

## Idea

- An edge data center should be able to elect a leader among the other edge data centers in reach (if they meet some arbitrary conditions).
- In my implementation a leader can't be an orchestrator. This can be easily changed.
- A leader (that can have a leader as well) can execute tasks on its VM or offload them to its subordinates.
- As last chance the task is offloaded on the Cloud.

## Implementation

### LeaderEdgeDevice class

I started by creating a custom class (LeaderEdgeDevice) that extends DefaultDataCenter using the proposed ClusterEdgeDevice as a base.

In startInternal a task with tag LEADER\_ELECTION is scheduled with  $\text{INITIALIZATION\_TIME} + 1$  delay.

In process event the custom tag is caught.

There is a check for:

- the device being an edge data center,
- the device being an orchestrator. This allows data centers not flagged as orchestrators to not have a leader (although possibly being elected as such by other data centers, even that case is easily fixable).
- "LEADER" is the orchestration method,

If everything is true then the method **leader** is called.

- in a loop every datacenter is taken into account and is checked whether it:
  - is not the same data center as the one evaluating,
  - is an edge data center
  - the distance between the two data centers is smaller than the range of the edge data centers.
- If it is the case then there is an evaluation about the MIPS (I used that criterion for electing a leader).

- If the MIPS of the candidate are greater than ones of the evaluator
- and the max MIPS of the data center seen until that point is lower than the MIPS of the candidate

then the candidate becomes the (potentially temporary) leader, the max MIPS seen is set to the new leader's MIPS.

Upon ending the loop if a leader has been found the current datacenter is added to its subordinates list.

Otherwise the orchestrator flag is removed as well as the data center presence in Orchestrators List.

## Leader algorithm

In order to find a suitable VM using this hierarchy I created a simulation algorithm called LEADER.

I extended Orchestrator in order to implement a custom findVM method.

If LEADER is used as simulation algorithm there is a check for the - presence of both Cloud and Edge in the simulation architecture. If it is not the case an error is displayed and the simulation stopped.

Otherwise the architecture is restricted to "Edge" and the method leader is called.

Here this order of seek for a suitable VM is followed:

1. Among the hosts of the orchestrator.
2. Among the hosts of the leader of the orchestrator (there should not be a situation where the orchestrator has no leader and is chosen for the previous step so the presence of a leader should be certain at this phase).
3. Among the hosts of the leader's subordinates.

If the previous steps should result in no VM selected a try is done with the algorithm INCREASE\_LIFETIME (could have been anything else) on the Cloud.

The condition based on which every level is scanned in search for a VM is completely arbitrary.

## Limitations

### Chaining

A case of chaining can occur when the data center A has B as leader but B in turn has C as leader.

Both A and B are orchestrator so they can both receive tasks.

During findVM, if a task has A as orchestrator, the task can only be offloaded to B, its subordinates (excluding A) and the Cloud.

This doesn't allow B to offload the original task to C, in fact B act only as leader in this scenario.