# Distributed Systems

## Assignment I

Mohn,n,ammad Mahdi Islami

Mahdi Karami

# توضیحات کلی

در ابتدا پس از آشنایی با زبان برنامه نویسی Go با استفاده از Socket Programing سه موجودیت سرور، کلاینت و بروکر ساده را طراحی کردیم به طوری که بروکر دارای دو سوکت، یک سوکت به برای اتصال به سرور و یک سوکت برای اتصال به کلاینت. در ابتدا تنها وظیفه ی بروکر انتقال پیام سرور به کلاینت بود. در ادامه با ایجاد یک صف در بروکر پیام های سرور را به ترتیبی که در صف قرار می‌گرفتند به کلاینت انتقال میدادیم.

برای حالت synchronous سرور باید منتظر بماند تا Ack پیامی که ارسال کرده را دریافت کند و حالت Asynchronous سرور مداوم پیام را ارسال می‌کند و در ادامه از بروکر انتظار دارد که خبر رسیدن بسته به کلاینت را به او بدهد.

در این حین اگر کلاینت ها در دسترس نباشند پیام های سرور در صف بروکر ذخیره می‌شوند تا به نوبت به صاحبانشان بفرستند.

کلاینت نیز موظف است پس از دریافت بسته Ack مناسب را تولید و به بروکر تحویل بدهد. بروکر Ack ها را نیز به ترتیب در صف ذخیره میکند و آن‌ها را در موقع مناسب به سرور ارسال میکند.

# سرور

سرور بر روی پورت 1234 localhost ران میشود و منتظر اتصال میماند. سرور که وظیفه ی اصلی آن تولید پیام و ارسال به بروکر است دارای دو Thread اصلی است که یکی پیام ها را از بروکر دریافت و دیگری پیام را به بروکر ارسال میکند را به حالات زیر افراز کرده ایم.

- **go sendToBrokerFunc(c)**
- **go receiveFromBrokerFunc(c)**

هر کدام از این تردها را به بخش های زیر افراز کرده ایم:

- **sendToBrokerFunc :**
  - RECEIVE_FROM_BROKER:

در این بخش سرور منتظر است تا پیامی از طرف بروکر به آن ارسال شود که سه حالت دارد:

پیام y است یعنی بروکر دچار overflow شده است و سرور در حالت انتظار میرود تا پیام دیگری از طرف بروکر ارسال شود.

پیام دریافتی n + number باشد در این صورت سرور میفهمد که بروکر هنوز جا دارد و عدد number میزان فضای خالی صف بروکر است.

پیام دریافی ACK#number باشد که در واقع تصدیق صحت ارسال پیام خواهد بود و سرور به حالت چک کردن عدد Ack می‌رود.

- CHECK_ACKNOWLEDGE_NUMBER

در این حالت سرور چک میکند که آیا Ack پیامی که ارسال کرده است دریافت شده است یا خیر. اگر درست بود به مرحله ی دریافت از کاربر سرور میرود در و آن را در یک آرایه ذخیره میکند. در غیر این صورت دوباره منتظر دریافت پیام از بروکر میشود.

## • sendToBrokerFunc :

- RECEIVE_FROM_INPUT

در این مرحله سرور منتظر است تا از طریق کنسول پیامی وارد شود. پیام برای این که درست عمل کند باید دارای ساختار زیر باشد.

- Client Index + text + (a or nothing)
- Client Index : starts from 0
- Message Text : Desired Text
- a: for Asynchronous

اگر در انتهای پیام a بگذاریم پیام به صورت آسینک ارسال میشود و اگر در انتهای پیام چیزی قرار ندهیم پیام به صورت سینک در نظر گرفته میشود.

- PRODUCING_PACKET

در این مرحله پکت با ساختاری ساخته میشود که برای بروکر و کلاینت قابل درک باشد:

```
"PACKET#" + messageInex + "*" + string(text[0]) + ":" + text[1:] + "\n"
```

- SEND_TO_BROKER

در این مرحله پکت با دستور زیر به بروکر ارسال میشود.

```
socket.Write([]byte(packet)|)
```

- SEND_OVERFLOW_BIT

- در این مرحله سرور از بروکر میخواهد که ظرفیت خود را به آن اعلام کند تا overflow پیش نیاید. و به حالت سیستم را به حالت RECEIVE_FROM_BROKER تغییر می‌دهد.

# Asynchronous vs synchronous

طبق توضیحات داده شده بروکر هر دوی این نوع انتقال پیام را پشتیبانی میکند. اگر درخواستی بدون a آخر برسد سرور منتظر میماند تا ACK برسد و ورودی نمی‌پذیرد اما اگر با a آخر برسد، تنها پیام را میفرستد و مجددا از کنسول ورودی میگرد.

# بروکر

بروکر آدرس سرور را می‌گیرد تا به آن متصل شود. خودش هم به پورت‌هایی که به عنوان ورودی میگیرد بر روی localhost منتظر اتصال کلاینت(ها) باقی می‌ماند.

بروکر دارای یک صف است که که پیام های سرور را از طریق آن به کلاینت(ها) ارسال میکند.

بروکر دارای دو thread است که که یکی مسئول دریافت داده ها از سرور و ذخیره ی آن‌ّها در صف و دیگر مسئول ارسال داده‌ّها به کلاینت(ها) است.

- **go serverSideFunc(serverSide)**
- **go clientSideFunc(serverSide, clientSides)**

هر کدام از این تردها به حالات زیر در کد افراز می‌شوند.

## • serverSideFunc

### ▪ RECEIVE_FROM_SERVER

در این حالت بروکر منتظر رسید پیام از سرور است. در این مرحله بروکر هدری از پیام را که مشخص میکند به کدام کلاینت باید ارسال شود را از پیام دیکود میکند و در یک دیکشنری پیام به همراه شماره کلاینت را ذخیره میکند تا در هنگام ارسال از این دیکشنری استفاده نماید. سپس سیستم به مرحله ی ذخیر در صف تغییر حالت میدهد.

- QUEUE_SERVER_MESSEGE

در این مرحله پیام سرور در صف بروکر ذخیره میشود. صفی که در بروکر به کار گرفته شده در واقع یک آرایه از استرینگ هاست که بدین صورت جهت enqueue و dequeue از آن استفاده میکنیم و در این مرحله از سایت gobyexample.com راهنمایی گرفته شده است.

```
func enqueue(queue[] string, element string) []string {
  queue = append(queue, element); // Simply append to enqueue.
  return queue
}

func dequeue(queue[] string) ([]string) {
  return queue[1:]; // Slice off the element once it is dequeued.

}
```

- **clientSideFunc**

- SEND_READY_TO_CLIENT

در این مرحله از ترد سمت کلاینت، سرور به کلاینت درخواست ارسال را میدهد تا ببنید آیا کلاینت آمادگی دریافت پیام را دارد یا خیر. با استفاده از دستور :

clientSideInputs[messages[queue[0]]].Write([]byte("READY FOR TRANSFER...?\n"))

messages[queue[۰] ] در واقع شماره ی کلاینتی است که پیام سر صف باید به آن ارسال شود. این شماره و پیام به صورت یک مپ نگهداری شده اند.

- RECEIVE_READY_FROM_CLIENT

در این مرحله بروکر منتظر اعلام آمادگی از طرف کلاینت باقی می‌ماند چناچه کلاینت "yes" بفرستد به گام بعدی یعنی ارسال پیام شیفت میکنیم و اگر "no" دریافت کنیم از سمت کلاینت، به مرحله ی قبل یعنی ارسال پیام آمادگی به کلاینت باز میگردیم.

- SEND_TO_CLIENT

در این مرحله شروع به ارسال پیام ها با توجه صف و مپینگی که میان پیام ها و شماره کلاینت ها برقرار کرده ایم میکنیم. و به ترتیب صف کل صف را تخلیه میکنیم.

همچنین ACK هایی هم که از سمت کلاینت ارسال میشود را در صف ذخیره میکنیم تا در موقع مناسب آن ها را تحویل سرور دهیم.

در این مرحله پیام هایی که ارسال شدند را از صف خارج میکنیم و ACK هایی که هم که به سرور داده میشوند را از صف ACK ها خارج میکنیم.

و بسته به شرطهایی که قرار داده ایم حالت سیستم بروکر را تغییر میدهیم.

```
if(clientSideStauts == SEND_TO_CLIENT){
  isNotReady := false
    for i := 0; i < len(queue); i++ {
      if(clientReady[messages[queue[0]]] == true){
          clientSideInputs[messages[queue[0]]].Write([]byte(queue[0]))
          clientMessage, ERROR_RECEIVE_FROM_CLIENT = bufio.NewReader(clientSideInputs[messages[queue[0]]]).ReadString('\n')
          if ERROR_RECEIVE_FROM_CLIENT != nil {
              fmt.Println(ERROR_RECEIVE_FROM_CLIENT)
              return
          }
          queue = dequeue(queue)
          AckQueue = enqueue(AckQueue, clientMessage)
          fmt.Fprintf(serverSideInput, AckQueue[0])
          AckQueue = dequeue(AckQueue)

      } else { clientSideStauts = SEND_READY_TO_CLIENT
            isNotReady = true
            break
      }
  }
```

# کلاینت

کلاینت وظیفه اش اعلام آمادگی جهت دریافت پیام از سمت سرور و بروکر و تولیدACK مناسب و نهایت ارسال آن به بروکر است.

کلاینت داری یک ترد است و با دریافت IP و port بروکر در ورودی برنامه، به آن متصل می‌شود.

کلاینت دارای یک ترد است و به حالات زیر افراض می‌شود.

- RECEIVE_READY_FROM_BROKER

در این مرحله کلاینت منتظر است تا از طرف بروکر درخواست اعلام آمادگی دریافت کند.

- SEND_READY_TO_BROKER

در این مرحله اگر وضعیت کلاینت آماده دریافت بود پیام "yes" در غیر این صورت "no" را ارسال میکند. اگر no ارسال شد دوباره به وضعیت قبلی تغییر حالت میدهد تا پیام آمادگی را دریافت کند.

- RECEIVE_FROM_BROKER

در این حالت کلاینت داده را از بروکر دریافت میکند

- PARSING_PACKET

در این حالت بروکر داده را باز میکند تا محتوای اصلی را بخواند و نیز ACK مناسب را تولید کند.

- ANALYSE_PACKET

در حالت کلاینت مختار است تا بر اساس محتوای اصلی پیام پردازشی رو آن انجام دهد که ما به پرینت کردن پیام اصلی کفایت کرده ایم.

- PRODUCING_PACKET

در این حالت کلاینت بسته ی مخصوص ACK را تولید میکند که ساختاری مانند شکل زیر دارد.

```
if( stauts == PRODUCING_PACKET) {
    //Packet Format Sample: ACK#56
    packet = "ACK#" + strconv.Itoa(acknowledge) + "\n"
    stauts = SEND_TO_BROKER

}
```

- SEND_TO_BROKER

در این مرحله کلاینت داده را با دستور زیر به بروکر ارسال میکند

# OverFlow

برای هندل کردن Overflow سرور در هر مرحله ظرفیت باقی‌مانده را از بروکر دریافت میکند و بر اساس آن تعدادی پیام را به بروکر ارسال میکند و مجددا این عملیات تکرار می‌شود تا بسته ای Loss نشود.

بروکر در هر مرحله اگر overflow رخ داده باشد،"yes" و در غیر این صورت "no"+number را میفرستد که nimber ظرفیت باقی‌مانده ی صف است.

سرور با توجه به مقداری که دریافت میکند تصمیم میگرد تا چند پیام را میتواند بفرستد.

باتشکر

محمدمهدی‌اسلامی

مهدی‌کرمی