FOG NODE IMPLEMENTATION

PROJECT REPORT

Team Members

1. Apoorva Jayaram
2. Manisha Arumugam
3. Vincy Shrine

High level Implementation Details

The fog node does the following:

- A TCP server thread is spawned. to receive and handle incoming offload packets and response time updates from neighboring fog nodes
- A UDP receiver thread is spawned, to receive and handle incoming IoT requests.
- The above threads make sure that either the fog node processes the request packets or offloads to best neighbor.
- A Fog Consumer thread is spawned, to serve the IoT requests that were placed in the fog queue.
- A Cloud Consumer thread is spawned, to serve the IoT requests that were placed in the Cloud queue.
- A TimerTask thread is spawned with an initial delay of 10 seconds (to allow all fog Nodes to come up), to send periodic response time update about its own queue to neighbors.
- Once a packet is served, a UDP response is sent back to the originating IOT node.

Issues faced

1. Choosing the best neighbor - resolving tie when 2 fogs have same processing time. Priority Queue Comparator was updated

2. Data sent as part of the response time update was huge. Hence we made unnecessary fields (used for internal purposes) as transient, so that they are not serializable and not persisted. This reduces the number of bytes sent across.

3. Redundant TCP client connections opened and closed to neighbors, every 10 secs. We tried to implement connection pool

Individual Contributions

Apoorva Jayaram

- Coded and implemented the TCP listener and TCP sender to handle incoming fog node offload and periodic updates.
- Design and planning of the basic algorithm.
- Testing and debugging for scalability with multiple fog nodes and IOT nodes.
- Worked on handling redundant TCP connections using reusable connection pool using hash map.

- Worked on the implementation of configuration reader for fog nodes.

Vincy Shrine

- Designed flow charts as part of project design phase
- Coded the base framework code
- Coded and implemented UDP listener to parse and handle incoming IOT request packets
- Coded and implemented periodic queue processing time update to neighbors using Timer Task Java API
- Coded and implemented fog and cloud processing mechanism in producer/consumer fashion using Java API of ArrayBlockingQueue
- Coded and implemented a two-level custom comparator based on processing time and fog Id, to select the best neighbor from priority queue, as part of the offload process.
- Added debug and error logs for fog node program execution
- Added logging details for IOT packet content
- Tested and bug fixed the code at every stage

Manisha Arumugam

- Involved in the basic planning and understanding the flow of the project
- Designed flowcharts for two modules as part of the initial project design report
- Involved in the implementation of framework code
- Tested using different test cases
- Designed the final project report

Test output – a sample execution

An IOT packet after being served by a fog node has contents as below:

```
RECEIVED: Packet [id=dc09.utdallas.edu:7000, T=1, PT=1000, FL=4,
seq#: 73, path=[1, 2, 3, 4], details=
Visited 1; available PT: 0; cannot process; FL: 0; neigh PQ: [2:40000]; offload to: 2;
Visited 2; available PT: 0; cannot process; FL: 1; neigh PQ: [1:40000, 3:40000]; 1 visited; offload to: 3;
Visited 3; available PT: 0; cannot process; FL: 2; neigh PQ: [4:-44000, 5:-15000, 2:40000]; offload to: 4;
Visited 4; available PT: 41000; can process; Served by fog;]
```

Below are the testing logs of IOT packets with the given topology consisting of 5 fog Nodes and 2 IOT nodes.


iot1_log.txt


iot2_log.txt

Attachments

Source code

project 07Dec16.zip

High Level Design Document

ACN Project - HLD.pdf