CS425 MP1

zecheng3 wanying5

February 2019

1 Introduction

The netids of the people in our group: zecheng3, wanying5.
The number of our VM cluster: 08.
The URL of our gitlab repository: https://gitlab.engr.illinois.edu/zecheng3/cs425_mp1
The full commit hash: 542a34b6aa6add597132baf2356c7147765e6130

2 How to run

First, compile with "go build mp1.go".
Then, run with "./mp1 <name> <port> <n>"

3 Design

- Marshalled Message Format
  We use a JSON message format All "message"s mentioned below will follow this format
  Message {
  UserName: Read from the input,
  TimeStamp: Maintain by our algorithm
  Address: The address of the sender
  Text: The content of messages
  }

- Algorithms used
  In order to achieve the reliability of delivery even in case of nodes failure, we have implemented R-multicast. We maintain a local map $ReceivedMsg$ for every node, whose key is the address of the sender, value is a slice of int representing the messages sent by the sender and received by that node. And in order to achieve the causal order of the messages, we have applied the algorithm of vector timestamps in our chatroom. In details, along with the message text, a node will send its timestamp to each other. The timestamp is stored as a map, whose key is the address of the sender, value is the updated timestamp of the sender thread. And the receiver will keep a timestamp as the local timestamp. The algorithm follows the following rules:

  – Whenever a node receives a message, it begins to deal with the timestamp received along with the message in the background. $Msg.TS$ is the timestamp stored in message, $localTS$ is the timestamp stored locally.

```
dealWithMessage(Msg)
if(Msg.TS[addr] in ReceivedMsg[addr]) //Implement R-Multicast to guarantee the reliable delivery
    Return
ReceivedMsg[addr].append(Msg.TS[addr])
send Msg to all other nodes // B-multicast the Msg to all other nodes
// do the following vector timestamps algorithm in the background
wait until (Msg.TS[Msg.addr] == localTS[Msg.addr] + 1
    and Msg.TS[addr] ≤ localTS[addr] for EACH addr in Msg.TS.keys()) other than Msg.addr)
    localTS = Msg.TS //Update the localTS
    get name and text from Msg then display on console according to the required format //deliver
Return
```

  - Whenever a node reads a text from the standard input, the local timestamp will be updated according to algorithm of vector timestamps and it will B-multicast the message to all the other nodes. The message follows the JSON message format mentioned above. The timestamp in this message is the updated local timestamp.(sender)

- Set up the network
  Basically, our idea of developing the chat room is multithreading. We use TCP connections between our nodes to create a reliable, ordered transport. We assign each node to be a server, and nodes can communicate with each other. We get the DNS of all the VMs and use **DNS:port** as the address. Each node will keep dialing all the alive nodes among these 10 VMs.

- Node Joins
  We have maintained a map **MemMap** to store the information (address and name) of all the members in the chat room. When a node joins, we add the information of this node to the map. If the length of this map reaches the capacity of this chat room, "READY" is printed, and members can begin to chat.

- Node Fails (Failure Detection)
  The node keeps dialing each other, and all of them can detect the failure when they fail to dial the node. If the node fails, it is removed from the **MemMap**. And a "**Name** has left" is printed in chat room of each node. Failure detection is achieved by TCP connection: if one node exits/fails, the other nodes will automatically detect this failure by simply listening to it. We catch that as failure detection.

- Node Leaves
  Node leaves are treated the same as the node failure.

4 pointers to where parts of the protocol are implemented in the code

- Algorithm of vector timestamps to guarantee causal order: line 218 to line 233 and line 95 to line 108
- R-multicast to guarantee the reliable delivery: line 89 to line 94
- Failure detection: line 252 to line 272