Distributed Communication 8th practice

Li Jianhao
lijianhao288@hotmail.com

1 Basics

1.1 RabbitMQ overview

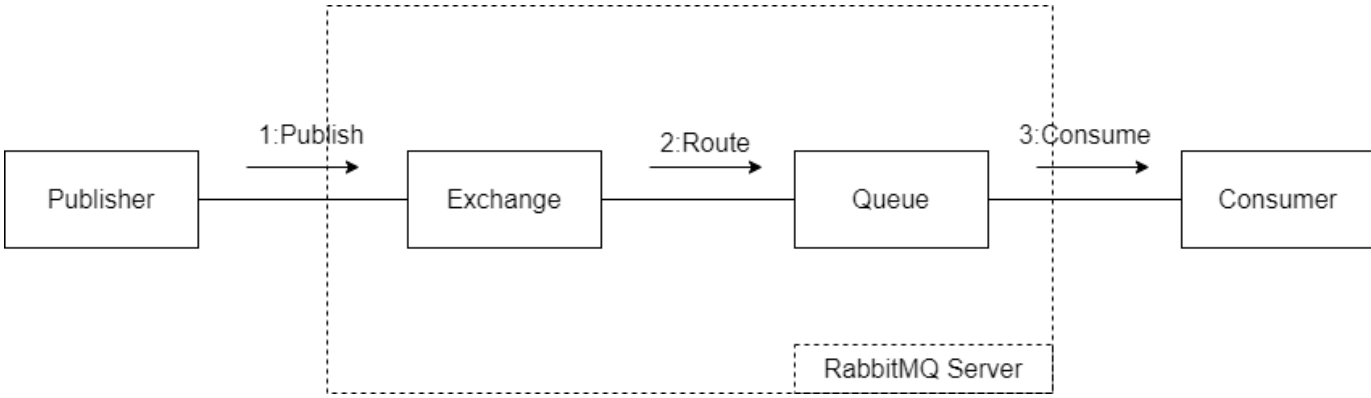

Figure 1: RabbitMQ

1.2 Go RabbitMQ Client Library

Documentation: https://pkg.go.dev/github.com/streadway/amqp

Preparation:
func Dial(url string) (*Connection, error)
func (c *Connection) Close() error
func (c *Connection) Channel() (*Channel, error)
func (ch *Channel) Close() error
func (ch *Channel) ExchangeDeclare(name, kind string, durable, autoDelete, internal, noWait bool, args Table) error

Consume related:
func (ch *Channel) QueueDeclare(name string, durable, autoDelete, exclusive, noWait bool, args Table) (Queue, error)
func (ch *Channel) QueueBind(name, key, exchange string, noWait bool, args Table) error
func (ch *Channel) Consume(queue, consumer string, autoAck, exclusive, noLocal, noWait bool, args Table) (<-chan Delivery, error)
func (d Delivery) Ack(multiple bool) error

Publish related:
func (ch *Channel) Publish(exchange, key string, mandatory, immediate bool, msg Publishing) error

1.3 RabbitMQ Hello World

```
package main

import (
    "fmt"
    "github.com/streadway/amqp"
    "log"
)

func main() {
    //Connection
    conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
    printErrorAndExit(err, "Failed to connect to RabbitMQ")
    defer conn.Close()

    //Channel
    ch, err := conn.Channel()
    printErrorAndExit(err, "Failed to open a channel")
    defer ch.Close()

    //Exchange
    err = ch.ExchangeDeclare(
        "jobExchange", // name
        "direct",      // type
        false,         // durable
        true,          // auto-deleted
        false,         // internal
        false,         // no-wait
        nil,           // arguments
    )
    printErrorAndExit(err, "Failed to declare an exchange")

    //Decalre and bind queue
    q, err := ch.QueueDeclare(
        "jobQueue", // name,,empty string let server generate id
        false,      // durable
```

```
        true,       // delete when unused
        false,      // exclusive
        false,      // no-wait
        nil,        // arguments
    )
    printErrorAndExit(err, "Failed to declare a queue")
    err = ch.QueueBind(
        q.Name,        // queue name
        "jobkey",      // routing key
        "jobExchange", // exchange
        false,
        nil)
    printErrorAndExit(err, "Failed to bind a queue")

    //Consume
    msgs, err := ch.Consume(
        q.Name, // queue
        "",     // consumer,empty string let server generate id
        false,  // auto-ack
        false,  // exclusive
        false,  // no-local
        false,  // no-wait
        nil,    // args
    )
    printErrorAndExit(err, "Failed to register a consumer")

    go func() {
        for d := range msgs {
            bodyString := string(d.Body)
            fmt.Println("Received:", bodyString)
            d.Ack(false)
        }
    }()

    fmt.Println("Waiting for msgs")
    forever := make(chan bool)
    <-forever
}

func printErrorAndExit(err error, msg string) {
    if err != nil {
        log.Fatalln(msg, ":", err)
    }
}
```

Listing 1: Consumer

```
package main

import (
    "fmt"
    "github.com/streadway/amqp"
    "log"
)

func main() {
    //Connection
    conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
    printErrorAndExit(err, "Failed to connect to RabbitMQ")
    defer conn.Close()

```

```
    //Channel
    ch, err := conn.Channel()
    printErrorAndExit(err, "Failed to open a channel")
    defer ch.Close()

    //Exchange
    err = ch.ExchangeDeclare(
        "jobExchange", // name
        "direct",      // type
        false,         // durable
        true,          // auto-deleted
        false,         // internal
        false,         // no-wait
        nil,           // arguments
    )
    printErrorAndExit(err, "Failed to declare an exchange")

    //Send Message
    body := "Hi"
    err = ch.Publish(
        "jobExchange", // exchange
        "jobkey",      // routing key
        false,         // mandatory
        false,         // immediate
        amqp.Publishing{
            ContentType: "text/plain",
            Body:        []byte(body),
        })
    printErrorAndExit(err, "Failed to publish a message")
    fmt.Println("Sent: ", body)
}

func printErrorAndExit(err error, msg string) {
    if err != nil {
        log.Fatalln(msg, ":", err)
    }
}
```

Listing 2: Publisher

Output:

Run the consumer first (declare and bind queue first, prevent message loss)

```
Waiting for msgs
Received: Hi
```

Run the publisher

```
Sent:  Hi
```

2 Practice

2.1 p1

Install RabbitMQ and run the Hello World example.

1. Golang install: https://golang.org/doc/install
2. Git Installation: https://git-scm.com/downloads/
3. Erlang Installation: https://www.erlang.org/downloads
4. RabbitMQ Server Installation:
   Windows:
   https://www.rabbitmq.com/install-windows.html
   MacOS:
   https://www.rabbitmq.com/install-homebrew.html
5. Start RabbitMQ service:
   In the windows start menu click on the "RabbitMQ service start".
   Or navigate to sbin folder, for example "rabbitmq_server-3.8.14\sbin", and run the command:
   $< rabbitmq - service.bat\ start >$
6. Create a folder, copy publisher.go and consumer.go to this folder.
7. Create a module. In the created folder, run: go mod init $< NameOfModule >$
   (NameOfModule here can be the folder name.)
8. Get the Go RabbitMQ Client Library. In the created folder, run:
   go get github.com/streadway/amqp
9. First run the consumer.go, wait until it printed "Waiting for msgs"
10. Run the publisher.go

2.2 p2

Modify the hello world example. The publisher sends a "3". The consumer print out what received, double it, and prints out the result.