1st Lab - CES 27 - Distributed Programming

August 20th, 2019

Prof. Celso Massaki HIRATA, hirata@ita.br

Prof. JULIANA de Melo Bezerra, juliana@ita.br

Igor Bragaia[1], igor.bragaia@gmail.com

[1]Computer Engineering undergraduate student at Aeronautics Institute of Technology (ITA)

This project aims to develop logical clocks through Lampert algorithm using both scalar and vector approach. It is the first assignment of Distributed Programming 2019.2 ITA's class.

1st TASK - SCALAR LOGIC CLOCK

```go
package main

import (
  "bufio"
  "fmt"
  "net"
  "os"
  "strconv"
  "time"
)

var myPort string
var nServers int
var CliConn []*net.UDPConn
var ServConn *net.UDPConn
var logicalClock int

func CheckError(err error) {
  if err != nil {
     fmt.Println("Error: ", err)
     os.Exit(0)
  }
}

func PrintError(err error) {
  if err != nil {
     fmt.Println("Error: ", err)
  }
}

func doServerJob() {
  buf := make([]byte, 1024)
  n, addr, err := ServConn.ReadFromUDP(buf)
  logicalClock_msg, err := strconv.Atoi(string(buf[0:n]))

  if logicalClock < logicalClock_msg {
     logicalClock = logicalClock_msg
  }
  logicalClock++

  fmt.Printf("Received %s from %s\nCurrent Logical Clock = %d\n", string(buf[0:n]), addr, logicalClock)

  if err != nil {
     fmt.Println("Error: ",err)
  }
}
```

```
func doClientJob(otherProcess int, i int) {
  Conn := CliConn[otherProcess]
  msg := strconv.Itoa(i)
  i++
  buf := []byte(msg)
  _,err := Conn.Write(buf)
  if err != nil {
     fmt.Println(msg, err)
  }
}

func initConnections() {
  myPortId, err := strconv.Atoi(os.Args[1])
  myPort = os.Args[myPortId+1]
  nServers = len(os.Args) - 2

   ServerAddr, err := net.ResolveUDPAddr("udp",myPort)
   CheckError(err)
   Conn, err := net.ListenUDP("udp", ServerAddr)
  CheckError(err)

  ServConn = Conn

  for i := 0; i < nServers; i++ {
     ServerAddr,err := net.ResolveUDPAddr("udp","127.0.0.1" + os.Args[i+2])
     CheckError(err)
     LocalAddr, err := net.ResolveUDPAddr("udp", "127.0.0.1:0")
     CheckError(err)
     Conn, err := net.DialUDP("udp", LocalAddr, ServerAddr)
     CheckError(err)

     CliConn = append(CliConn, Conn)
  }
}

func readInput(ch chan string) {
  reader := bufio.NewReader(os.Stdin)
  for {
     text, _, _ := reader.ReadLine()
     ch <- string(text)
  }
}

func main() {
  initConnections()
  defer ServConn.Close()
  for i := 0; i < nServers; i++ {
     defer CliConn[i].Close()
  }

  ch := make(chan string)
  go readInput(ch)
  logicalClock = 0

  for {
     go doServerJob()
     select {
        case x, valid := <-ch:
           if valid {
              i1, err := strconv.Atoi(x)
              if err == nil && i1 < len(os.Args) - 1 {
                 fmt.Printf("Notify port %s\n", os.Args[i1+1])
                 go doClientJob(i1-1, logicalClock)
              } else {
                 fmt.Println("Invalid number")
              }
           } else {
              fmt.Println("Channel closed!")
           }
```

```
        default:
           time.Sleep(time.Second * 1)
     }
  }
}
```

Code 1: process.go file

Using Code 1, we ran different processes into different terminals in order to test the scalar logic clock.

```
$ go run process.go 1 :10003 :10004 :10005
```

Code 2: first terminal

```
$ go run process.go 2 :10003 :10004 :10005
```

Code 3: second terminal

```
$ go run process.go 3 :10003 :10004 :10005
```

Code 4: third terminal

We aim to get the following illustrative output

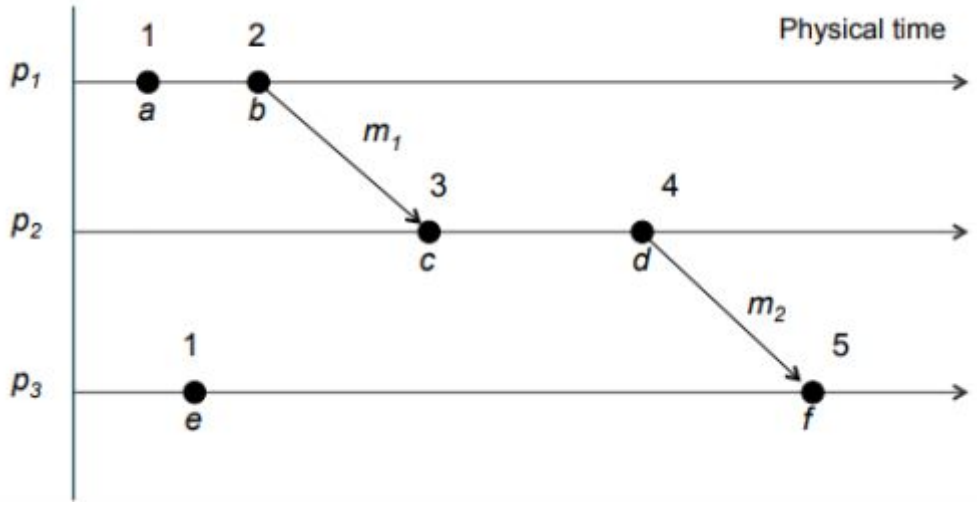

Image 1: wanted output from each process at first, second and third terminals

Note that the below output is correct according to the image above

first terminal :10003	second terminal :10004	third terminal :10005
<< 1 >> Notify port :10003 >> Current Logical Clock = 1		
		<< 3 >> Notify port :10005 >> Current Logical Clock = 1
<< 1 >> Notify port :10003 >> Current Logical Clock = 2		
<< 2 >> Notify port :10004	>> Current Logical Clock = 3	
	<< 2 >> Notify port :10004	

	>> Current Logical Clock = 4	
	<< 3 >> Notify port :10005	>> Current Logical Clock = 5

2nd task - Vector logic clock

```
package main

import (
  "fmt"
  "net"
  "os"
  "strconv"
  "time"
  "bufio"
  "encoding/json"
)

type ClockStruct struct {
  Id int
  Clocks []int
}

var err string
var myPortId int
var myPort string
var nServers int
var CliConn []*net.UDPConn
var ServConn *net.UDPConn
var logicalClock ClockStruct

func CheckError(err error) {
  if err != nil {
     fmt.Println("Error: ", err)
     os.Exit(0)
  }
}

func PrintError(err error) {
  if err != nil {
     fmt.Println("Error: ", err)
  }
}

func doServerJob() {
  buf := make([]byte, 1024)
  n, addr, err := ServConn.ReadFromUDP(buf)
  var logicalClockMessage ClockStruct
  err = json.Unmarshal(buf[:n], &logicalClockMessage)
  if err != nil {
     fmt.Println("Unmarshal server response failed.")
  }

  logicalClock.Clocks[myPortId]++
  for i := 1; i < len(logicalClock.Clocks); i++ {
     if logicalClockMessage.Clocks[i] > logicalClock.Clocks[i] {
        logicalClock.Clocks[i] = logicalClockMessage.Clocks[i]
     }
  }

  fmt.Printf("Received %s from %s\nCurrent Logical Clock = %d\n", string(buf[0:n]), addr, logicalClock.Clocks)

  if err != nil {
     fmt.Println("Error: ",err)
  }
}

func doClientJob(otherProcess int, i ClockStruct) {
```

```
  Conn := CliConn[otherProcess]

  jsonRequest, err := json.Marshal(logicalClock)
  if err != nil {
     fmt.Println("Marshal connection information failed.")
  }

  _, err = Conn.Write(jsonRequest)
  if err != nil {
     fmt.Println(jsonRequest, err)
  }
}

func initConnections() {
  id, err := strconv.Atoi(os.Args[1])
  myPortId = id
  myPort = os.Args[myPortId+1]
  nServers = len(os.Args) - 2

   ServerAddr, err := net.ResolveUDPAddr("udp",myPort);
   CheckError(err)
   Conn, err := net.ListenUDP("udp", ServerAddr)
  CheckError(err)

  ServConn = Conn

  for i := 0; i < nServers; i++ {
     ServerAddr,err := net.ResolveUDPAddr("udp","127.0.0.1" + os.Args[i+2])
     CheckError(err)
     LocalAddr, err := net.ResolveUDPAddr("udp", "127.0.0.1:0")
     CheckError(err)
     Conn, err := net.DialUDP("udp", LocalAddr, ServerAddr)
     CheckError(err)

     CliConn = append(CliConn, Conn)
  }
}

func readInput(ch chan string) {
  reader := bufio.NewReader(os.Stdin)
  for {
     text, _, _ := reader.ReadLine()
     ch <- string(text)
  }
}

func main() {
  initConnections()
  defer ServConn.Close()
  logicalClock.Id = myPortId
  logicalClock.Clocks = append(logicalClock.Clocks, 0)
  for i := 0; i < nServers; i++ {
     logicalClock.Clocks = append(logicalClock.Clocks, 0)
     defer CliConn[i].Close()
  }

  ch := make(chan string)
  go readInput(ch)

  for {
     go doServerJob()
     select {
        case x, valid := <-ch:
           if valid {
              i1, err := strconv.Atoi(x)
              if (err == nil && i1 < len(os.Args) - 1 ){
                 fmt.Printf("Notify port %s\n\n", os.Args[i1+1])
                 go doClientJob(i1-1, logicalClock)
              } else {
                 fmt.Println("Invalid number")
```

```
            }
         } else {
            fmt.Println("Channel closed!")
         }
      default:
         time.Sleep(time.Second * 1)
   }
  }
}
```

Code 5: process.go file

Using Code 5, we ran different processes into different terminals in order to test the vector logic clock.

```
$ go run process.go 1 :10003 :10004 :10005
```

Code 6: first terminal

```
$ go run process.go 2 :10003 :10004 :10005
```

Code 7: second terminal

```
$ go run process.go 3 :10003 :10004 :10005
```

Code 8: third terminal

We aim to get the following illustrative output

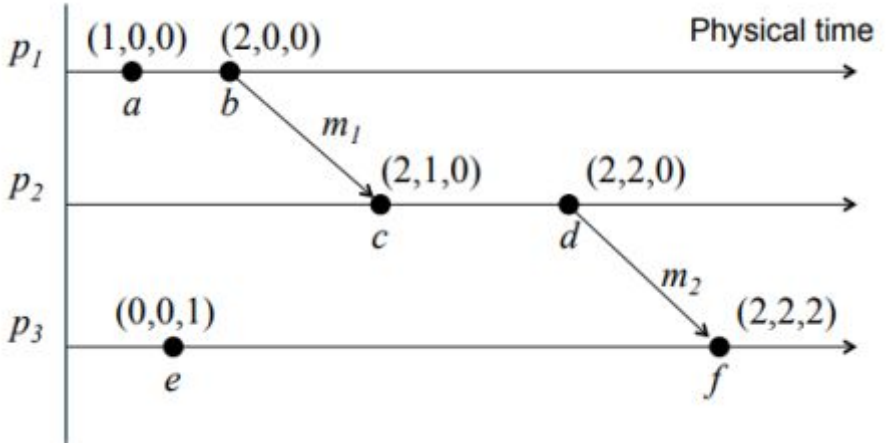

Image 2: wanted output from each process at first, second and third terminals

Note that the below output is correct according to the image above

first terminal :10003	second terminal :10004	third terminal :10005
<< 1 >> Notify port :10003 >> Current Logical Clock = [1 0 0]		
		<< 3 >> Notify port :10005 >> Current Logical Clock = [0 0 1]
>> Notify port :10003 >> Current Logical Clock = [2 0 0]		
<< 2 >> Notify port :10004	>> Current Logical Clock = [2 1 0]	
	<< 2 >> Notify port :10004	

	>> Current Logical Clock = [2 2 0]	
	<< 3 >> Notify port :10005	>> Current Logical Clock = [2 2 2]

In order to make sure the correct algorithm behavior, let's consider another example. Now, we aim to get the following illustrative output

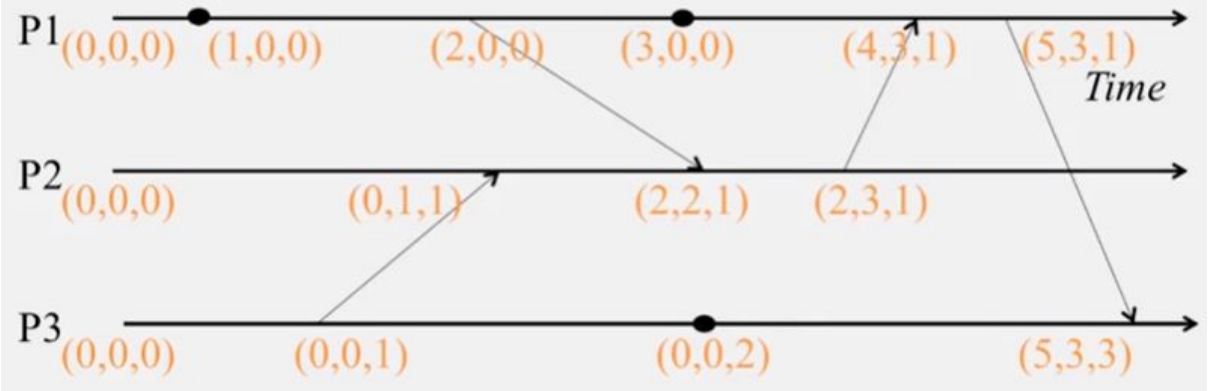

Image 3: wanted output from each process at first, second and third terminals

Note that the below output is correct according to the image above

first terminal :10003	second terminal :10004	third terminal :10005
<< 1 >> Notify port :10003 >> Current Logical Clock = [1 0 0]		
		<< 3 >> Notify port :10005 >> Current Logical Clock = [0 0 1]
	>> Current Logical Clock = [0 1 1]	<< 2 >> Notify port :10004
<< 1 >> Notify port :10003 >> Current Logical Clock = [2 0 0]		
<< 2 >> Notify port :10004	>> Current Logical Clock = [2 2 1]	
<< 1 >> Notify port :10003 >> Current Logical Clock = [3 0 0]		
		<< 3 >> Notify port :10005 >> Current Logical Clock = [0 0 2]
	<< 2 >> Notify port :10004 >> Current Logical Clock = [2 3 1]	
>> Current Logical Clock = [4 3 1]	<< 1 >> Notify port :10003	
<< 1 >> Notify port :10003 >> Current Logical Clock = [5 3 1]		
<< 3 >> Notify port :10005		>> Current Logical Clock = [5 3 3]