**Assignment 3**

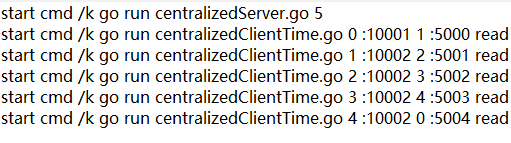
Using the GO language, implement the following variant of Ivy architecture:

1. **(10 marks) First implement the Ivy architecture discussed in the class**

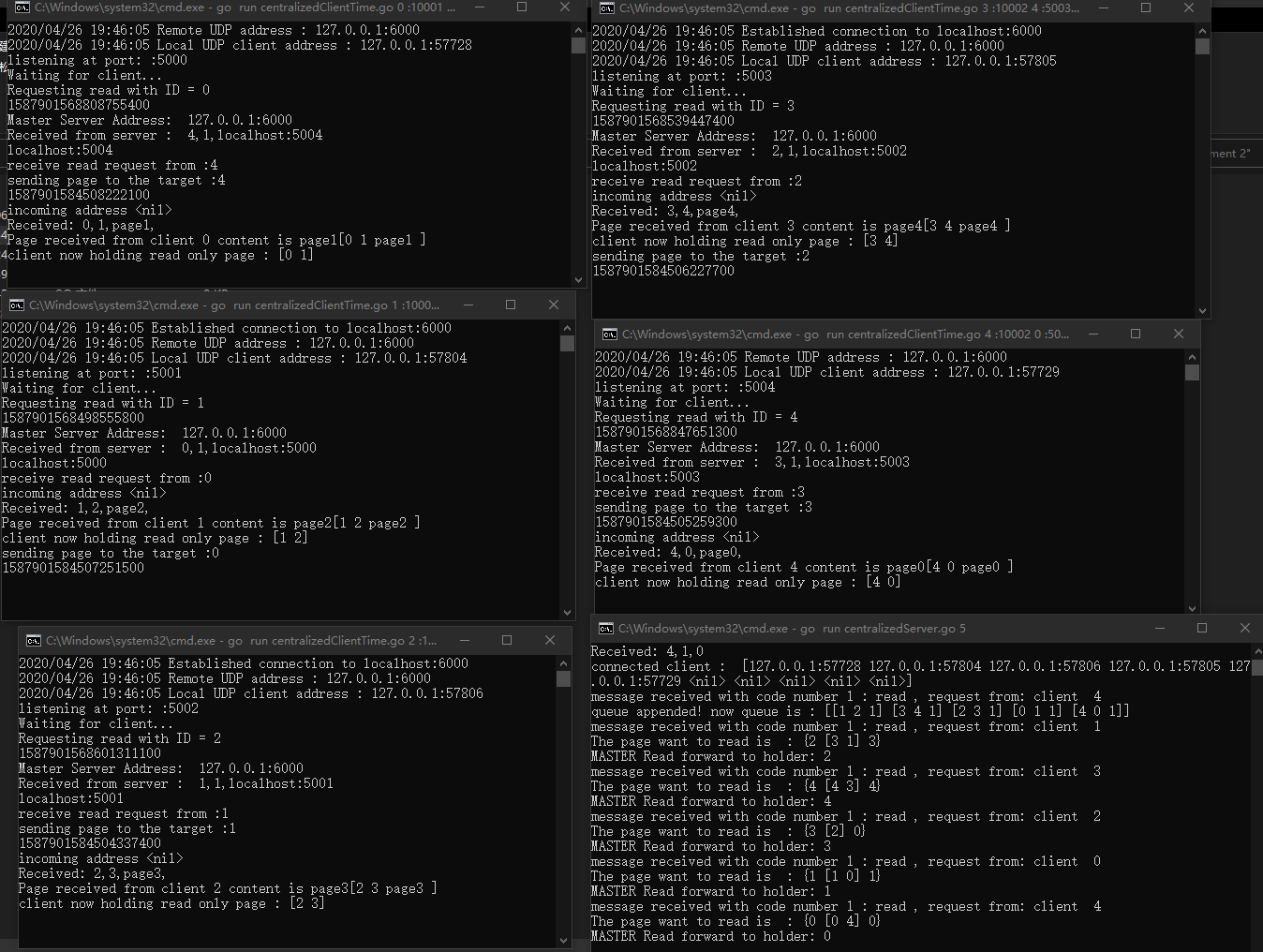
* Initially, each client will be the page holder for the page with same id.

For example, client 0 will hold page0, client 1 will hold page 1

Read Request:

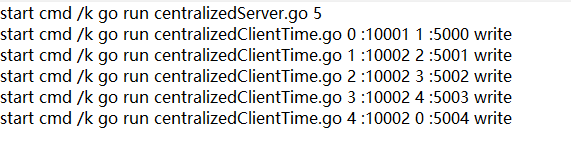


Each Client will request to read the page that has id = client+1 (form a circle)

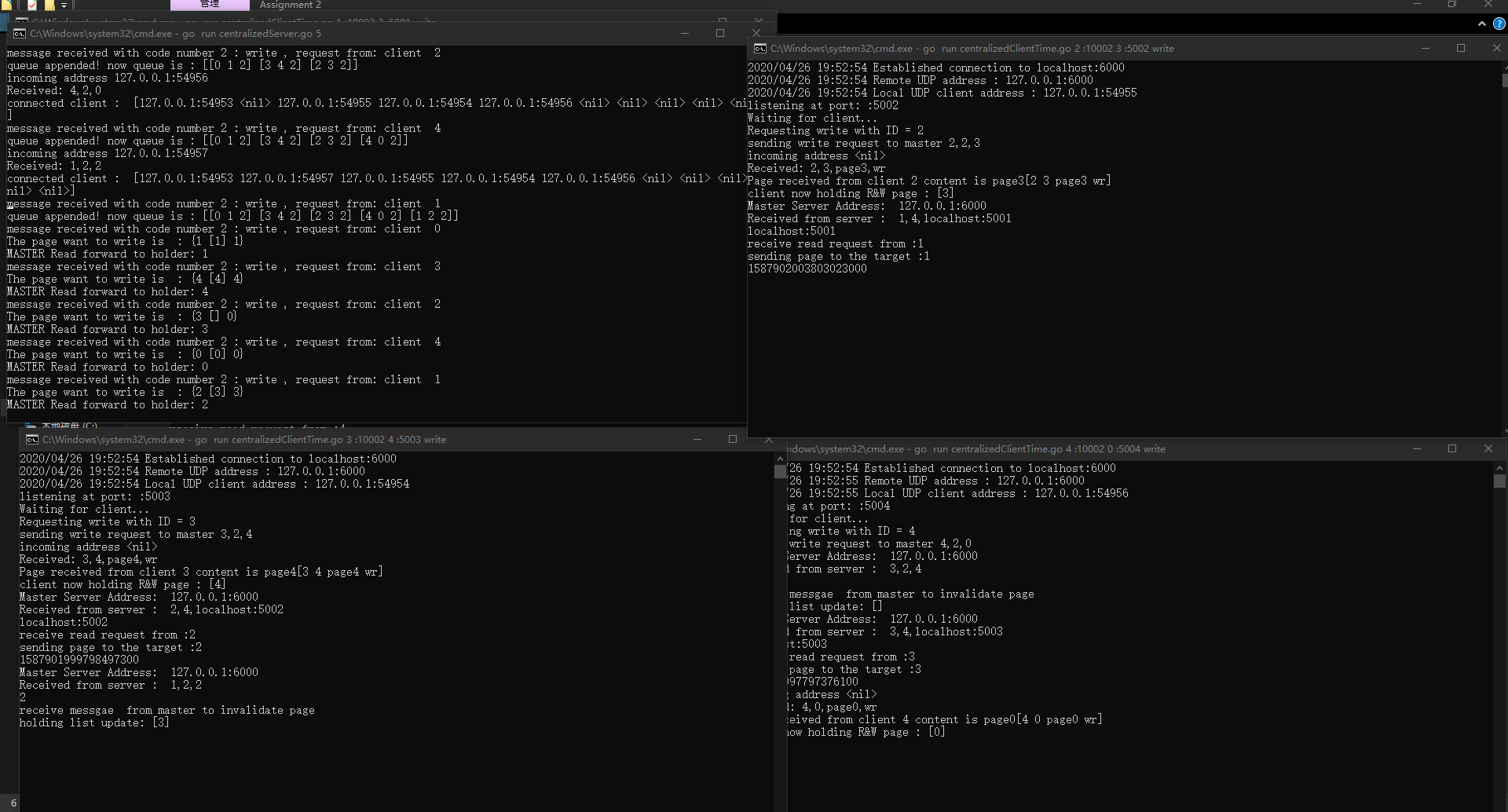


From the result we can see the holding list of each client is updated, the holding list not only holds its own page and holds the page that is readable.

Write Request



Each Client will request to write the page that has id = client+1 (form a circle)

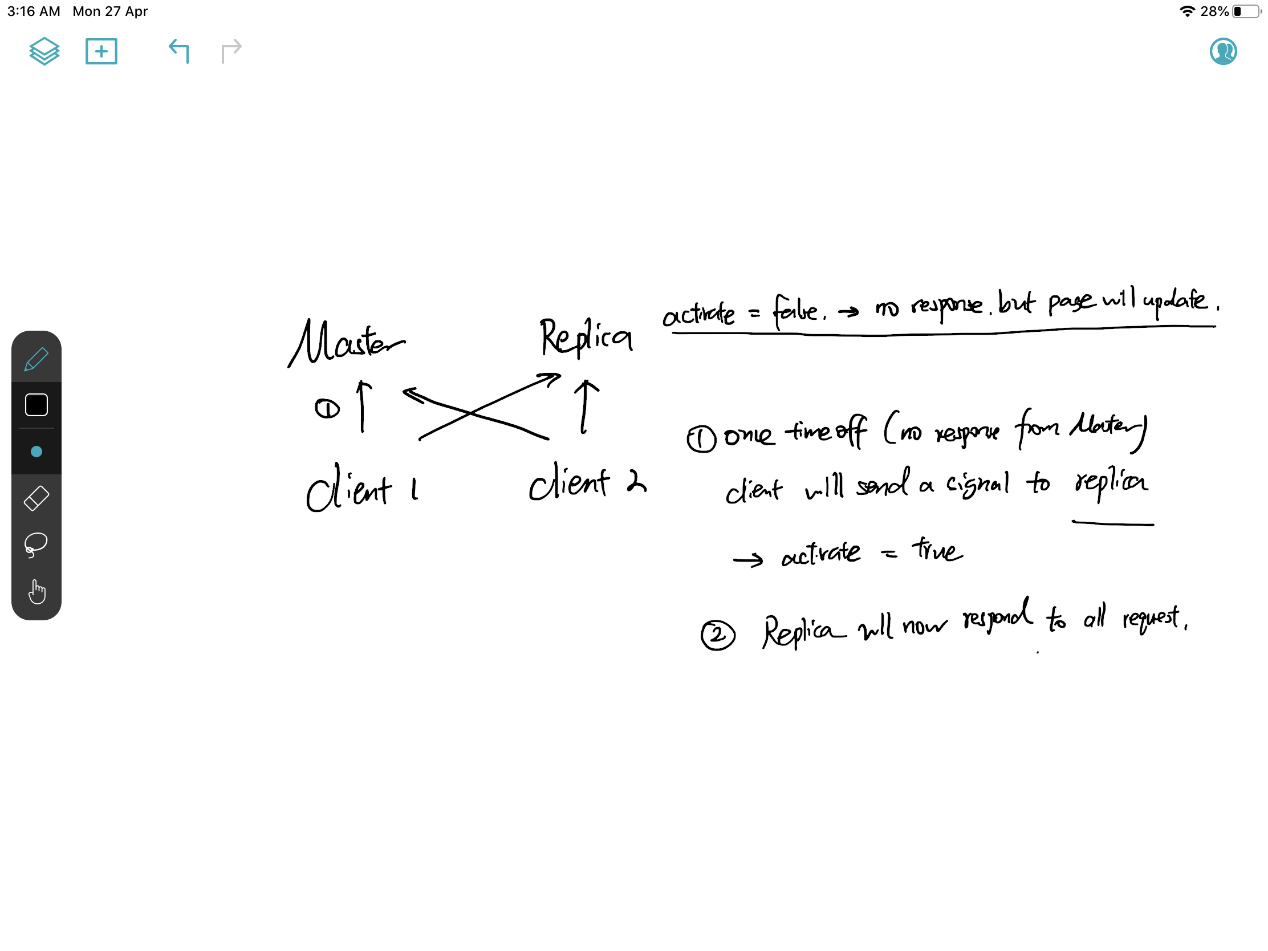


There is a priority queue in server to maintain the request queue.

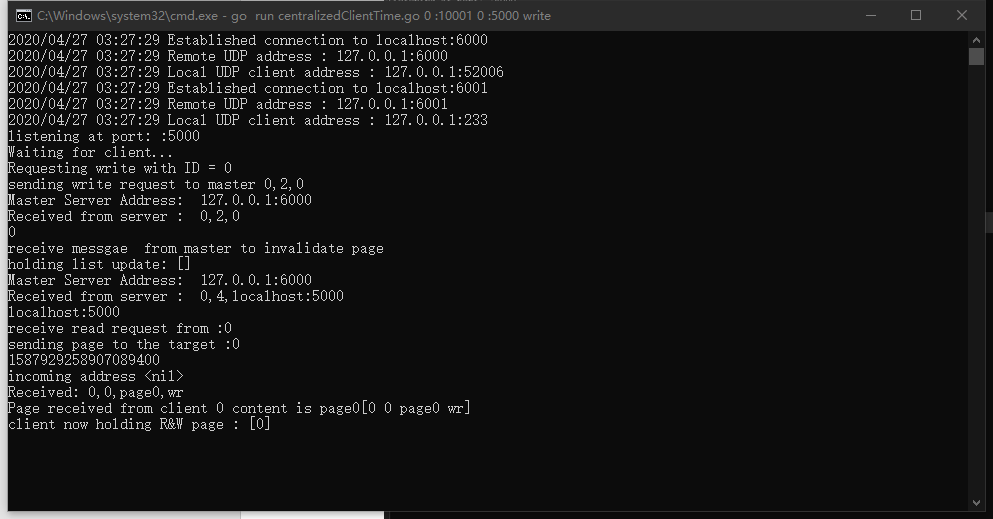
From the result, the original holding list will be invalidated by the server message and the copy will be sent to target machine, at the same time, the holding READ AND WRITE list will be updated.

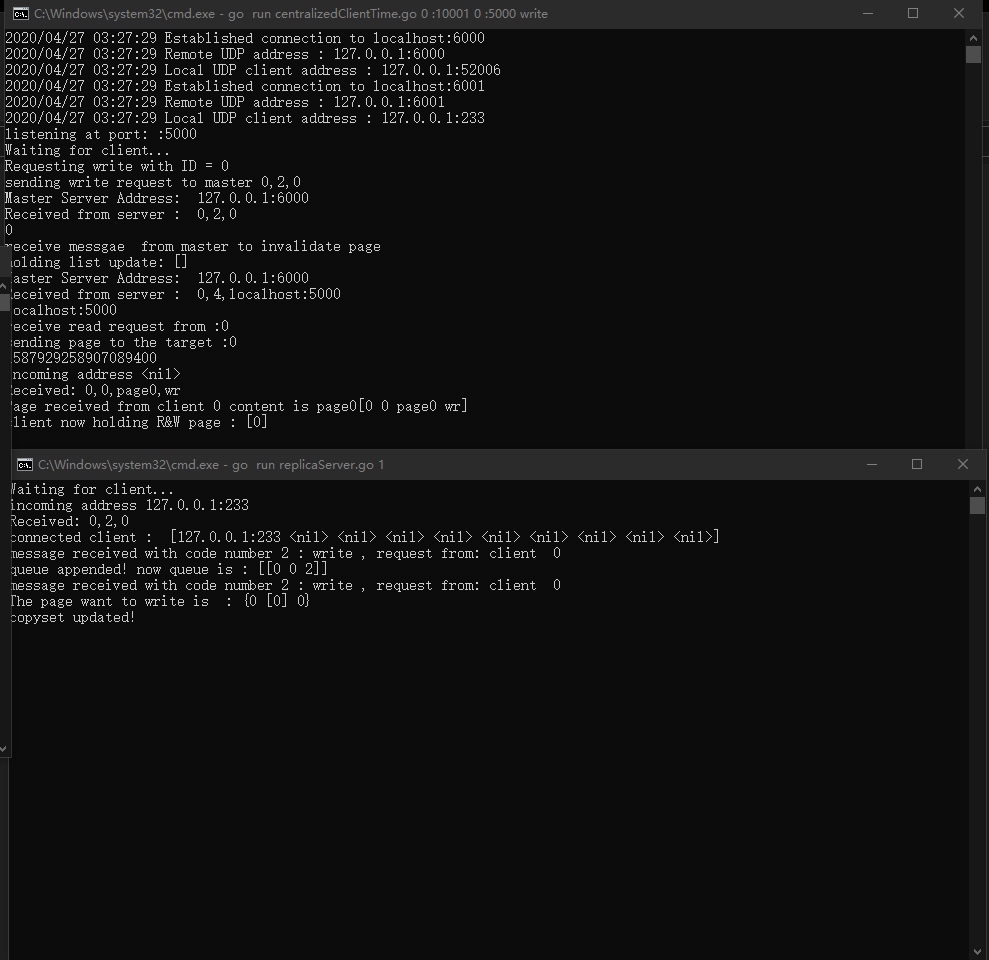
1. **(15+15+15 marks) Based on the aforementioned ideas, design and implement the fault tolerant version of the Ivy architecture (the primary and secondary replicas for the CM with consistency + election to choose a primary replica + related changes in the basic Ivy protocol)**

* **My design:** 
  + **The clients will send requests to both master and replica at same time**
  + **However, when activate = false in replica, the replica will update page according to request but not sending any messages (Therefore, the replica will hold same record with master)**
  + **When master fails, the clients detect the fail by timeout, an activate message will be sent to replica, Activate = true**
  + **Since there are multiple replicas received activate message, the bully algorithm election will start, one replica will be selected.**
  + **Now the replica will replace master and start to send message**



**When there is one client and there is no fault occurs, the client will send write request to both Master and Replica**

****

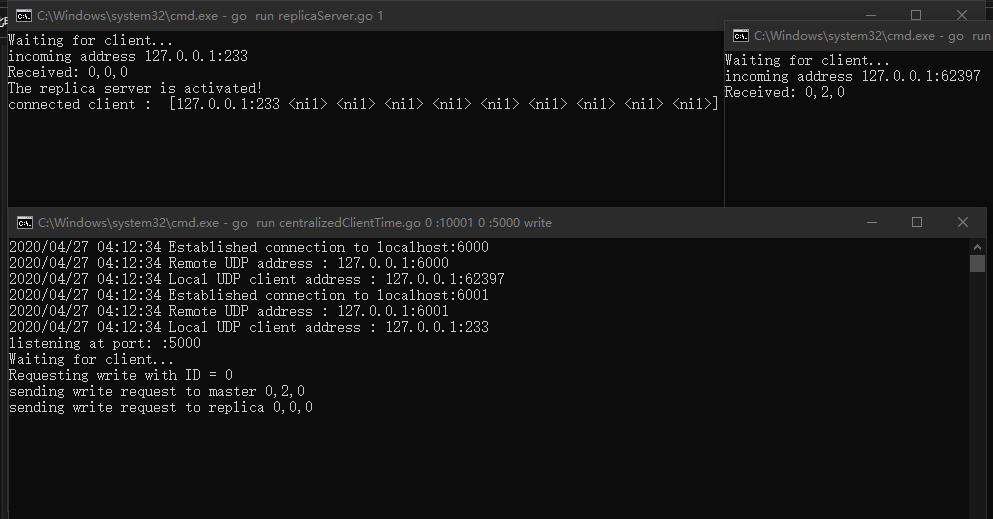
****

**The master server will respond to the request and send messages, but replica server will only update the record to keep the record same as the master.**

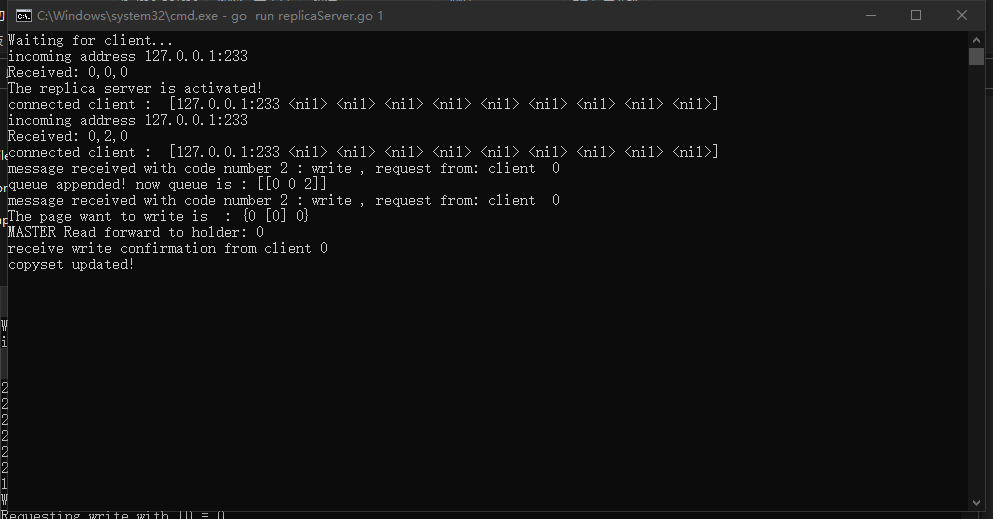
**When there is fault:**

When there is a fault, the client sends normal message to master but there is no reply,]

The client will send 0,0,0 to replica to say replica is activated

****

**The replica server will take over master’s job**

****

Argue whether your design still preserves sequential consistency (a short paragraph will suffice).

According to the rules of sequential consistency:

***#1: All operations in one machine are executed in order (e.g. in program order)***

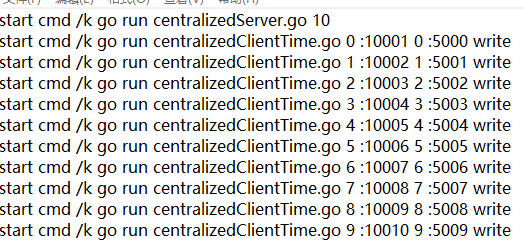
***#2: All machines observe results according to some total order (i.e. reads see most recent writes according to this order)***

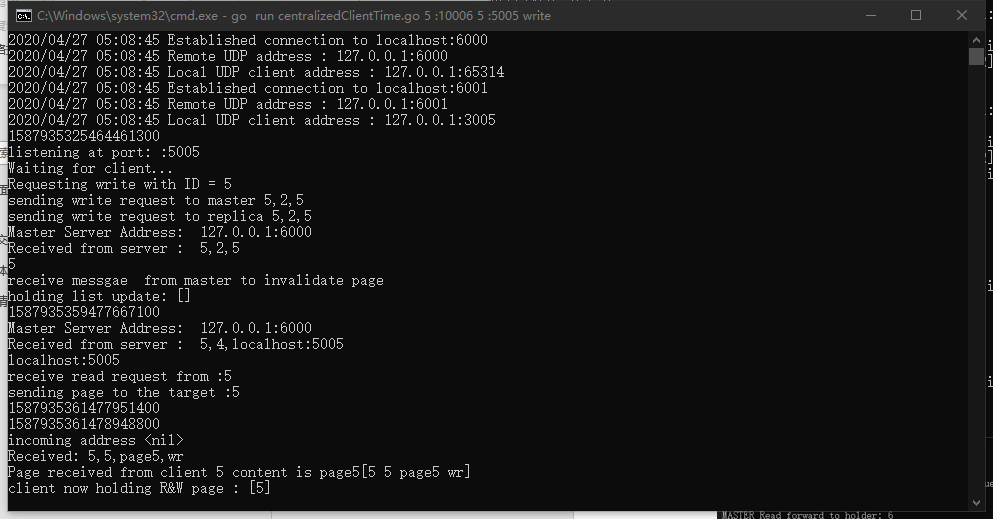
sequential consistency implies that operations appear to take place in some total order, and that that order is consistent with the order of operations on each individual process.

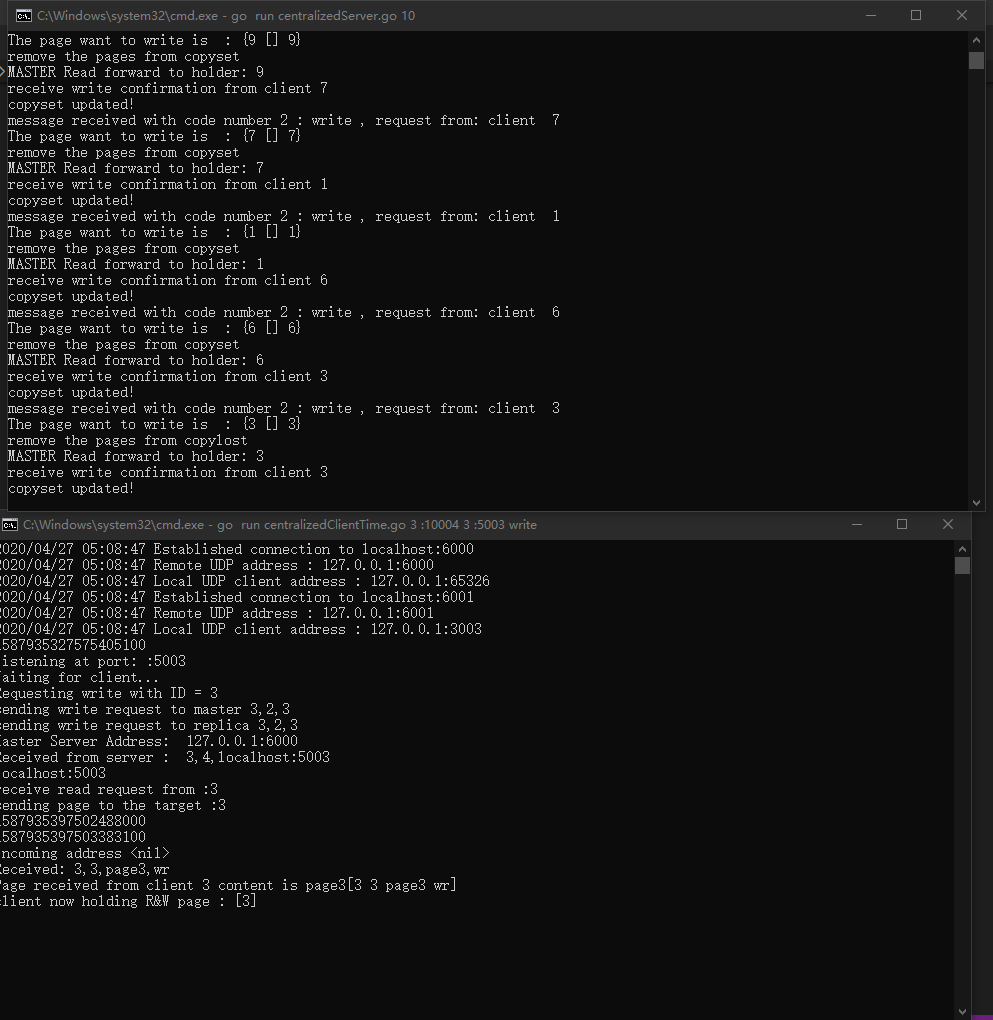
Both Rule 1 and Rule 2 are satisfied as even there is a fault occur, both the master and the replica have a FIFO queue to store the operation order, therefore, the operations are executed in order and there will be total order(preserved in the FIFO queue).

Experiment

Without any faults, compare the performance of the basic version of Ivy protocol and the new fault tolerant version using requests from at least 10 clients. You should assume at least three replicas for the central manager.:







Original design:

Time: 72038921728

Add fault tolerance:

Time: 72042456780

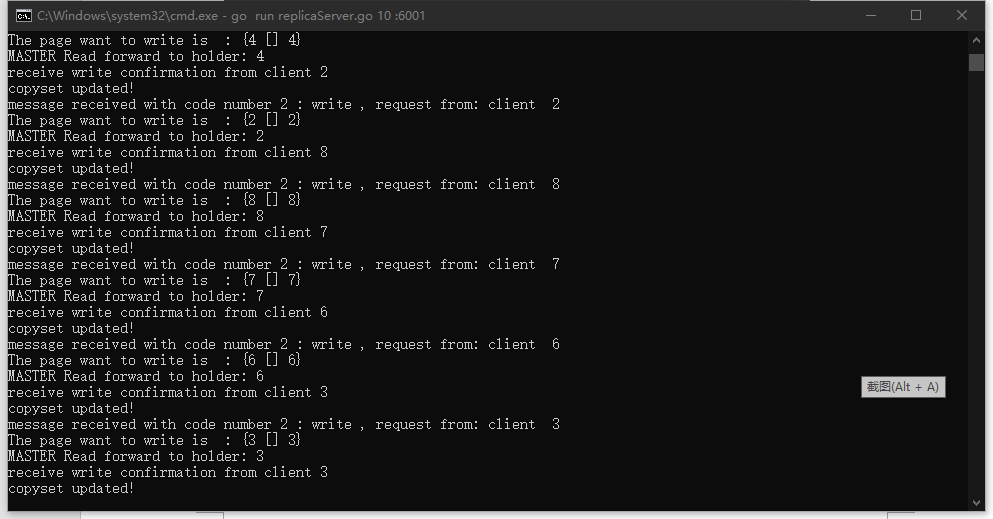
Reason:

There is very little difference

Evaluate the new design in the presence of faults. Specifically, you can simulate two scenarios a) when the primary CM fails at a random time, and b) when the primary CM restarts after the failure. Compare the performance of these two cases with the equivalent scenarios without any CM faults.

First case when the primary CM fails at a random time,

Replica server replace CM when CM fails



Time: 87042456780

Second case when the primary CM restarts after the failure

Time: 87032455310

Reason:

There is no clear difference between the two cases because the in my design, when the master restart, it will join the replica group without affecting the new selected master.

Therefore, the execution order of these two cases is the same, so the time cost is similar.

The time is larger than case without fault because the election algorithm and timeout wait and resend message to replica takes time.