# Distributed Communication 12th practice

Li Jianhao lijianhao288@hotmail.com

### 1 Basics

## 1.1 Generator Example

New programs:

```
package main
import (
     "fmt"
     "github.com/streadway/amqp"
     "log"
     "os/exec"
func main() {
     conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
     failOnError(err, "Failed to connect to RabbitMQ")
                                                                                                                 10
     defer conn.Close()
     ch, err := conn.Channel()
                                                                                                                 12
     \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{open}_{\sqcup} \texttt{a}_{\sqcup} \texttt{channel"})
     err = ch.ExchangeDeclare("dispatch", "direct", false, true, false, false, nil)
     failOnError(err, "Failed to declare and exchange")
queueArg, err := ch.QueueDeclare("generatorQueue", false, true, false, false, nik)
failOnError(err, "Failed to declare ad queue")
     err = ch.QueueBind(queueArg.Name, "generator", "dispatch", false, nil)
                                                                                                                 18
     failOnError(err, "Failed_to_bind_a_queue")
     msg, err := ch.Consume(queueArg.Name, "", false, false, false, false, nil)
                                                                                                                 20
     \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{register}_{\sqcup} \texttt{a}_{\sqcup} \texttt{consumer"})
                                                                                                                 21
     go func() {
          for d := range msg {
                cmd := exec.Command("cmd", "/C", "start", "go", "run", "./Reply_Server.
                err = cmd.Run()
                \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{generate}_{\sqcup} \texttt{server"})
                {\tt fmt.Println("generated\_one\_server")}
                d.Ack(false)
                                                                                                                 28
                                                                                                                 29
     }()
     forever := make(chan bool)
                                                                                                                 31
     <-forever
                                                                                                                 32
func failOnError(err error, msg string) {
                                                                                                                 34
     if err != nil {
                                                                                                                 35
          log.Fatalf("%s:⊔%s", msg, err)
                                                                                                                 36
                                                                                                                 37
```

Listing 1: Generator example, generator

```
package main
import (
     "fmt"
                                                                                                                  3
     "github.com/streadway/amqp"
                                                                                                                  5
                                                                                                                  6
func main() {
     numWorkers := 10
                                                                                                                  8
     conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
                                                                                                                  9
     failOnError(err, "Failed_{\sqcup}to_{\sqcup}connect_{\sqcup}to_{\sqcup}RabbitMQ")
                                                                                                                  10
     defer conn.Close()
                                                                                                                  11
     cho, err := conn.Channel()
                                                                                                                  12
     failOnError(err, "Failed_{\sqcup}to_{\sqcup}open_{\sqcup}a_{\sqcup}channel")
                                                                                                                  13
     err = cho.ExchangeDeclare("dispatch", "direct",
                                                                                                                  14
     false, true, false, false, nil) failOnError(err, "Failed_{\sqcup}to_{\sqcup}declare_{\sqcup}an_{\sqcup}exchange")
                                                                                                                  15
                                                                                                                  16
     for i := 0; i < numWorkers; i++ {
                                                                                                                  17
          msg := "1"
                                                                                                                  18
          err = cho.Publish("dispatch", "generator", false, false,
                                                                                                                  19
                amqp.Publishing{
                                                                                                                  20
                     ContentType: "text/plain",
                                                                                                                  21
                     Body:
                                       []byte(msg),
                                                                                                                  22
                })
                                                                                                                  24
          \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{publish}_{\sqcup} \texttt{a}_{\sqcup} \texttt{message"})
                                                                                                                  25
          fmt.Printf("Organizer_{\square}Published_{\square}%s_{\square}\n", string(msg))
                                                                                                                  27
                                                                                                                  28
                                                                                                                  29
func failOnError(err error, msg string) {
                                                                                                                  30
     if err != nil {
                                                                                                                  31
          log.Fatalf("%s:⊔%s", msg, err)
                                                                                                                  32
     }
                                                                                                                  33
}
                                                                                                                  34
```

Listing 2: Generator example, organizer

### Programs in previous example:

```
package main
import (
                                                                                                               3
     "fmt"
                                                                                                               4
     "github.com/rs/xid"
     "github.com/streadway/amqp"
                                                                                                               6
     "log"
                                                                                                               7
     "strconv"
     "sync"
                                                                                                               9
)
                                                                                                               10
                                                                                                               11
func main() {
                                                                                                               12
     conn1, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
                                                                                                               13
     \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{connect}_{\sqcup} \texttt{to}_{\sqcup} \texttt{RabbitMQ")}
                                                                                                               14
     defer conn1.Close()
                                                                                                               15
                                                                                                               16
     conn2, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
                                                                                                               17
     failOnError(err, "Failed_{\sqcup}to_{\sqcup}connect_{\sqcup}to_{\sqcup}RabbitMQ")
                                                                                                               18
     defer conn2.Close()
                                                                                                               19
                                                                                                               20
     cho, err := conn1.Channel()
```

```
failOnError(err, "Failedutouopenuauchannel")
                                                                                              22
defer cho.Close()
                                                                                              23
chi, err := conn2.Channel()
\texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{open}_{\sqcup} \texttt{a}_{\sqcup} \texttt{channel"})
                                                                                              25
defer chi.Close()
                                                                                              26
                                                                                              27
err = cho.ExchangeDeclare("replyExchange", "direct", false, true, false, false, 28 il)
failOnError(err, "Failed_to_declare_an_exchange")
q, err := chi.QueueDeclare("", false, true, false, false, nil)
failOnError(err, "Failedutoudeclareuauqueue")
                                                                                             32
                                                                                             33
err = chi.QueueBind(q.Name, q.Name, "replyExchange", false, nil)
                                                                                              34
failOnError(err, "Failed_{\sqcup}to_{\sqcup}bind_{\sqcup}a_{\sqcup}queue")
                                                                                              35
                                                                                              36
var jobCorr = make(map[string]string)
                                                                                              37
var mu sync.Mutex
                                                                                              38
                                                                                              39
msgs, err := chi.Consume(q.Name, "", false, false, false, false, nil)
                                                                                             40
failOnError(err, "Failed_{\sqcup}to_{\sqcup}register_{\sqcup}a_{\sqcup}consumer")
                                                                                              41
                                                                                              42
go func() {
                                                                                              43
    for d := range msgs {
                                                                                              44
         mu.Lock()
                                                                                              45
         v, ok := jobCorr[d.CorrelationId]
                                                                                             46
         if ok {
                                                                                              47
              fmt.Println("Job", v, "Get_response:"+string(d.Body))
                                                                                              48
              delete(jobCorr, d.CorrelationId)
                                                                                              49
         } else {
                                                                                              50
              fmt.Println("Get_{\sqcup}a_{\sqcup}not_{\sqcup}related_{\sqcup}msg")
                                                                                              51
                                                                                              52
         mu.Unlock()
         d.Ack(false)
                                                                                              54
    }
                                                                                              55
}()
                                                                                              57
ints := []string{}
                                                                                              58
for i := 0; i < 10; i++ {
                                                                                              59
    s := strconv.Itoa(i)
                                                                                              60
     ints = append(ints, s)
                                                                                              61
                                                                                              62
                                                                                              63
for _, i := range ints {
                                                                                              64
     var corrId = randomString()
                                                                                              65
     err := cho.Publish("replyExchange", "key", false, false,
                                                                                              66
         amqp.Publishing{
                                                                                              67
              ContentType:
                                "text/plain",
                                                                                              68
              CorrelationId: corrId,
                                                                                              69
                               q.Name,
              ReplyTo:
                                                                                              70
              Body:
                                []byte(i),
                                                                                              71
         })
                                                                                              73
    failOnError(err, "Failed to publish")
                                                                                              74
     fmt.Println("Published" + i)
                                                                                              75
    mu.Lock()
                                                                                              76
     jobCorr[corrId] = i
                                                                                              77
    mu.Unlock()
                                                                                              78
}
                                                                                              79
                                                                                              80
forever := make(chan bool)
                                                                                              81
<-forever
                                                                                              82
```

```
83
func failOnError(err error, msg string) {
    if err != nil {
        log.Fatalf("%s:⊔%s", msg, err)
                                                                                           86
                                                                                           87
}
                                                                                           88
                                                                                           89
func randomString() string {
                                                                                           90
    guid := xid.New()
                                                                                           91
    return guid.String()
                                                                                           92
                                                                                           93
```

Listing 3: Reply example, Client

```
package main
                                                                                                       1
                                                                                                       2
import (
                                                                                                       3
    " fmt. "
                                                                                                       4
     "github.com/streadway/amqp"
    "log"
                                                                                                       6
    "strconv"
                                                                                                       8
func main() {
                                                                                                       10
    conn1, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
                                                                                                       11
    failOnError(err, "Failed_{\sqcup}to_{\sqcup}connect_{\sqcup}to_{\sqcup}RabbitMQ")
                                                                                                       12
    defer conn1.Close()
                                                                                                       14
    conn2, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
                                                                                                       15
    failOnError(err, "Failed to connect RabbitMQ")
    defer conn2.Close()
                                                                                                       17
                                                                                                        18
    cho, err := conn1.Channel()
                                                                                                       19
    \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{open}_{\sqcup} \texttt{a}_{\sqcup} \texttt{channel"})
                                                                                                       20
    defer cho.Close()
                                                                                                       21
    chi, err := conn2.Channel()
    \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{open}_{\sqcup} \texttt{a}_{\sqcup} \texttt{channel"})
                                                                                                       23
    defer chi.Close()
    err = cho.ExchangeDeclare("replyExchange", "direct", false, true, false, false, 2m(il)
    failOnError(err, "Failed_to_declare_an_exchange")
                                                                                                       28
    q, err := chi.QueueDeclare("jobQueue", false, true, false, false, nil)
    failOnError(err, "Failed to declare a queue")
                                                                                                       30
                                                                                                       31
    err = chi.QueueBind(q.Name, "key", "replyExchange", false, nil)
    failOnError(err, "Failed_{\sqcup}to_{\sqcup}bind_{\sqcup}a_{\sqcup}queue")
                                                                                                       33
                                                                                                       34
    msgs, err := chi.Consume(q.Name, "", false, false, false, false, nil)
                                                                                                       35
    failOnError(err, "Failed_{\sqcup}to_{\sqcup}register_{\sqcup}a_{\sqcup}consumer")
                                                                                                       36
                                                                                                       37
    forever := make(chan bool)
                                                                                                       38
                                                                                                       39
    go func() {
                                                                                                       40
         for d := range msgs {
                                                                                                       41
              i, err := strconv.Atoi(string(d.Body))
                                                                                                       42
              failOnError(err, "Failed_to_convert")
                                                                                                       43
              result := strconv.Itoa(i * 2)
                                                                                                       44
              fmt.Println("Reply_result:", result)
                                                                                                        45
              err = cho.Publish(
                                                                                                       46
                    "replyExchange", d.ReplyTo, false, false,
                                                                                                       47
```

```
amqp.Publishing{
                                                                                                                     48
                                                 "text/plain",
                            ContentType:
                                                                                                                     49
                            CorrelationId: d.CorrelationId,
                                                                                                                     50
                                                 []byte(result),
                                                                                                                     51
                                                                                                                     52
                \texttt{failOnError(err, "Failed}_{\sqcup} \texttt{to}_{\sqcup} \texttt{publish}_{\sqcup} \texttt{a}_{\sqcup} \texttt{message"})
                                                                                                                     53
                d.Ack(false)
                                                                                                                     54
     }()
                                                                                                                     56
     fmt.Println("Waiting_{\sqcup}for_{\sqcup}jobs")
                                                                                                                     57
                                                                                                                     58
                                                                                                                     59
                                                                                                                     60
func failOnError(err error, msg string) {
                                                                                                                     61
     if err != nil {
                                                                                                                     62
          log.Fatalf("%s:⊔%s", msg, err)
                                                                                                                     64
}
                                                                                                                     65
```

Listing 4: Reply example, Server

#### Output:

Run three generators and one organizer.

#### Three generators:

```
go run generator.go
generated one server
generated one server
generated one server
generated one server
4
generated one server
5
```

```
go run generator.go 1
generated one server 2
generated one server 3
generated one server 4
```

```
go run generator.go
1
generated one server
2
generated one server
3
generated one server
4
```

#### Organizer:

```
go run organizer.go
                                                                                        1
Organizer Published 1
Organizer Published 1
                                                                                        3
Organizer Published 1
                                                                                        4
Organizer Published 1
                                                                                        5
Organizer Published 1
                                                                                        6
Organizer Published 1
Organizer Published 1
                                                                                        8
Organizer Published 1
                                                                                        9
Organizer Published 1
                                                                                        10
Organizer Published 1
                                                                                        11
```

## Run the reply client.

### Reply client:

```
go run Reply_Client.go
Published job:0
Published job:1
Published job: 2
Published job: 3
                                                                                                       5
Published job:4
Published job:5
Published job:6
Published job:7
Published job:8
Published job:9
                                                                                                        10
                                                                                                        11
Job: 0 Got response:0
Job: 1 Got response:2
                                                                                                        13
Job: 2 Got response:4
                                                                                                        14
Job: 3 Got response:6
                                                                                                        15
Job: 5 Got response:10
                                                                                                        16
Job: 8 Got response:16
                                                                                                        17
Job: 7 Got response:14
                                                                                                        18
Job: 4 Got response:8
                                                                                                        19
Job: 9 Got response:18
                                                                                                        20
Job: 6 Got response:12
                                                                                                        21
```

#### Reply servers:

rtepry servers.	
Waiting for jobs	1
Received job: 0 Published response: 0	2
Waiting for jobs	1
Received job: 2 Published response: 4	2
Waiting for jobs	1
Received job: 4 Published response: 8	2
Waiting for jobs	1
Received job: 5 Published response: 10	2
Waiting for jobs	1
Received job: 1 Published response: 2	2
Waiting for jobs	1
Received job: 3 Published response: 6	2
Waiting for jobs	1
Received job: 6 Published response: 12	2
Waiting for jobs	1
Received job: 7 Published response: 14	2
Waiting for jobs	1
Received job: 9 Published response: 18	2
Waiting for jobs	1
Received job: 8 Published response: 16	2

# 2 Practice

# 2.1 p1

Create program "Solution\_generator" and "Solution\_organizer".

Run one "Reply\_Server" manually. Run one "Solution\_generator" and one "Solution\_organizer" to generate 5 "Reply\_Client". All the "Reply\_Client" will send requests and get responses.