga. Didapat hasil 8 is divisible by 4

kondisi ke tiga memiliki awalan nilai $num = 9$ dengan kondisi, jika $num < 0$ maka cetak "9 is negative", jika $num < 10$ maka cetak "9 has 1 digit", jika tidak keduanya maka cetak "9 has multiple digits". Didapat hasil 9 has 1 digit

Soal No 3 (array dan function)

```
package main
import "fmt"
func main() {
    var a [5]int
    fmt.Println("emp:", a)

    a[4] = 100
    fmt.Println("set:", a)
    fmt.Println("get:", a[4])

    fmt.Println("len:", len(a))

    b := [5]int{1, 2, 3, 4, 5}
    fmt.Println("dcl:", b)

    var twoD [2][3]int
    for i := 0; i < 2; i++ {
        for j := 0; j < 3; j++ {
            twoD[i][j] = i + j
        }
    }
    fmt.Println("2d: ", twoD)
}
```

```
package main
import "fmt"
func plus(a int, b int) int {
    return a + b
}
func plusPlus(a, b, c int) int {
    return a + b + c
}
func main() {
    res := plus(1, 2)
    fmt.Println("1+2 =", res)

    res = plusPlus(1, 2, 3)
    fmt.Println("1+2+3 =", res)
}
```

Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

PROGRAM ARRAY

```
[timor@localhost nomor3]$ go run array.go
emp:  [0 0 0 0 0]
set:  [0 0 0 0 100]
get:  100
len:  5
dcl:  [1 2 3 4 5]
2d:  [[0 1 2] [1 2 3]]
```

program array terdapat variabel integer a[5] sebagai array dengan nilai yg masih kosong lalu dicetak. Didapat hasil emp: [0 0 0 0 0]

Lalu, array pada index ke empat diisi dengan nilai 100 dan dicetak untuk semua array. Didapat hasil set: [0 0 0 0 100]. Kemudian

dicetak kembali untuk index ke empat, didapat hasil get: 100

Kemudian mencetak panjang array dengan sintak len(a), didapat hasil len: 5

Selanjutnya, inisialisasi array kedua dengan semua index array b diisi dengan nilai integer dan dicetak. Didapat hasil: [1,2,3,4,5]

Terakhir dilakukan nested loop untuk membentuk array 2 dimensi dengan nilai array didapat dari penjumlahan i+j didalam nested loop. Didapat hasil [[0 1 2] [1 2 3]]

PROGRAM FUNCTION

```
[timor@localhost nomor3]$ go run function.go  
1+2 = 3  
1+2+3 = 6
```

program function memanggil function plus dan plusPlus pada fungsi main.

Function plus mengembalikan nilai a+b, sedangkan function plusPlus mengembalikan nilai a+b+c. Lalu, pada function main, memanggil function plus(1,2) yg disimpan pada variabel res dan dicetak, didapat hasil 1+2 = 3.

Kemudian nilai variabel res diupdate dengan memanggil function plusPlus(1,2,3) dan dicetak, didapat hasil 1+2+3 = 6.

Soal No 4 (struct dan method)

```
package main

import "fmt"

type person struct {
    name string
    age  int
}

func main() {

    fmt.Println(person{"Bob", 20})

    fmt.Println(person{name: "Alice", age: 30})

    fmt.Println(person{name: "Fred"})

    fmt.Println(&person{name: "Ann", age: 40})

    s := person{name: "Sean", age: 50}
    fmt.Println(s.name)

    sp := &s
    fmt.Println(sp.age)

    sp.age = 51
    fmt.Println(sp.age)
}
```

```
package main

import "fmt"

type rect struct {
    width, height int
}

func (r *rect) area() int {
    return r.width * r.height
}

func (r rect) perim() int {
    return 2*r.width + 2*r.height
}

func main() {
    r := rect{width: 10, height: 5}

    fmt.Println("area: ", r.area())
    fmt.Println("perim:", r.perim())

    rp := &r
    fmt.Println("area: ", rp.area())
    fmt.Println("perim:", rp.perim())
}
```

Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

PROGRAM STRUCT

```
[timor@192 nomor4]$ go run struct.go
{Bob 20}
{Alice 30}
{Fred 0}
&{Ann 40}
Sean
50
51
```

program struct membuat tipe data buatan, yaitu person dengan atribut string name dan integer age.

Lalu dicetak dengan langsung memasukkan nilai ke parameternya yaitu dengan format: person({"Bob",20}). Didapat hasil {Bob 20}

Atau bisa juga dengan format: person({name: "Alice", age: 30}). Didapat hasil {Alice 30}.

Jika yang diberi nilai hanya salah satu atribut, maka yang lain didefault nol untuk integer dan string kosong untuk string. Didapat hasil {Fred 0}

Jika type person menjadi nilai dari suatu variabel, contoh s:= person{name: "Sean", age: 50}, maka untuk memanggil salah satu atribut harus dengan format s.name atau s.age. Didapat hasil Sean

jika ingin mengisi nilai dengan variabel s harus dengan format sp:= &s (memakai & sebagai pointer). Lalu dapat memanggil atribut person. Jika dilakukan sp.age = 51, maka nilai age akan diupdate menjadi 51 yang semulanya 50.

PROGRAM METHOD

```
[timor@192 nomor4]$ go run method.go
area: 50
perim: 30
area: 50
perim: 30
```

program method dapat mengakses ke atribut struct dengan cara deklarasi variabel objek func dan nama fungsi, seperti: `func (r *rect) area() int {`
`return r.width * r.height`

```
func (r rect) perim() int {
    return 2*r.width + 2*r.height
}
```

untuk memanggil fungsi dengan format, misal: `r.area()` dengan `r:=rect{width: 10, height: 5}`.

jika ingin mengisi `r` pada variabel lain misal `sp`, maka formatnya adalah `rp := &r` (menggunakan `&` sbg pointer), lalu memanggil fungsi dengan `rp.area()`

Soal No 5 (multiple return value dan command line)

```
package main

import "fmt"

func vals() (int, int) {
    return 3, 7
}

func main() {

    a, b := vals()
    fmt.Println(a)
    fmt.Println(b)

    _, c := vals()
    fmt.Println(c)

}
```

```
package main

import "flag"
import "fmt"

func main() {

    wordPtr := flag.String("word", "foo", "a string")

    numbPtr := flag.Int("numb", 42, "an int")
    boolPtr := flag.Bool("fork", false, "a bool")

    var svar string
    flag.StringVar(&svar, "svar", "bar", "a string var")

    flag.Parse()

    fmt.Println("word:", *wordPtr)
    fmt.Println("numb:", *numbPtr)
    fmt.Println("fork:", *boolPtr)
    fmt.Println("svar:", svar)
    fmt.Println("tail:", flag.Args())

}
```

Jalankan masing-masing program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

PROGRAM MULTIPLE RETURN

```
[timor@192 nomor5]$ go run multipleReturn.go
3
7
7
```

program ini dapat mengembalikan nilai return sebanyak maksimal 2 kali sesuai parameter fungsi vals.

Terlihat jika a,b := vals(), maka a akan bernilai 3 dan b bernilai 7. Sedangkan yang terakhir _, c := vals, maka c akan bernilai 7 karena 3 telah diisi oleh _.

PROGRAM COMMAND LINE

```
[timor@192 nomor5]$ go run commandLine.go
word: foo
numb: 42
fork: false
svar: bar
tail: []
```

program ini seperti melakukan pengecekan kondisi. Jika kita eksekusi command "go run commandLine.go" maka word, numb, fork, svar, tail akan bernilai default seperti gambar di sebelah kiri. Namun, hasilnya akan berbeda jika dengan command line yg berberda, yaitu seperti gambar di bawah

```
[timor@192 nomor5]$ go run commandLine.go -word=abc -numb=99 -fork=true -svar=siap sekali
word: abc
numb: 99
fork: true
svar: siap
tail: [sekali]
```

Soal No 6 (simple web application)

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
    })

    http.ListenAndServe(":80", nil)
}
```

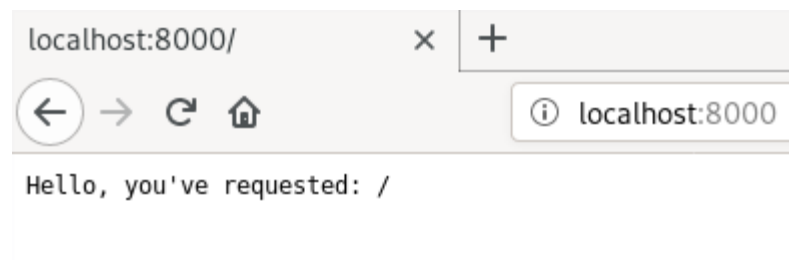
Sebelum menjalankan program diatas, gantilah port 80 ke port 8000. Buka browser kemudian ketikkan alamat localhost:8000.

Jalankan program diatas, apakah outputnya (berikan printscreen) dan jelaskan cara kerjanya!

Jawaban:

PROGRAM SIMPLE WEB APPLICATION

```
[timor@192 nomor6]$ go run simpleWebApp.go
```



program ini bertindak sebagai server untuk melayani request dari client dengan mengakses localhost:8000 pada browser. Port yang digunakan 8000 dan jika berhasil maka server akan menampilkan pesan "Hello, you've requested"

Soal No 7 (create config file)

Buatlah sebuah config file untuk aplikasi web application pada soal no 6 dengan menggunakan library berikut: <https://github.com/spf13/viper>!

Jelaskan susunan directory dari program serta bagaimana cara untuk melakukan konfigurasi file config yang telah anda buat!

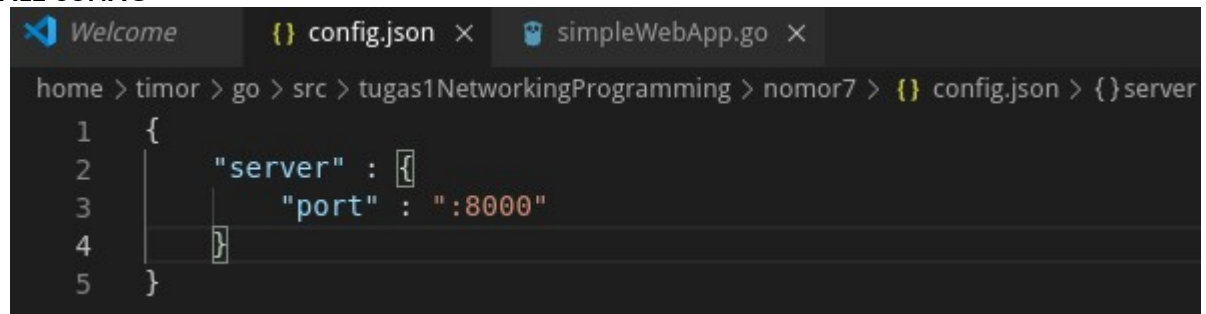
Printscreen hasil dan penjelasan kode untuk membuat file config disini!

Jawaban:

PROGRAM CONFIG FILE

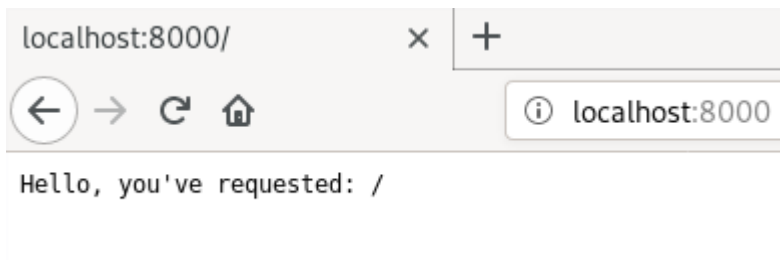
Pertama jalankan perintah `go get -u github.com/spf13/viper` pada command line. Lalu bikin file `config.json` seperti di bawah dengan port 8000. Kemudian bikin code `simpleWebApp.go` dan jalankan, lanjut ke browser dengan `localhost:8000`.

FILE CONFIG



```
home > timor > go > src > tugas1NetworkingProgramming > nomor7 > {} config.json > {} server
1  {
2      "server" : {
3          "port" : ":8000"
4      }
5  }
```

HASIL RUNNING PADA BROWSER



localhost:8000/

⏪ ⏩ ↺ 🏠 ⓘ localhost:8000

Hello, you've requested: /

FILE simpleWebApp.go

```
Welcome  {} config.json  simpleWebApp.go x
home > timor > go > src > tugas1NetworkingProgramming > nomor7 > simpleWebApp.go > {}main
1  package main
2
3  import (
4      "fmt"
5      "net/http"
6
7      "github.com/spf13/viper"
8  )
9
10 func main() {
11     viper.SetConfigType("json")
12     viper.AddConfigPath(".")
13     viper.SetConfigName("config")
14     viper.ReadInConfig()
15
16     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
17         fmt.Fprintf(w, "Hello, you've requested: %s\n", r.URL.Path)
18     })
19
20     http.ListenAndServe(viper.GetString("server.port"), nil)
21 }
22
```

Fungsi `viper.SetConfigType()` digunakan untuk set jenis file konfigurasi.

Fungsi `.AddConfigPath()` digunakan untuk mendaftarkan path folder dimana file-file konfigurasi berada. Fungsi ini bisa dipanggil beberapa kali, jika memang ada banyak file konfigurasi tersimpan dalam path berbeda.

Statement `.SetConfigName()` dieksekusi dengan parameter berisi nama file konfigurasi secara eksplisit tanpa ekstensi. Misalkan nama file adalah `app.config.json`, maka parameter cukup ditulis `app.config`.

Fungsi `.ReadInConfig()` digunakan untuk memproses file-file konfigurasi sesuai dengan path dan nama yang sudah ditentukan.

