Internet of Things - Smart Home Edition

Noman Bashir, Shubham Mukherjee

April 23, 2017

1 Latency Results

We measure the latency of different actions performed by the system. All the results are presented below:
Cache Results: The first one is the time taken to process a request with and without cache. We evaluate this by taking a sample user activity and measuring the total time taken by the sequence of actions with and without cache. For example, in order to process the user activity profile 1 the times taken are:

- Time taken with caching: 11.7785770893
- Time taken without caching: 12.8137729168

There are total of three cache misses without cache and the total time taken by them all is 1.035193 second which results in an average of 0.345 second per cache miss.
Failure Results: We configure the wait time for the heartbeat as the heartbeat period + extra time to account for the message delays.

- Time taken to detect failure: heartbeat period + extra time + arbitrary time. The evaluator can configure these to any desired values. However, extra time should be higher than the typical message propagation delays which is 10ms. If heartbeat period is 5second, extra time is 1seconds, teh total time to detect failure is around 8second.
- Time taken taken to recover from a failure: detecting failure time plus extra time of 5second. For the values given above, it takes around 14seconds on average to recover from the crash.

2 Test Case 1

This test case aims to show that our mechanism dynamically balances the load across the gateways.

- We run the registerProcesses.py script. It asks the user to input the number of devices to be registered between (1-6).

- After user enters the no. of devices, the script registers the devices with the two gateways. The device ID and the gateway with which it is registered is printed.
- It can be seen from the command line output that the devices are evenly divided across gateways. However, in case of odd number of devices, one gateways gets one device more than the other.
- the results can be further verified by running the Pyro4-nsc list for both the servers.

3 Test Case 2

The purpose of this test case is to demonstrate the implementation of consistency mechanism. In order to perform this test, the evaluator needs to uncomment the **test_consistency** test case in run.py.

- After completing the registration process, the user performs some actions on different devices distributed across both gateways.
- We then retrieve all the values from the database and print them to show that the state of databases in both cases is same.

4 Test Case 3

This test case targets to demonstrate the working of caching mechanism in our system. For this test case, we request the evaluators to change the value of **constant *cacheSize* in config filec**. They should change the cache size from 0 to 3. We provide the reference output for all the cache cases. You can enable any one test available in the run.py file and evaluate its performance with different cache sizes.

- When the cache size is set to 0, all of the requests for database will result in cache miss as there won't be any value stored in the cache.
- In case of cache size 1, there will be some cache hits and some misses. This will happen as we need to last two entries for event order logic in some cases.
- For cache size 2 or greater than 2, every request for database will result in a cache hit. This is because our event order logic requires only last two entries.

5 Test Case 4

This test case aims to evaluate the fault tolerance mechanism of our project. In order to evaluate this test case, the evaluators need to crash one name server.

- The evaluator can enable any of the three user activity test profiles in the run.py file.
- The evaluator will have to crash one name server at any point in time they desire.

- The devices will register themselves with the other gateway after some time.
- Sample output is provided in the test directory.