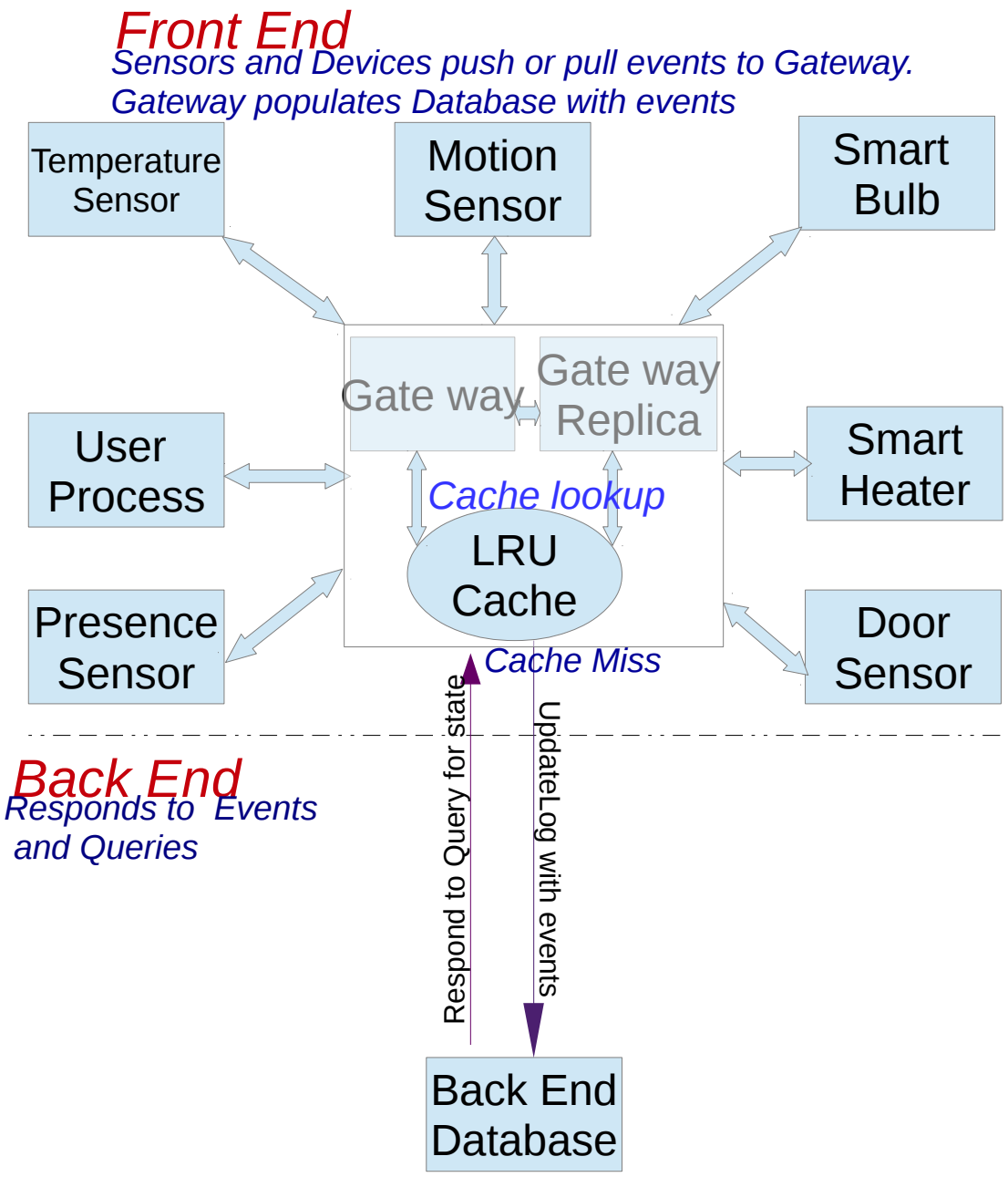
Programming Assignment 3: **Internet of Things – Fault tolerance, Replication, and Consistency**

Fubao Wu 28277607

Ashraf Ali Shaik 28211687

Replication and fault tolerant Architecture

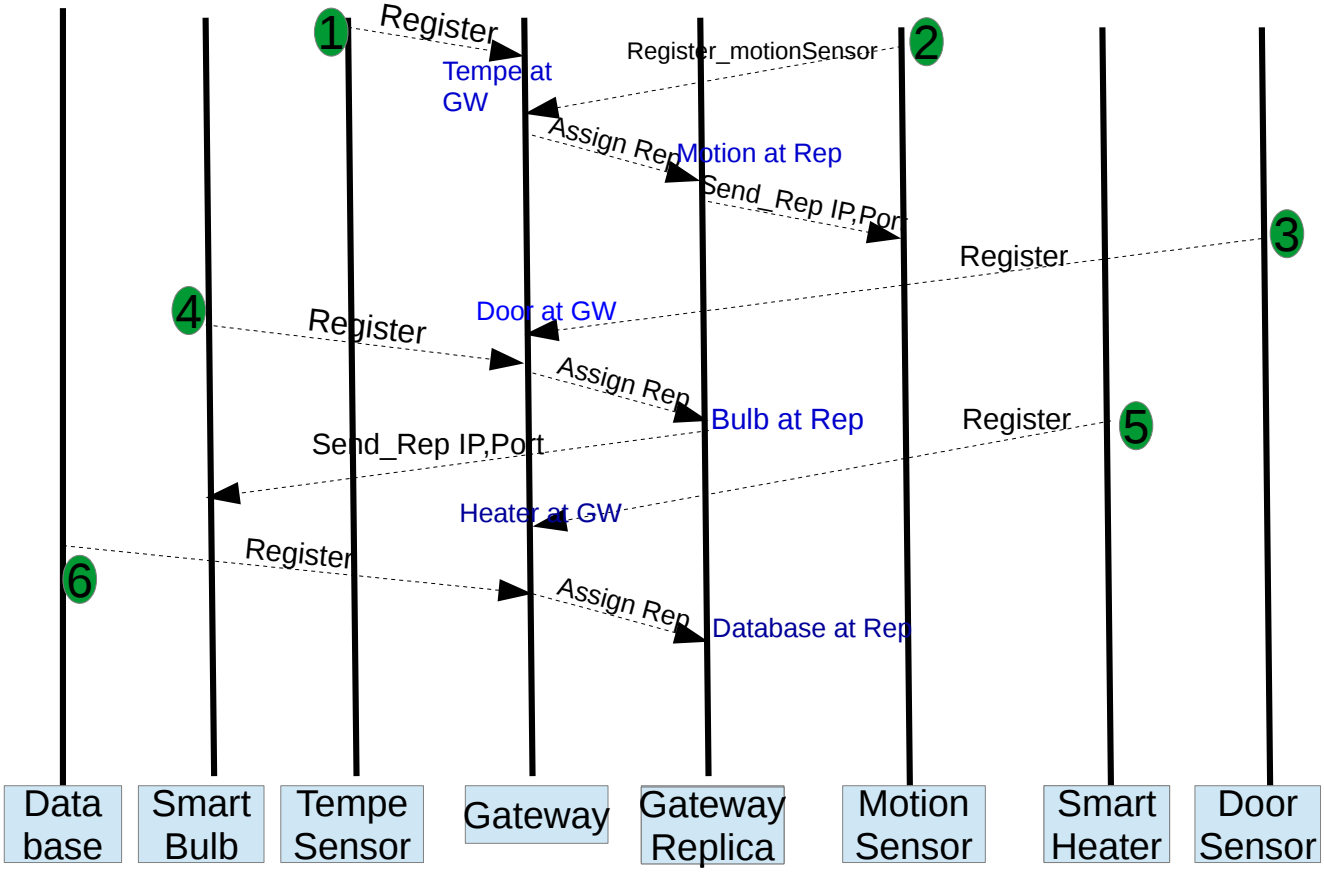
The Gateway is made fault tolerant by adding an replication of data base that can take care of all the devices and sensors when failures occur. Gateway and its replication can also share load between them by sharing the sensors and devices which they can co-ordinate. A Cache is present between database and gateway to improve performance.



Drawing 1: Block Diagram

Load Balancing

Load balancing is achieved by sharing the number of devices and sensors each gate way and replica will handle. Initially all the sensors and devices will talk to the gateway. The gateway takes control of first registered device/sensor and then assigns the control of second registered device/sensor to Gateway replication. Third to gateway, fourth to replication and so on. This ensures load is equally distributed. The devices/sensors are registered at Gateway or Gateway Replication as shown in the figure.



Drawing 2: Load Balancing

Initially Devices/Sensors have Ip address and port details of only Gateway. But after assigning some of them to replication. The Gateway replica will send its ip address and port number to devices/sensors under its co ordination so that they can directly interact with Replication.

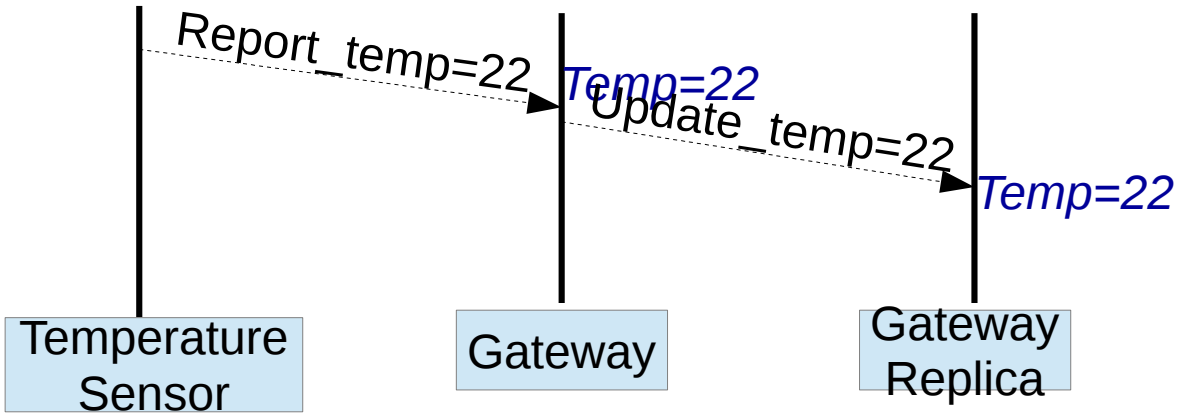
Consistency (Eventual Consistency)

Consistency is achieved by maintaining the state information of sensors and devices at both Gateway and Gateway Replica. When ever a Sensor or device pull or push its state to the gateway or replica we have updated the state in another by a non blocking thread.

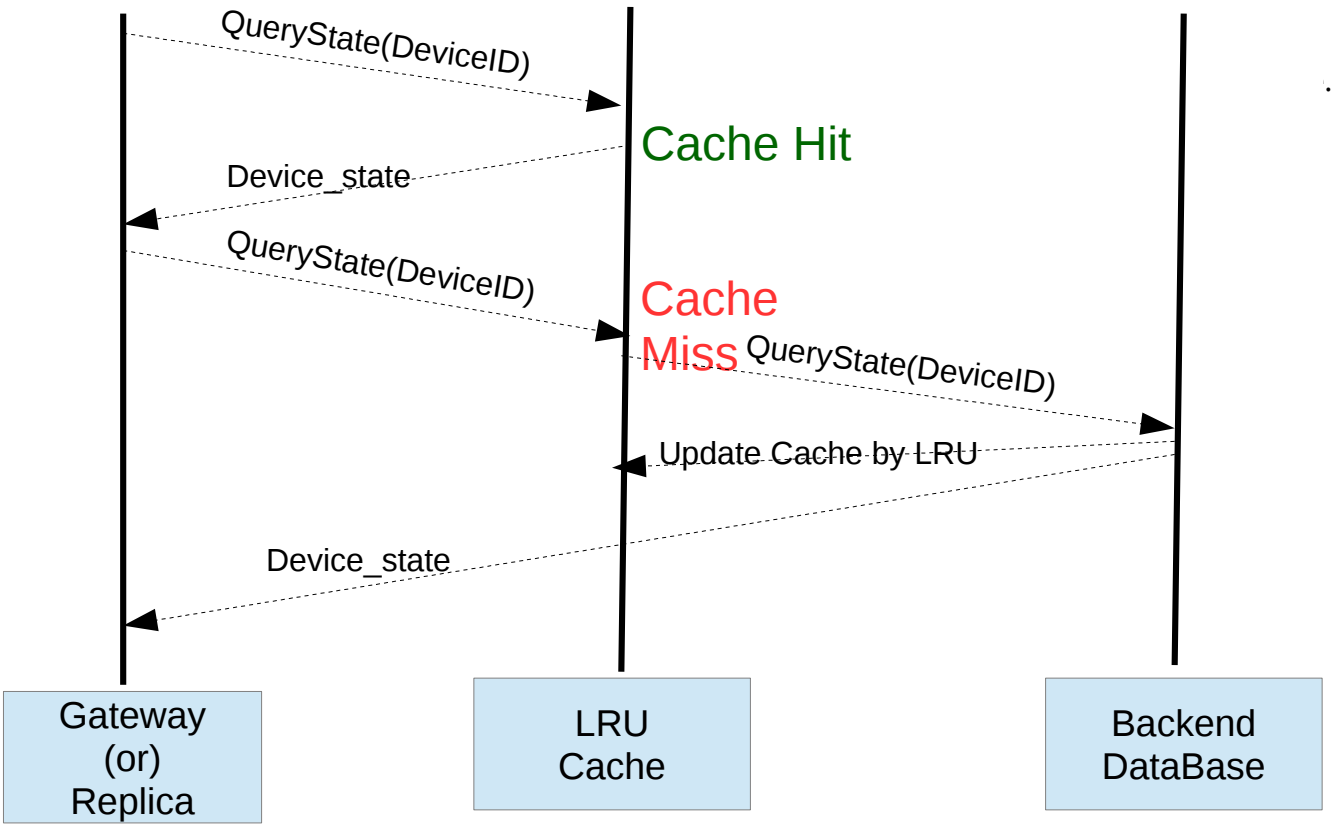
An example test case below explains how consistency is achieved in the design.

We have cache consistency and disk database consistency. When there is write update in one gateway, it would write into the cache and immediately send to its gateway replica to keep consistency. Every replica should be consistent to the last updated values.

For database replication, we have cache written, we would check database whether it is needed to be update or not. If needed, we will write back to database. Write-write conflict is not frequent. If there is mutliple sensors or devices write to different gateway replicas, we will write into different cache using locks.. But we we do consistency, receving message and write it into cache before check whether it is locked or not. Also try to use database file locking or avoid writing database at the same time.



Drawing 3: Consistency Achievement



Drawing 4: LRU Cache Implementation

Fault tolerance:

To make the system operate continuously, the gateway is replicated. But if the gateway is fault, how do we do the detection and recovery.We use event consistency to keep consistent between gateway and its gateway replication called asGw_replica.We will talk about how do we detect the replica. We send the heart message between two replicas.

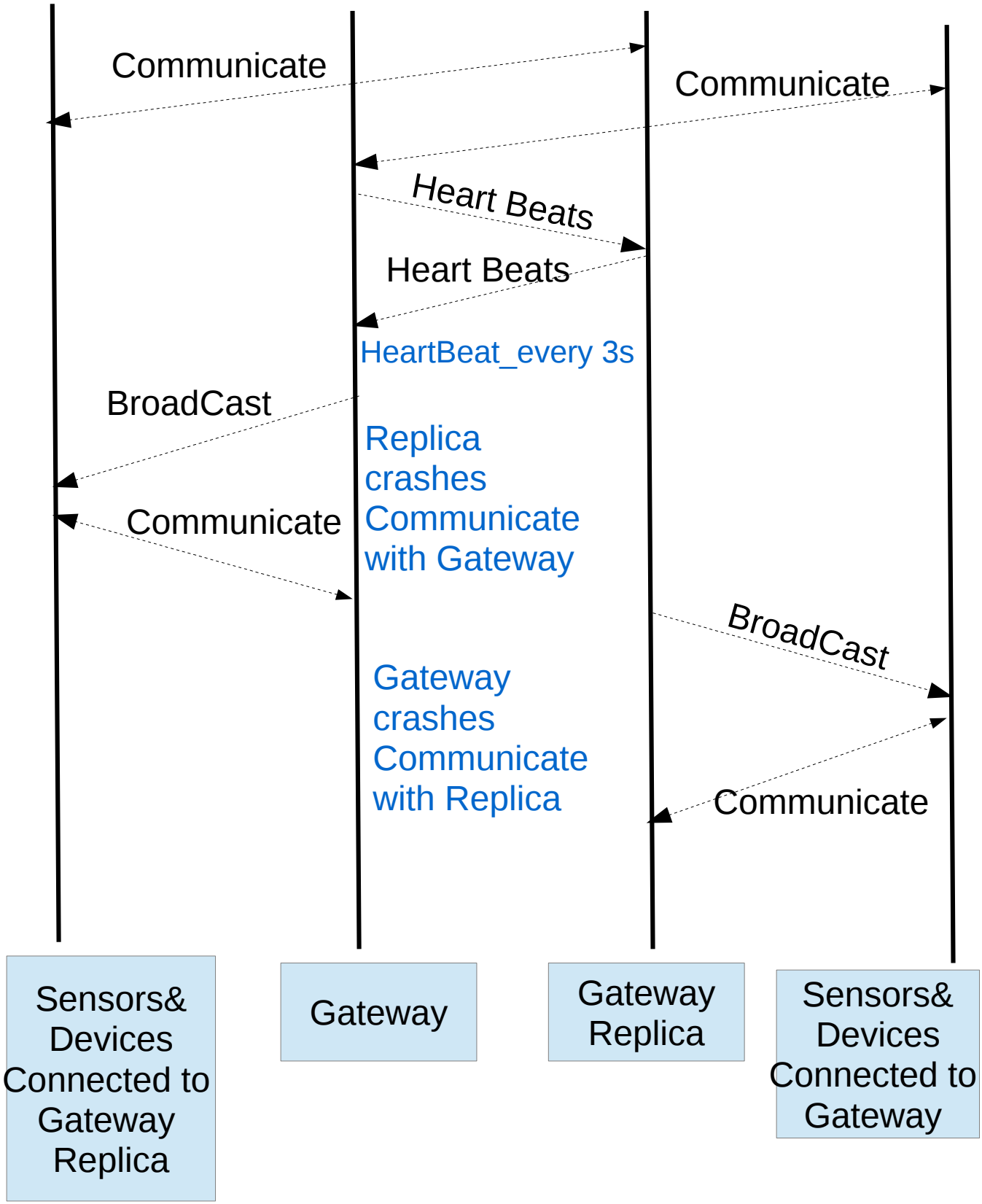
Gateway A push heartbeat message every time_interval(set as 3seconds), Gw_replica A' also sends a heartbeat message every other a time_inteval(set as 3 seconds as well).A and A's will also initialize a separate thread task to detect the pushed message from the other side. The Time interval is also set 3 seconds here.But it will detect for the maximum times(set as 3 here), it reaches 3 times continuously, it will judge the other side is disconnected, we here simply think it is crashes!If one replica crashes, the other would take over the functionality of crashed one.

If Gw_replica is initially registered (connected) with motion sensor, bulb smart. After Gw_replica crashes, these motion sensor and bulb would be notified by the Gateway.Then all the sensors and devices would communicate with the gateway. After the gateway replica recovers, the motion sensor and bulb smart would be notified again. It will be notified again to connect to replica gateway replicas.

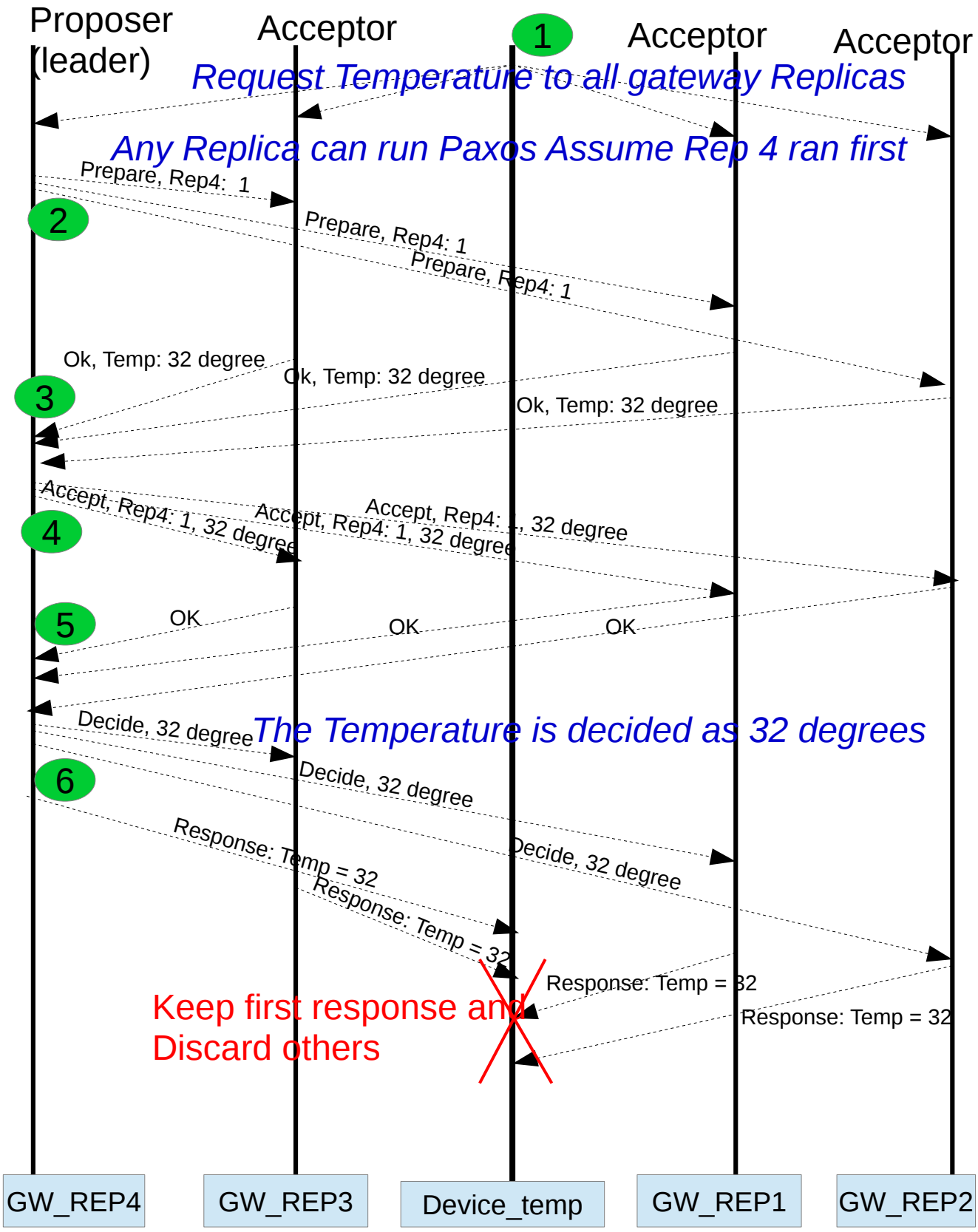
Explain the effect of fault:

Because we use event-consistency which is weak consistency. It doesn't fullfill the stronger constraints. We can basically guarantee exchanging versions or updates of data between gateway. If gateway has updated ,it will immediately send to gateway replica. Or gateway replica has updates, it will also send to gateway to keep consistent immediately.

In our system there are not frequent writing. But if there are multiple sensors write to gateway and gateway replica's at the same time, one gateway replica's value is likely to be lost. The impact of such data loss in our system is not huge. Because if we find data consistency is not consistent, we do query data from sensors to check which one is consisten with the sensor and update the new value with the sensors!



Drawing 5: FaultTolerance operation Timing Diagram



Drawing 6: Paxos Design

Paxos Design details

Assuming there are 4 replicas. As shown in above figure

Step 1: A Device queries for its state to gateway and all replicas.Take Temperature sensor as example and we are querying for temperature.

Step 2: A leader or proposer will be selected randomly. Let the leader in this case be Gateway replica4. It sends prepare message to all the other gateway replicas including the election (agreement) number.

Step 3: If the election (agreement) number is greater than the election number present at the acceptors then acceptors will send the temperature along with OK message.

Step 4:The Leader then find outs the value with majority and broadcasts all the intended results to all other replicas.

Step 5: Acceptors acknowledge with an OK message it they have no concern with the value selected by the proposer.

Step 6: Leader will send the Agreed value to all other replicas. Upon receiving the value all the replicas can transmit the temperature value to the temperature sensor.

Temperature sensor can then only take the first temperature value and discard the next duplicate responses. Temperature value sent by all replicas is redundant but it makes the operation to be almost guaranteed even if one packet of data looses due to network failure. The consecutive responses can be identified and discarded by caching the received value at the sensor end and then matching the identical received packets and making them eligible to discard.

Implementation Platform details:

→ Java

→ RMI

→ Linux Platform supported scripts.

Instructions for running the Code and Test scripts:

The code can be can using two different scripts:

1.run_LoadBalance_Consistency_Caches.sh --- This script is used to demonstrate the implementation of loadBalance,Consistency and Querying for state of a device through Cache.

2.run-part_FT.sh --- This script is used to demonstrate our application is **fault tolerant**. Even if one of the Gateway replicas crashes, the other one will take control.

We have decentralized our code into various Java Packages where each Package corresponds either to a component (the gateway, a sensor or a device or Back end database). In order to avoid the complexity and exceptions due to the supporting code being in different packages, **In addition to the source code we have also provided the executable jar files for each component above described. We have Submitted Source code to verify code and Jar files to execute.**

IP Address Recognition and Allocation:

We are needed to provide only the IP Address of the Gateway in the Configuration file (**configips.csv**). The Default IP Address in configips.csv is local host. This IP Address is needed for all the other components. Each Component can figure out the value of its IP Address when initiated and will register at the Gateway. Gateway stores the IP Address and can access the other components when required.

Jar files & command line arguments in various cases:

For LoadBalance, Consistency, Caching (run_LoadBalance_Consistency_Caches.sh):

The following Jar files will take the command line argument as Path to the Configuration file . **Please place all the Jar files, configips.csv in the same Directory.**

```
gnome-terminal -x sh -c "java -jar GatewayServer.jar configips.csv; bash"&
sleep 3
gnome-terminal -x sh -c "java -jar GatewayServerReplica.jar configips.csv; bash"&
sleep 3
gnome-terminal -x sh -c "java -jar motionSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar tempeSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar DoorSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar HeaterSmart.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar bulbSmart.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar BackendDatabase.jar configips.csv; bash"&
```

For Fault Tolerance (run-part_FT.sh): The following Jar files will take the command line arguments as Path to the Configuration file and “lab3_test”. “lab3_test” is used to distinguish between the above mentioned LoadBalance mode and Fault tolerance mode. **Please place all the Jar files, configips.csv in the same Directory.**

```
gnome-terminal -x sh -c "java -jar GatewayServer.jar configips.csv lab3_test; bash"&
sleep 3
gnome-terminal -x sh -c "java -jar GatewayServerReplica.jar configips.csv lab3_test;
bash"&
sleep 3
gnome-terminal -x sh -c "java -jar motionSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar tempeSensor.jar configips.csv; bash"&
```

```
gnome-terminal -x sh -c "java -jar HeaterSmart.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar bulbSmart.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar DoorSensor.jar configips.csv; bash"&
gnome-terminal -x sh -c "java -jar BackendDatabase.jar configips.csv; bash"&
```

Executable Script files:

run_LoadBalance_Consistency_Caches.sh : This Script files corresponds to the automation of the "*For LoadBalance, Consistency, Caching"* case discussed above. This script is executable and has all permissions. User Just need to run this script and Terminals Pop up.

(i)Load Balance will occur automatically and The nodes which are assigned to Gateway Replica will show which they are assigned . The Gateway and Gateway Replica terminal also displays the registered nodes under them. See Appendix 1 screenshot of obtained results.

(ii)For consistency Testing User need to manually enter and report .For example take temperature sensor and it is assigned to Gateway. Even if we report to Gateway the value will also be updated in Gateway replica. User can see that value is updated in both replicas. User Input Needed. The few starting output in terminal will convey the operation performed by that process. See Appendix 2 screenshots for obtained results.

(iii) For Cache testing User need to follow the instructions on the screen in **Gateway terminal or Gateway Replica terminal** . On entering one of the Integer values of **Device/Sensor Ids** as mentioned in the screen Querying takes place and Initially user will get a cache miss as this is the first entry and we read from database. If user query using the same **Device/Sensor Ids** then a cache hit occurs and we get data from cache. User Input Needed. The few starting output in terminal will convey the operation performed by that process. See Appendix 3 screenshots for obtained results.

run-part_FT.sh : This Script files corresponds to the automation of the "*For Fault Tolerance"* case discussed above. This script is executable and has all permissions. User Just need to run this script and Terminals Pop up. See Appendix 4 screenshots for obtained results.

(i) As terminals pop up the Gateway and Gateway Replica will be sending Heart beats to each other.

(ii)**Enter 1** in either Gateway or Gateway Replica **to crash** one of them. After that user will not see any heartbeats.

(iii)**Enter 0** in the previously crashed Gateway/Gateway Replica **to recover** the Gateway/Gateway Replica. User will see heart beats being exchange again.

Performance Analysis:

We have performed experiments on the performance by measuring the delay from between Gateway and the Sensors.

→ Push Performance delay test which Gateway Received push temperature from Motion sensor.

→ Pull Performance delay test which Gateway pull temperature from temperature sensor

Interval(average taken)	500ms	1000ms	2000ms	5000ms	10000ms
push(report) delay(ms)	3.454	2.898	3.572	3.452	3.126
pull(report) delay(ms)	4.224	3.554	4.232	4.522	4.134

Broadcast delay(ms)	3.756	2.967	3.421	3.653	2.983
Load balancing additional avg delay(ms)	0.457	0.348	0.569	0.317	0.613
Time saved by caching(ms) = (Database AccessTime) - (Cache Access Time)	2.568	3.185	2.157	2.893	3.027

→ From basic statistics of this table, the delay of push and pull are very small where as pull delay is a more than push.

→ An overhead of time due to load balancing is observed while assigning nodes to replicas.

→ Caching have helped in reducing access time than accessing the external file every time.

Possible improvements and extensions to your program

(1) the consistency is not very strict with eventual consistency. If there are frequent multiple sensors write to same gateway and replica, there will be some lost, so we can sequential consistency to improve it to be sequential consistency

(2) the fault tolerant is based on heartbeat. It is difficult to differentiate the long delay and dead. We can improve to accumulate historical experience about the reponse time and approximate estimation response time to decide whether it is long delay or really dead.

Conclusion:

→ Application can run in two modes one for testing one mode tests the implementation of load balancing, consistency and caching. Another for testing Fault tolerance.

→ Paxos conceptual design is illustrated.

→ Performance analysis of various cases were specified based on delay between various communicating nodes.

→ We have extensively tested all the Test cases as mentioned above to ensure the application is executing as expected.

Appendix 1: Load Balance

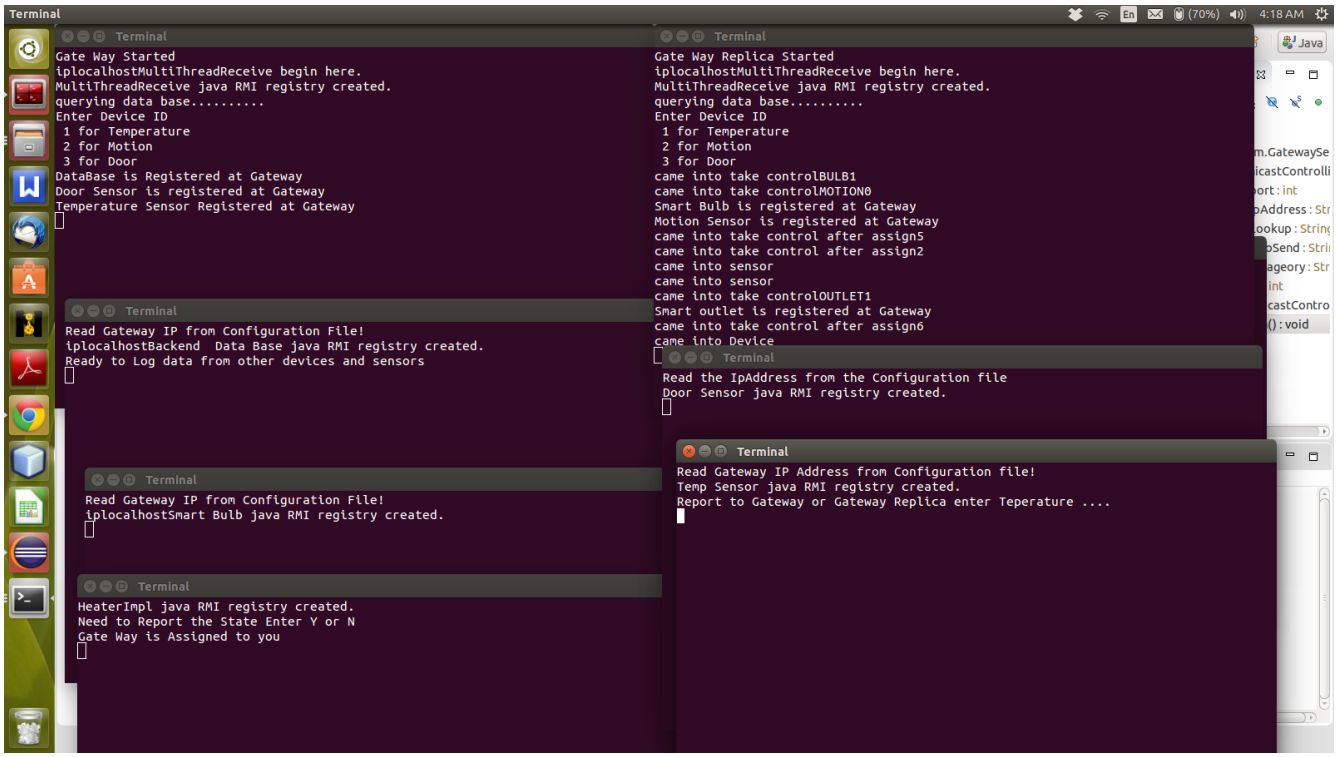


Illustration 1: Load Balancing Executing screenshot

Appendix 2: Consistency

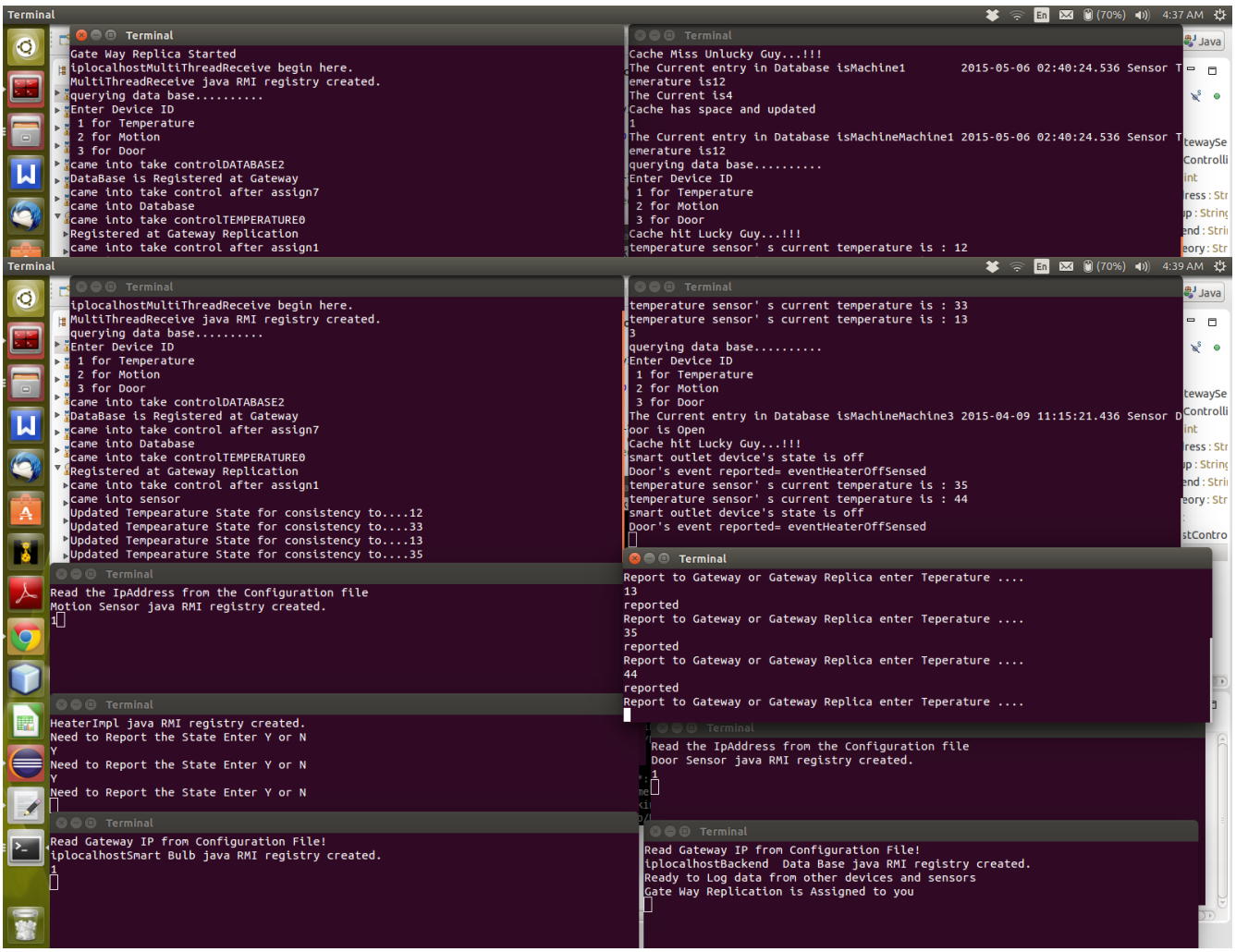
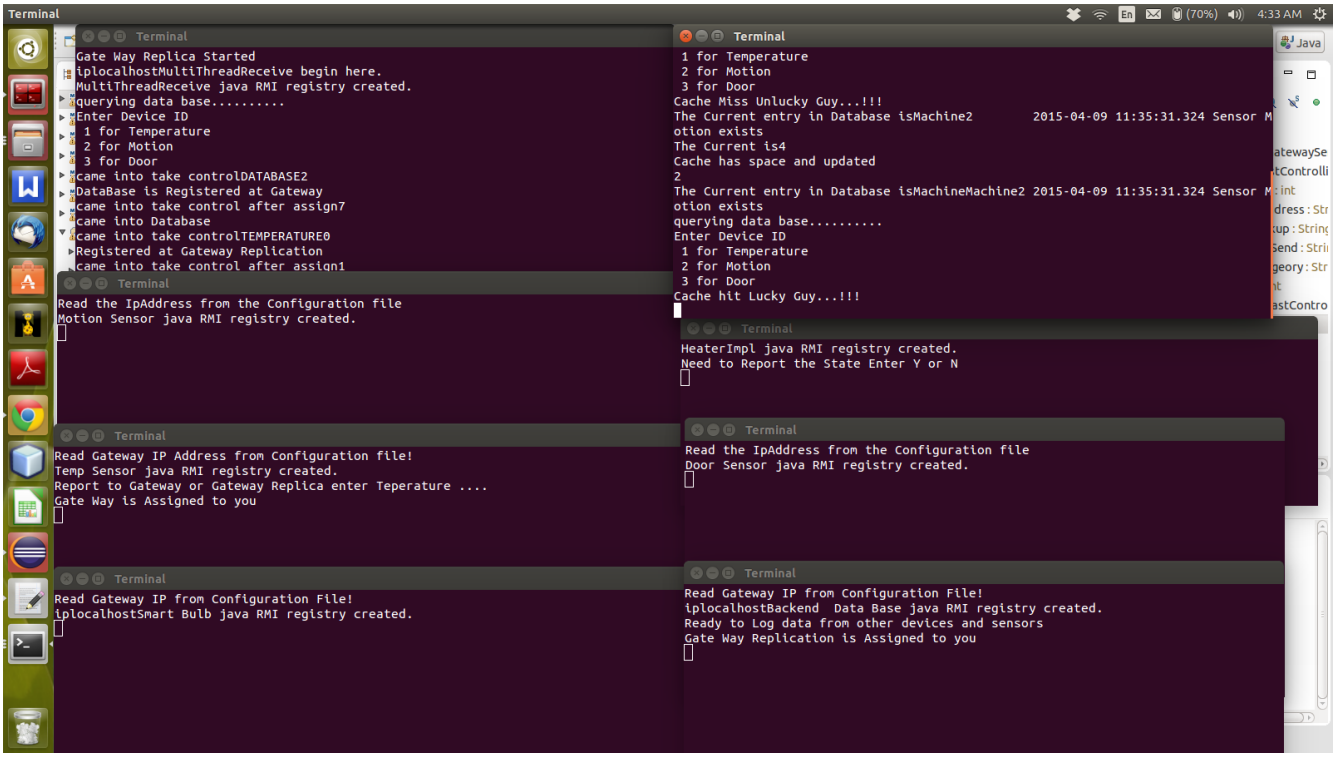


Illustration 2: Consistency Implementation screenshots

Appendix 3: Caching



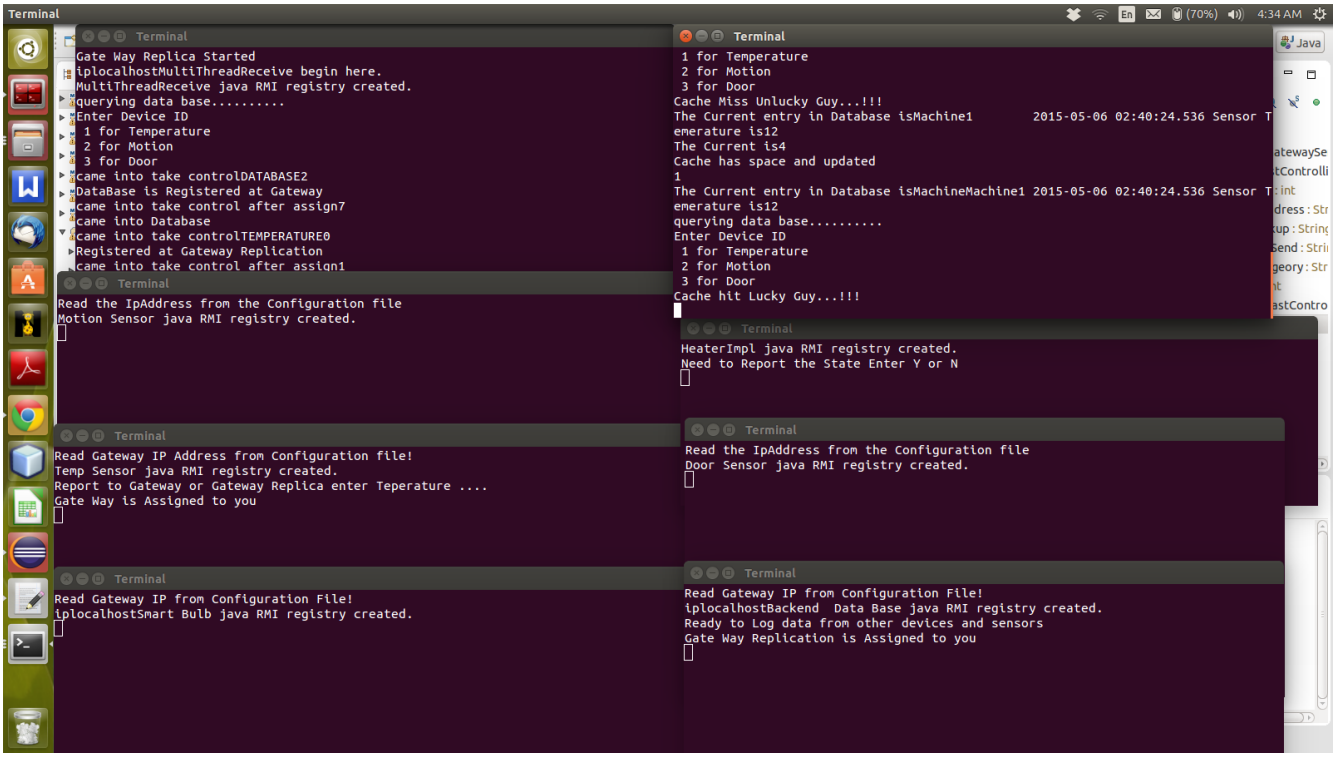


Illustration 3: Caching Implementation Cache Miss and Cache Hit

Appendix 3: Fault Tolerance

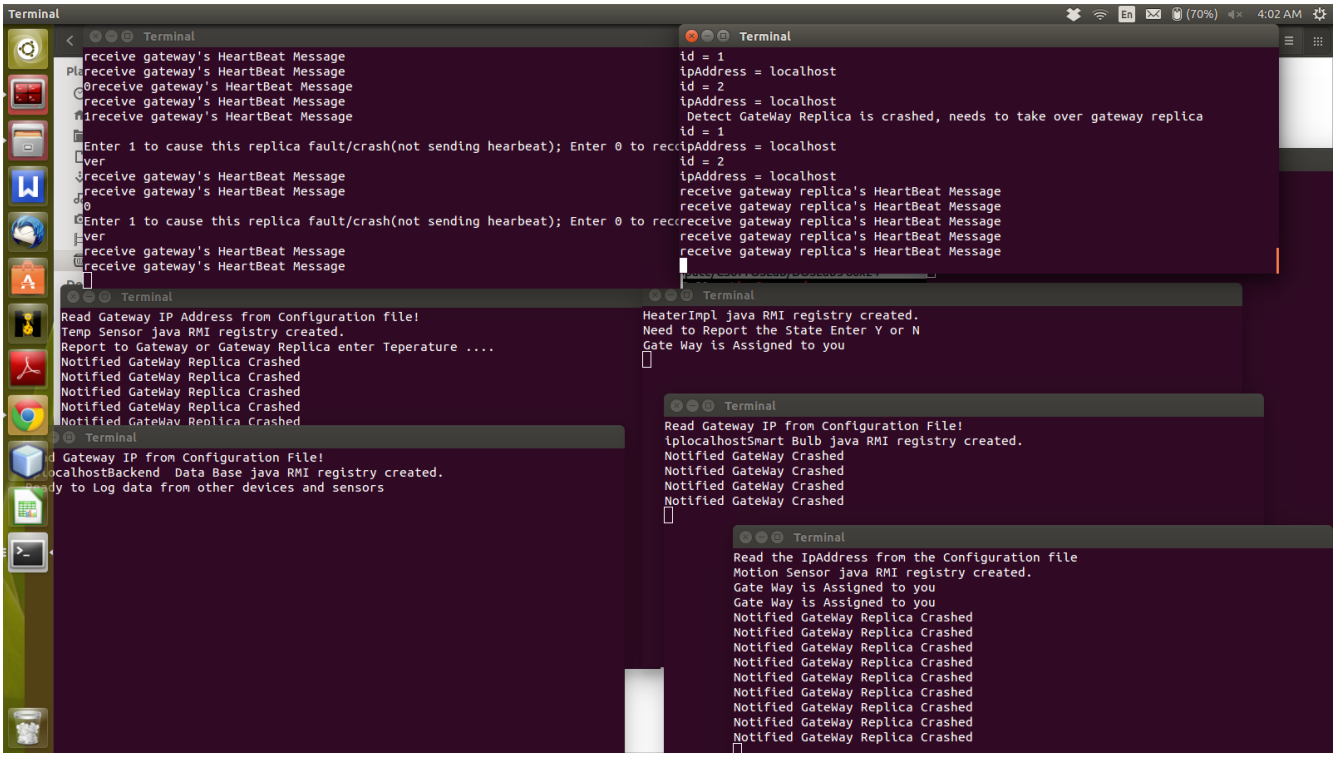


Illustration 4: Gateway Failure Detection

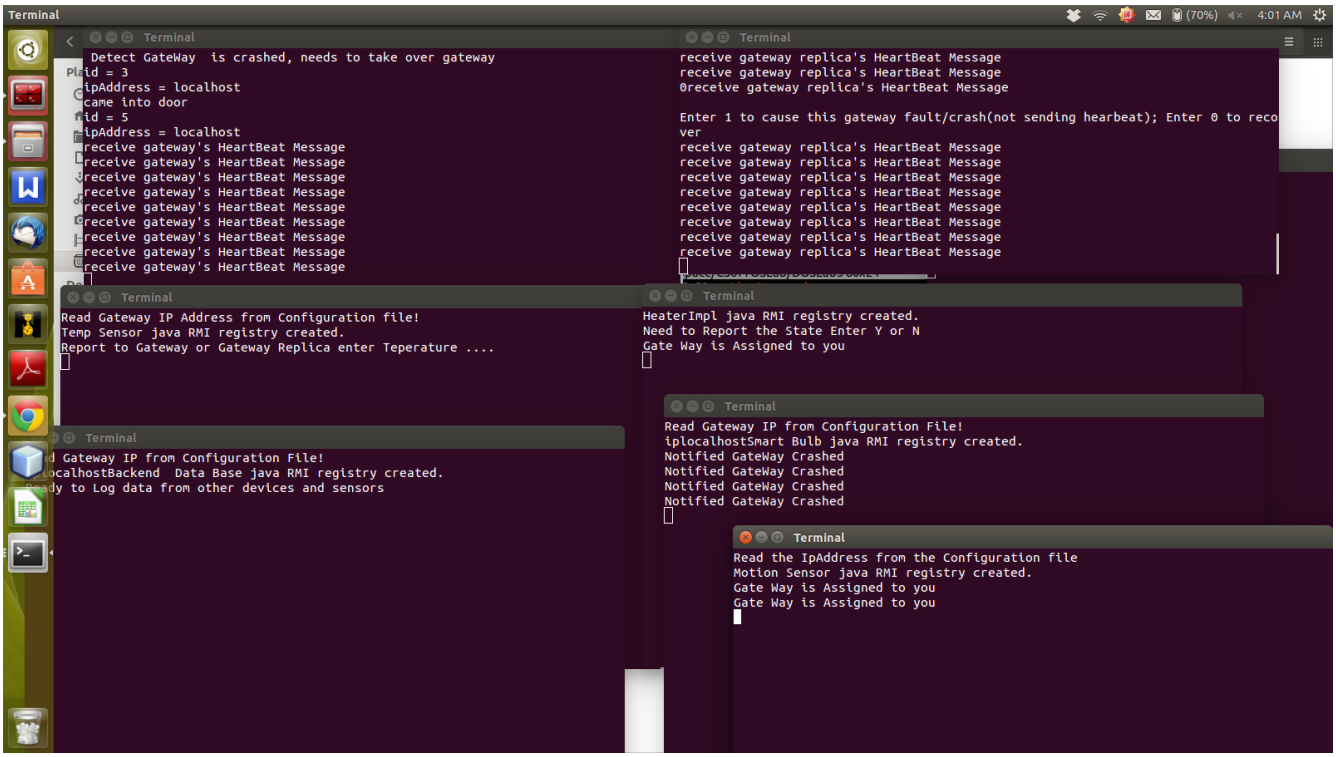


Illustration 5: Gateway Replication Failure Detection

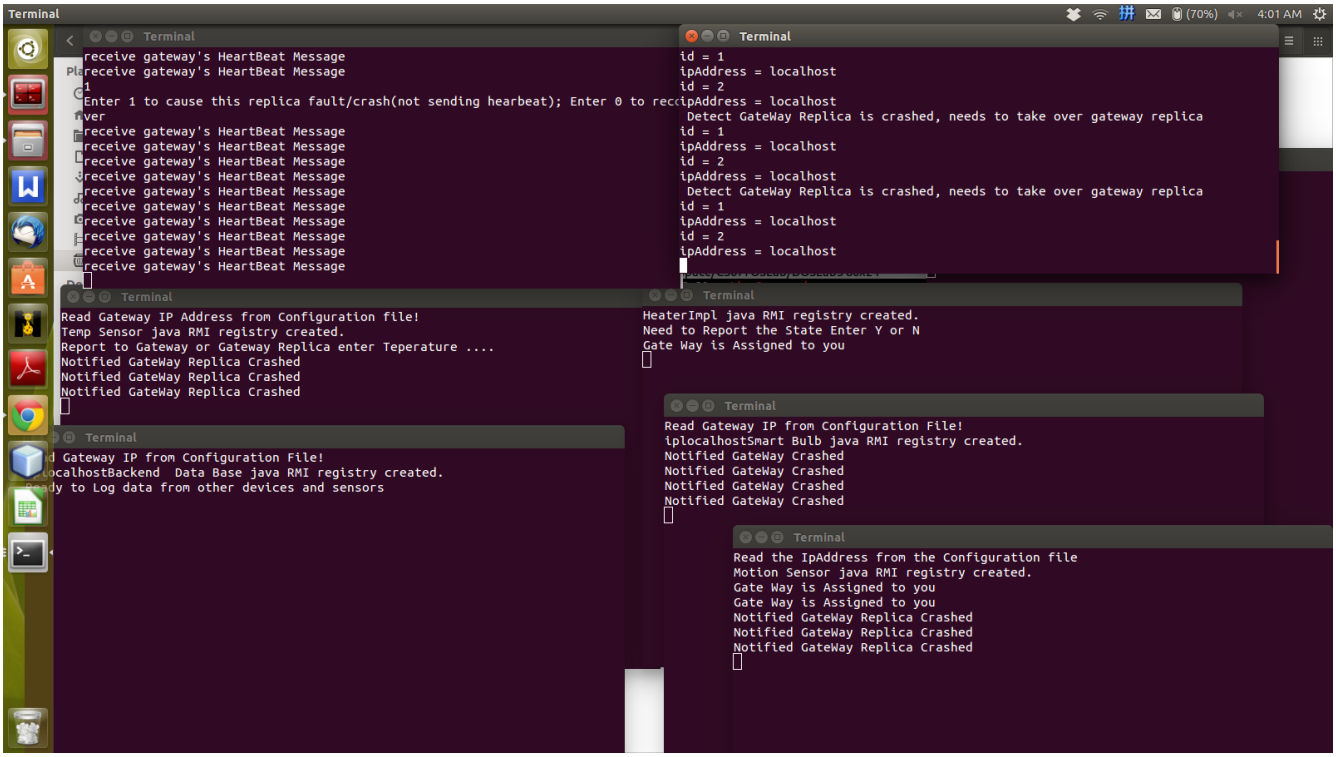


Illustration 6: Gateway Replication Recovery

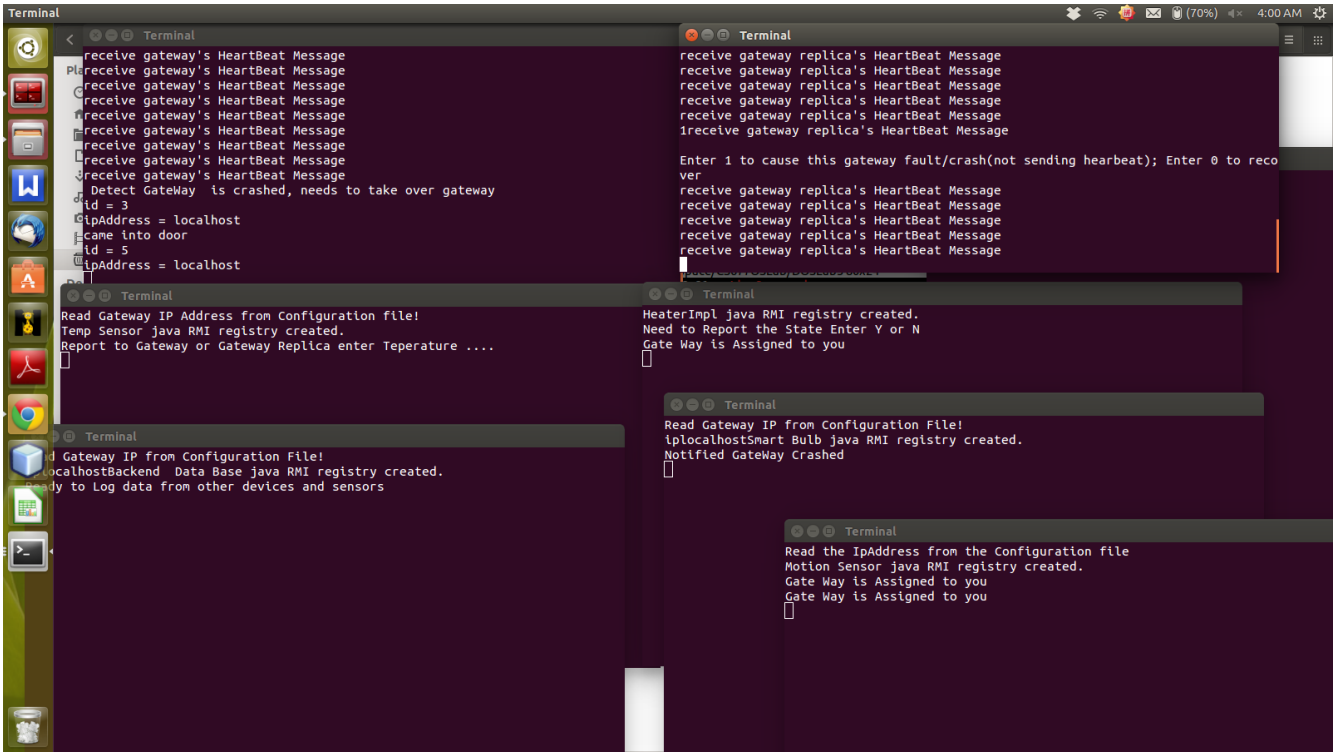


Illustration 7: Heart Beat Messages