Quorum PROTOCOL USING PUB-SUB (websockets)

Requirements:

go get "github.com/gorilla/websocket"

go get "github.com/satori/go.uuid"

Assumptions

* None of the backup servers fails to respond
* The broker does not fail

-run controller.go

-run database1.go

-run database2.go

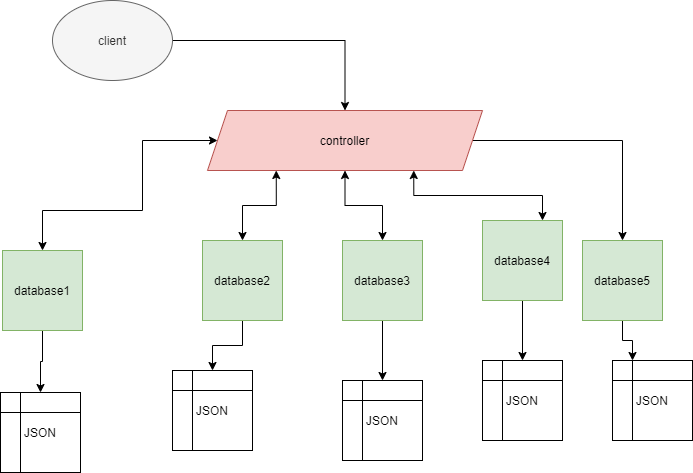
-run database3.go

-run database4.go

-run database5.go

At last check the database1 to 5 JSON files before running the client

-run client.go (upddates the database and its JSON files)



Controller acts as a broker ; client as publisher , databases as subscribers

Considering a huge employee company like AT&T, which has strict regulations on compliance online course training for their employees , maintains learning completions data in 5 databases.Any client when querying about their completion courses count should get a reply number . If the compliance coursse count is less than 70 they are non complaint .

Data stored in databases are saved as JSON internally

{  
 "UserNameValue": [  
 {  
 "UniqueUserName": "aravind",  
 "Completions": "70",  
 "Status": "COMPLIANT",  
 "Version": 29  
 },  
 {  
 "UniqueUserName": "krishnan",  
 "Completions": "89",  
 "Status": "COMPLIANT",  
 "Version": 8  
 }  
 ]  
}

Considering two employees “aravind and krishnan”, programmed in such away that if client calls for data throught the controller , the database 4 and 5 fails to update for employee krishnan ; but using quorum protocol should return the latest using the version numbers.

The broker.go takes in clients request,appends them to the clients slice and based on the “action “and “topic” parameters it is then appended to subscriptions slice or is used to publish data based on the topic.

Clients send publish and subscribe request using

'{"action":" ","topic":" ",”message”:”message data”}'

Communication is established using websockets importing

"github.com/gorilla/websocket"

Unique id to the clients are given using uuid importing

"github.com/satori/go.uuid"

Producer uses Dial function to connect to broker

websocket.DefaultDialer.Dial(u.String(), nil)

Subscriber uses javascript

conn = new WebSocket("ws://localhost:8000/broker");

And chart js to plot data

Simple websocket program:

//url address to connect to

var addr = flag.String("addr", "localhost:8000", "http service address"

func main(){

//url with path  
 u := url.URL{Scheme: "ws", Host: \*addr, Path: "/broker"}  
 log.Printf("connecting to %s", u.String())

//Dial with url to connect to the broker  
  
 c, \_, err := websocket.DefaultDialer.Dial(u.String(), nil)

if err != nil {  
 log.Fatal("dial:", err)  
 }

//to read messages from broker

\_, message, err := c.ReadMessage()

if err != nil {  
 log.Println("read:", err)  
 return  
 }  
fmt.Println(message)

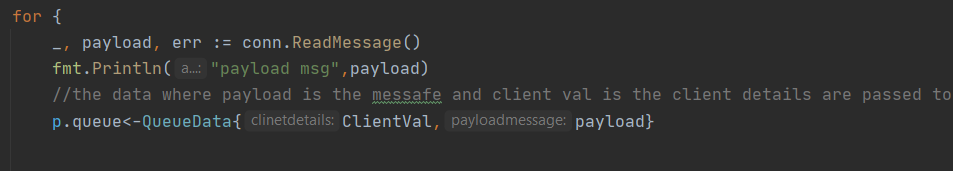
newsnotification:="hello this is a new news "  
msg:="{\"action\":\"publish\",\"topic\":\"abc\",\"message\":\""+newsnotification+"\"}"

//msg is sent using write message and is passed as bytes

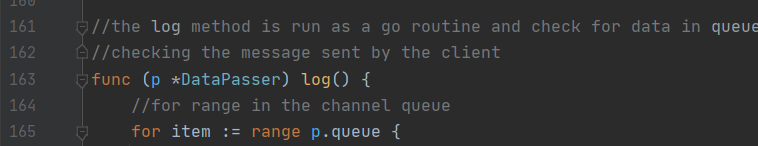
if err = c.WriteMessage(1, []byte(msg)); err != nil {  
 fmt.Println(err)  
}

Controller .go

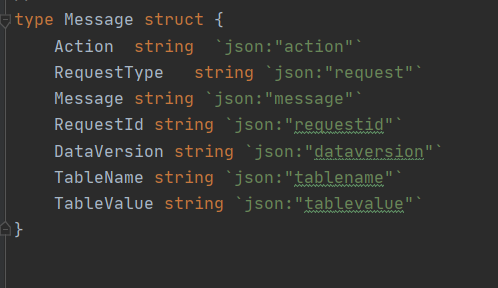
Whenever a the websocket connection reads a message it send the data throught the queue to the go routine log()

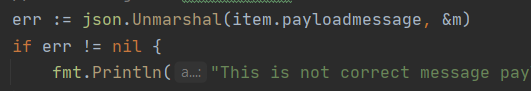


Log()



The data is converted from JSON to go struct



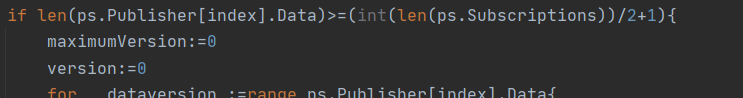


With action message in the json we determine wherether to publish or subscribe or readquorum or write quorum(response from the databases)



In both read and write make total databses/2+1 values are read or written

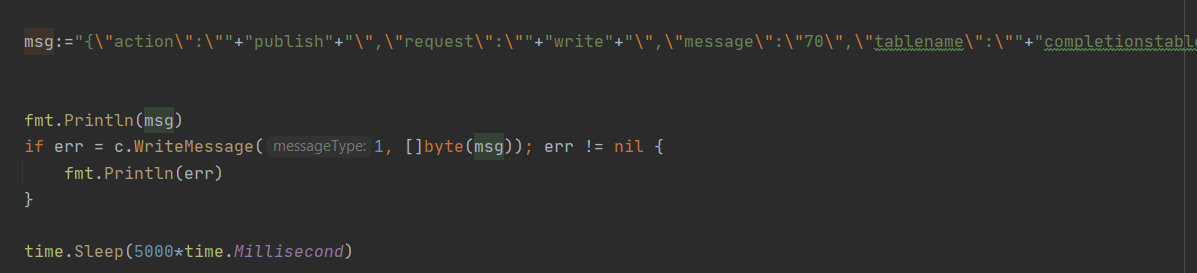
The number of databases are the subscribers



Client.go

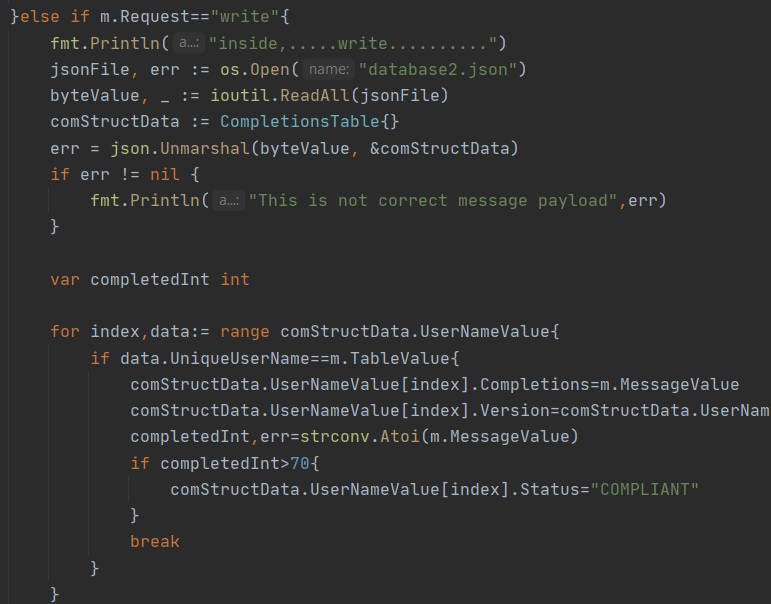
Client messages the commands in the form of JSON

Based on the data in message and tablevalue the data is edited by the databases



Databses uses JSON as internal storage; if write then data is read and updated with the new value and version number is increased

Expect for krishnan where the data is not written to databases 4 and 5 to show quorum read protocol



If read then the data is sent back using the versin number

