# INFO8002: Programming project

**Miss Gómez Herrera María Andrea Liliana**[1] and **Mr Sebati Ilias**[2]
[1] *mariaandrealiliana.gomezherrera@student.uliege.be (s198387)*
[2] *ilias.sebati@student.uliege.be (s181506)*

## I. PROBLEM STATEMENT

This project aims to make us implement a consensus between several flight computers. We have to handle several computers which may be deficient. Hence, they may crash. Our algorithm must be fault-tolerant and the consensus must converge at a good rate (see **DISCUSSION**).

We decided to use the `without-ksp` program. See the **README.md** file to know how to start the program.

## II. CONTRIBUTIONS AND RESULTS

We divided this project in two parts:

- Networking one, it was essentially making each computer as a process. We used flask.
- Leader election, we used the raft leader election.

Since the main consensus is in the Leader election, it was a long part. We devided the project equally.

The results are pretty good, we are printing in each computer an indication:

- `heartbeat`: The computers that receive heartbeat from a leader. Thus they are follower.
- `I am leader`: The computer that is leader and that sends heartbeat to all other computers.

To test the leader election we can kill a computer (e.g: Ctrl+c) and we will see that another computer becomes leader. However, if a strict majority of computers are down, nobody will become the leader.

## III. DISCUSSION

### A. Leader election

The leader election is inspired by the raft consensus algorithm. The election is pretty simple. Each computer keeps a counter called `term` that represents the current epoch. Each computer starts a **timeout** with a random time. The fact that the time is random is useful because it will reduce the probability that two computers have the same timeout and begin the election at the same time (which can result in an election where nobody win). As we said, the timer is set randomly between two ranges (100-200ms by default), but this range changes depending on the number of election where no computers became leader (higher the number of election is, higher the range will be).When the timeout occurs for a computer, it tries to become leader by starting a new election with a new `term`. Three scenarios can happen:

- The computer won the election. In this case, it starts sending acknowledge to all others computers to prevent them to start a new election.
- The computer lost and someone else won. It becomes follower and reset its timer.
- Nobody won, the computer waits that the timeout previously set ends to restart a new election.

Each time that a computer (which is not leader) receives a heartbeat signal or any time of requests from the leader, the computer set the timer (randomly) again until the timer reaches 0. In this case, it starts a new election.

If a computer that is leader crashes and then recover after a while, this computer will start acting like the leader but all other computers will tell him that it is no longer the leader. Thanks to the heartbeat sent by the true leader the computer will be able to update his view of the 'world'.

This is an overview of what is happening, to see more refer to the **raft** paper.

### B. Assumptions

We only made two assumptions.

The first one is that a computer (leader of follower) cannot modify the state that it received from the client. Hence, all **states** are acceptable.

The second one is that we cannot have a strict majority of computers that are either `FullThrottleFlightComputer` or `RandomThrottleFlightComputer`. The reason is simple, both computers modify a field in `action`, so all corrects computers will never accept their action, and those computers (random and full) will never accept the actions of the correct computers. However, having a strict majority of **only** `FullThrottleFlightComputer` is acceptable in the sense that they will reach consensus between themselves. Unfortunately, they will change the action of the rocket.

### C. Convergence rate

The convergence rate depends on the `correct-fraction` of computers. Let us first talk of the convergence rate with all computers that are correct. In this case, a leader will be elected and this leader will remain the same for the entire life of the program. The leader will send requests to all other computers to ask them about some states or actions. But as long as the leader does not crash or does not have a problem in sending the heartbeat to other computers, it will remain the leader.
Now what if a strict majority of computers are correct but a minority are incorrect? If the first leader elected is part of the correct computers, we will have no problems. It will still send the acknowledge and all computers (even the incorrect one) will let him be leader. If a incorrect computer crashes or timeout quicker than expected (because of some software/hardware problems) the incorrect computer will try to become leader. Now, if we have a leader that is an incorrect computer, this computer will be leader until it crashes or it is too slow to send the heartbeat. Other computers will then try to become the leader. When a leader receives a request from the client, the leader will send to all other computers a request for them to accept (or not) the client's request (e.g. acceptable_state). The client does not know (and does not care) about which one is the leader. The client send his requests to a random computer. Here it is random but in practice it can be the closest computer from the client or other criterion. When a computer receives the request, if it is the leader it will handle it (see above) otherwise, it will send the request to the leader and respond to the client after that the leader handled it.

If the number of incorrect computers is a strict majority, we may have a lot of re-election and the computers will probably take a long time before answering the leader's requests. The consensus will be quite slow.

### D. What about slow (correct) flight computers

Slow flight computers will more likely never be leader since they are slow to take decisions (such as starting a new election). Even if a slow computer becomes leader, it will probably take too much time to send the heartbeat and be replaced by a faster computer.