**Programming Assignment 3: Internet of Things - Fault tolerance, Replication, and Consistency**

**TEST CASES DOCUMENT**

SUBMITTED BY: RAHUL RAJ AND OLENKA DEY

---

We inspected our program for different conditions few of the conditions are:

- Testing the Load Balancing is happening as intended or not.
- Check if both the replicas are working properly or not and the data are consistent in the database or not.
- Testing the security system whether it can detect intrusion or not.
- Detecting user home or away based on the motion sensor and door sensor ordering.
- Testing the caching mechanism by giving cache size as 0 and cache size as very large number and check how the system performs.
- Checking the fault tolerance detection as well as how fault recovery is done.
- Checking fault tolerance by crashing both Gateway-1 and Gateway-2.
- Checking the paxos implementation.
- Checking paxos behavior when one of the gateway replica denies to accept the consensus.
- Testing normal activities of the events and ordering of them by inspecting the history file as well as the current state file.

We performed all these test on the system having following event sequence.

```
START
GATEWAY:QUERY BULB STATE
USER:BULB ON
GATEWAY:QUERY BULB STATE
USER:DOOR OPEN PS
GATEWAY:QUERY DOOR STATE
USER:DOOR OPEN PS
GATEWAY:QUERY DOOR STATE
USER:MOTION ACTIVE
USER:MOTION INACTIVE
USER:DOOR OPEN PS
GATEWAY:QUERY DOOR STATE
GATEWAY:QUERY BULB STATE
STOP
```

**TEST-1: LOAD BALANCING**

```
================================================================================================================================
All device registered to gateway - 1 are !!!

Node Name and Node Type are: {'doorSensor': 'Sensor', 'tempSensor': 'Sensor', 'smartOutlet': 'Device'}
Node Name and ID are: {'doorSensor': 4, 'tempSensor': 2, 'smartOutlet': 3}
================================================================================================================================
All device registered to gateway - 2 are!!!

Node Name and Node Type are: {'motionSensor': 'Sensor', 'smartBulb': 'Device'}
Node Name and ID are: {'motionSensor': 6, 'smartBulb': 5}
================================================================================================================================
```

Both the gateways have almost equal number of devices connected and the port number was dynamically selected. The gateway-1 has 3 devices connected and gateway-2 has 1 device connected. This ensures that load balancing is done properly.

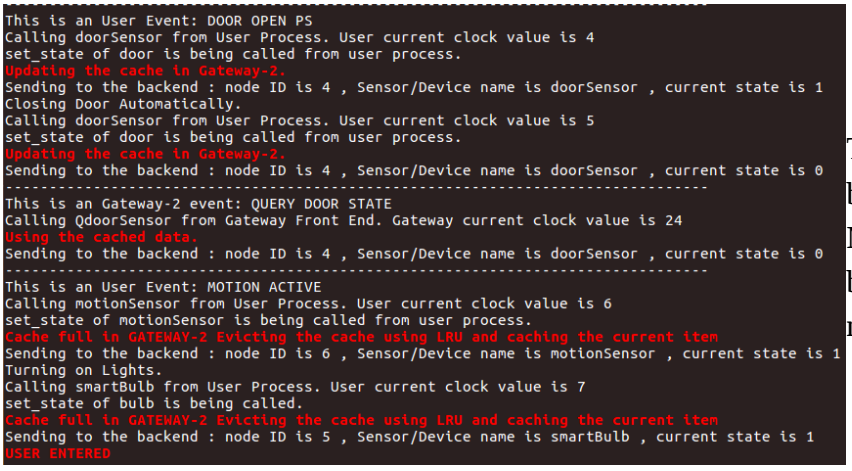
## TEST-2: CONSISTENCY AND REPLICATION

In this test case we check whether the consistency is maintained in both the database or not. We ran the same event sequence and finally the data in both the database should be same.

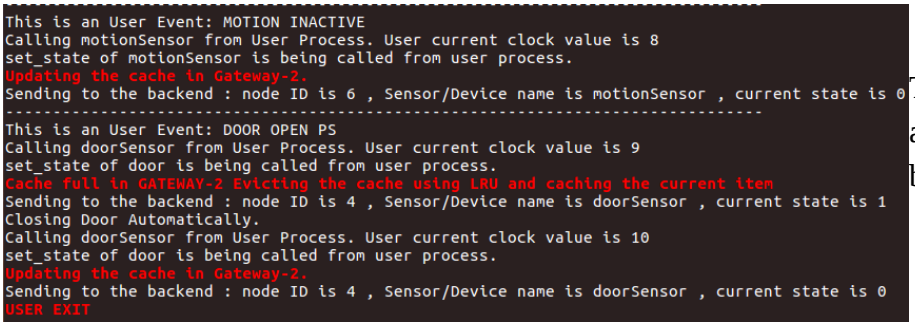

As shown above both the database have same entries indicating that the gateway is replicated properly and consistency is maintained.

## TEST-3: TESTING USER ENTRY AND EXIT BASED ON DATABASE ENTRY

In this test case we check user entry and exit by looking into the database and checking the time stamp of door event and motion event.



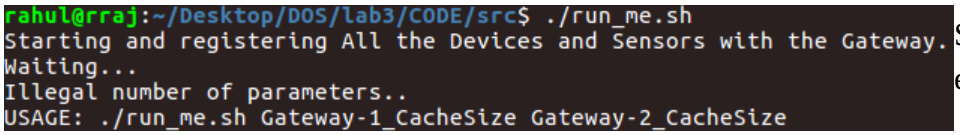

This example on left shows user entry event by checking the order of DOOR OPEN PS and MOTION ACTIVE. It is an ENTRY event because door open has happened before the motion active.



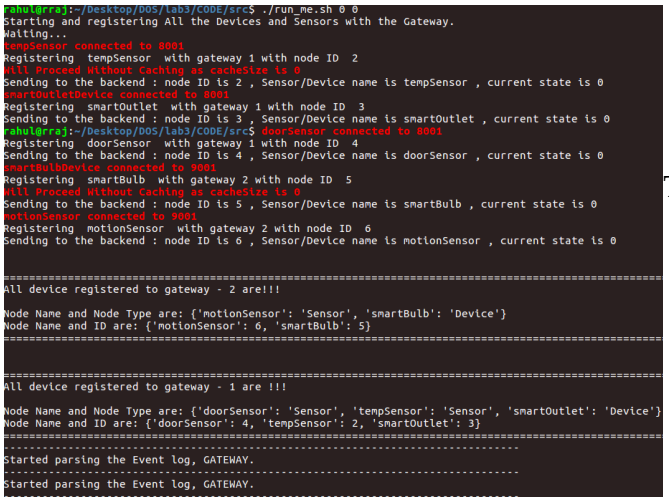

This example shows USER EXIT event as MOTION INACTIVE has happened before DOOR OPEN PS.

## TEST-4: TESTING THE CACHING AND LRU POLICY

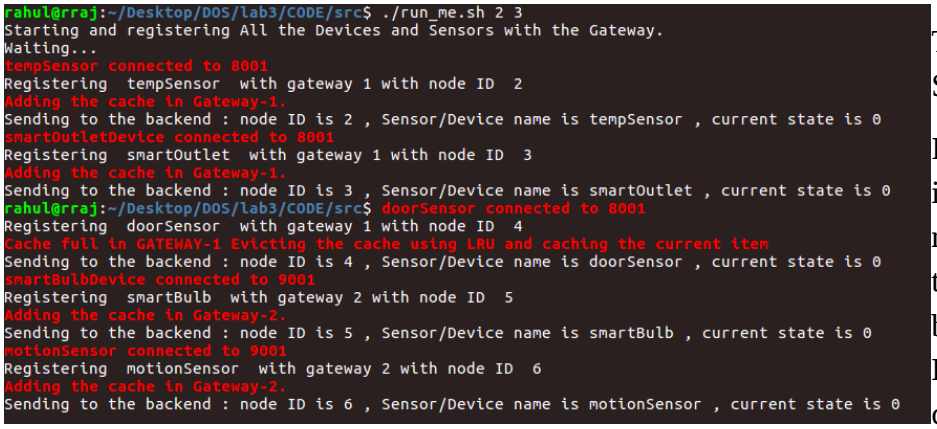
In this test we test the caching as well as LRU policy. First we can test the system by not providing any cache limit at all and see whether that condition is handled or not and further we can pass different cache size as input and observer the behavior of system.




System exits if cache size is not specified explicitly.




Testing with cache Size as 0.




Testing with some random value as cache Size.

It is also clear from this example is that LRU is working properly, because at the time of registration itself when door sensor is trying to get registered it see that the cache is full because the cache size is 2 in Gateway-1 and LRU policy is implemented to evict the cache in order to accommodate door sensor.

## TEST-5: TESTING CACHE HIT AND MISS AND TIME TO COMPLETE THE PROGRAM

For cache size of 0 and 0 for Gateway-1 and Gateway-2 respectively.

```
Total time taken to complete the GATEWAY-2 Program is 11.150799036
Total cache hits are 0 and miss are 0
Total time taken to complete the GATEWAY-1 program is 11.644589901
Total cache hits are 0 and miss are 0
```

For cache size of 1 and 1 for Gateway-1 and Gateway-2 respectively.

```
Total time taken to complete the GATEWAY-2 Program is 10.734459877
Total cache hits are 10 and miss are 1
Total time taken to complete the GATEWAY-1 program is 11.3740828037
Total cache hits are 9 and miss are 11
```

For cache size of 6 and 6 for Gateway-1 and Gateway-2 respectively.

```
Total time taken to complete the GATEWAY-2 Program is 10.695797205
Total cache hits are 11 and miss are 0
Total time taken to complete the GATEWAY-1 program is 11.0526080132
Total cache hits are 15 and miss are 0
```

As it is clear from the above 3 examples that the cache size is directly proportional to cache hit and cache miss number and as the number of cache miss increases the total running time of the program also increases and this is the desired behavior of most of the caching systems.

## TEST-6: TESTING FAILURE DETECTION

In this test cases we check whether the failure is detected by other replica or not. In the example below replica 1 failed after some defined amount of time and the failure is detected by the replica 2 and the failure recovery is initiated.

```
This is an Gateway-2 event: QUERY BULB STATE
Calling smartBulb from Gateway Front End. Gateway current clock value is 10
Using the cached data.
Sending to the backend : node ID is 5 , Sensor/Device name is smartBulb , current state is 1
Devices connected to Failed replica are {'doorSensor': 4, 'tempSensor': 2, 'smartOutlet': 3}
GATEWAY-1 FAILED. INITIATE FAILURE RECOVERY.
```

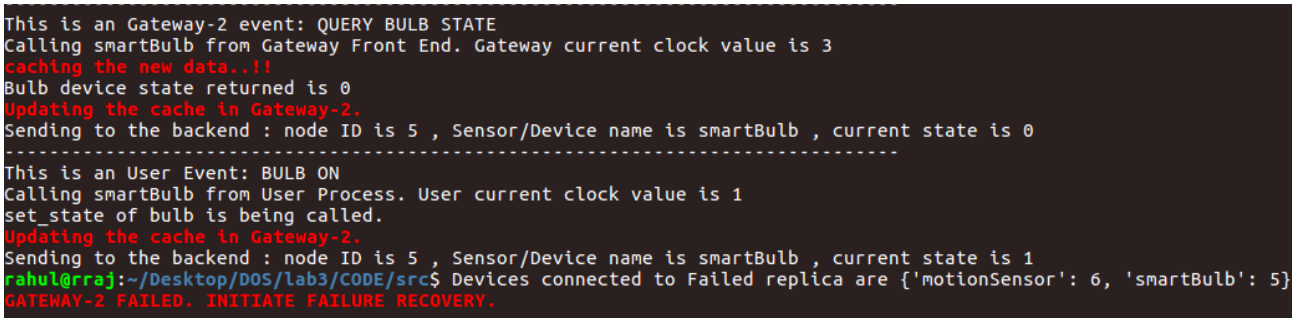
Similarly, if the replica 2 fails it should be detected by replica 1 as shown below.

```
This is an User Event: BULB ON
Calling smartBulb from User Process. User current clock value is 1
set_state of bulb is being called.
Updating the cache in Gateway-2.
Sending to the backend : node ID is 5 , Sensor/Device name is smartBulb , current state is 1
rahul@rraj:~/Desktop/DOS/lab3/CODE/src$ Devices connected to Failed replica are {'motionSensor': 6, 'smartBulb': 5}
GATEWAY-2 FAILED. INITIATE FAILURE RECOVERY.
```

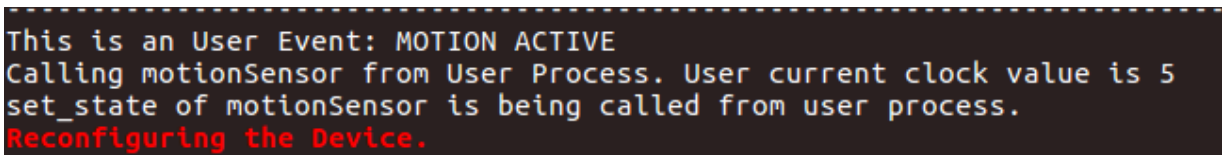
## TEST-7: TESTING FAILURE RECOVERY

In this test case we will look whether the system is able to recover from the failure or not. For example if replica 2 fails then replica 1 should take over its job and complete all the desired task without the loss of data.

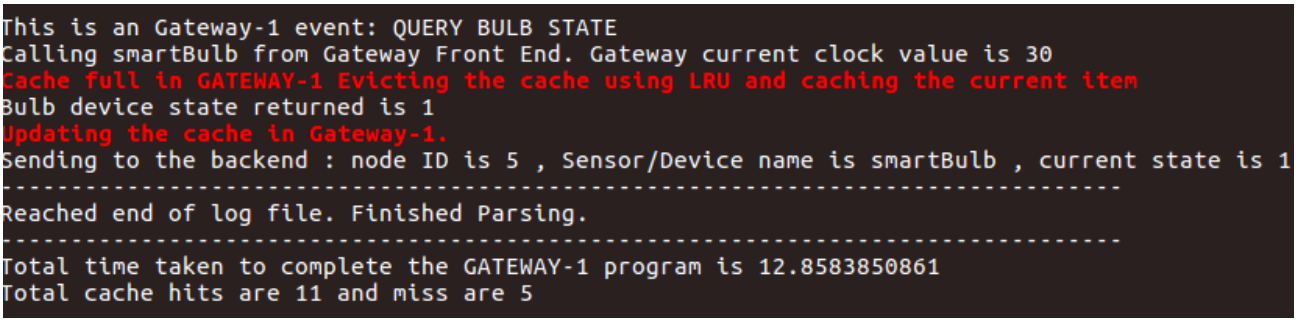
We are achieving this by continuously tracking both the replicas and as soon as one of the replica fails the fault recovery will be initiated by the other replica and all the devices/sensors connected to the failed replica should reconfigure themselves and connect to the now working replica.




At this point GATEWAY-2 FAILED.



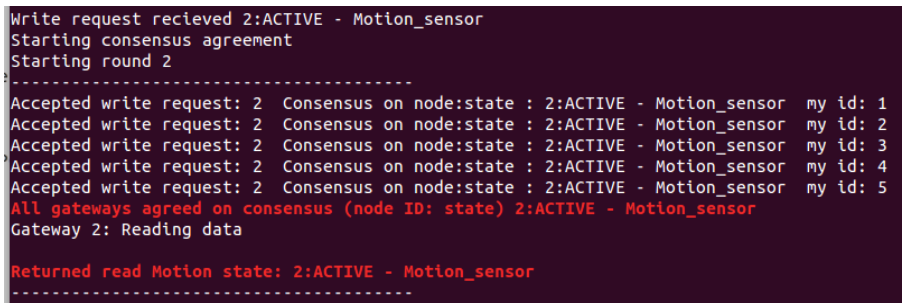

Motion Sensor was part of the GATEWAY-2, so it reconfigured itself at this point of time.




At the end the program exited gracefully with only GATEWAY-1 alive in the end and shows the statistics of GATEWAY-1.

**TEST-8: TESTING PAXOS**

In this, we test if paxos is initiated only on write/change state requests from the sensors/device and not on reads. During reads, the respective gateway serves them.



We also test that all the connected (to sensor/device) and back-up gateway replicas contain same order of events and maintain consistency. Below figure shows the log of two files, both of which are consistent.

```
olenka@olenka-Inspiron-7559:~/Desktop/DOS/PAXO$ cat event_log2.txt
Seq No. 1 Device:State -> 1:68 - Tempsensor
Seq No. 2 Device:State -> 2:ACTIVE - Motion_sensor
Seq No. 3 Device:State -> 3:HIGH INTENSITY - SamrtBulb
Seq No. 4 Device:State -> 2:ACTIVE - Motion_sensor
Seq No. 5 Device:State -> 3:OFF - SamrtBulb
olenka@olenka-Inspiron-7559:~/Desktop/DOS/PAXO$ cat event_log3.txt
Seq No. 1 Device:State -> 1:68 - Tempsensor
Seq No. 2 Device:State -> 2:ACTIVE - Motion_sensor
Seq No. 3 Device:State -> 3:HIGH INTENSITY - SamrtBulb
Seq No. 4 Device:State -> 2:ACTIVE - Motion_sensor
Seq No. 5 Device:State -> 3:OFF - SamrtBulb
olenka@olenka-Inspiron-7559:~/Desktop/DOS/PAXO$
```

**TEST-9: TESTING PAXOS BEHAVIOR WHEN ONE DOESN'T ACCEPT THE CONCENSUS**

We verify the behavior of the paxos algorithm in case one of the replica does not accpet the concensus proposed. In that case, the leader repeats the round and and will repeat until all agrees to one consensus and accepts to log the event data on their database.

```
Waiting to connect
Write request recieved 1:48 - Tempsensor
Starting consensus agreement

Starting round 1
--------------------------------------
Accepted write request: 1  Consensus on node:state : 1:48 - Tempsensor  my id: 1
Round failed !! Starting another round
Round failed !! Starting another round
Accepted write request: 1  Consensus on node:state : 1:48 - Tempsensor  my id: 4
Accepted write request: 1  Consensus on node:state : 1:48 - Tempsensor  my id: 5

Starting round 2
Accepted write request: 2  Consensus on node:state : 1:48 - Tempsensor  my id: 1
Accepted write request: 2  Consensus on node:state : 1:48 - Tempsensor  my id: 2
Round failed !! Starting another round
Accepted write request: 2  Consensus on node:state : 1:48 - Tempsensor  my id: 4
Accepted write request: 2  Consensus on node:state : 1:48 - Tempsensor  my id: 5

Starting round 3
Accepted write request: 3  Consensus on node:state : 1:48 - Tempsensor  my id: 1
Accepted write request: 3  Consensus on node:state : 1:48 - Tempsensor  my id: 2
Accepted write request: 3  Consensus on node:state : 1:48 - Tempsensor  my id: 3
Accepted write request: 3  Consensus on node:state : 1:48 - Tempsensor  my id: 4
Accepted write request: 3  Consensus on node:state : 1:48 - Tempsensor  my id: 5
All gateways agreed on consensus (node ID: state) 1:48 - Tempsensor
```

Clearly, we can see in 1st round, gateway-2 and gateway-3 were not ready, so the leader conducted one more round. In round 2, gateway-2 accepted but gateway-3 was not ready, so again the leader conducted one more round. Finally, in round-3 when all accepted the consensus, the leader closed the agreement.