Presentation of IOT Satellite case study

Students:

Daniele Semeraro

Professor:

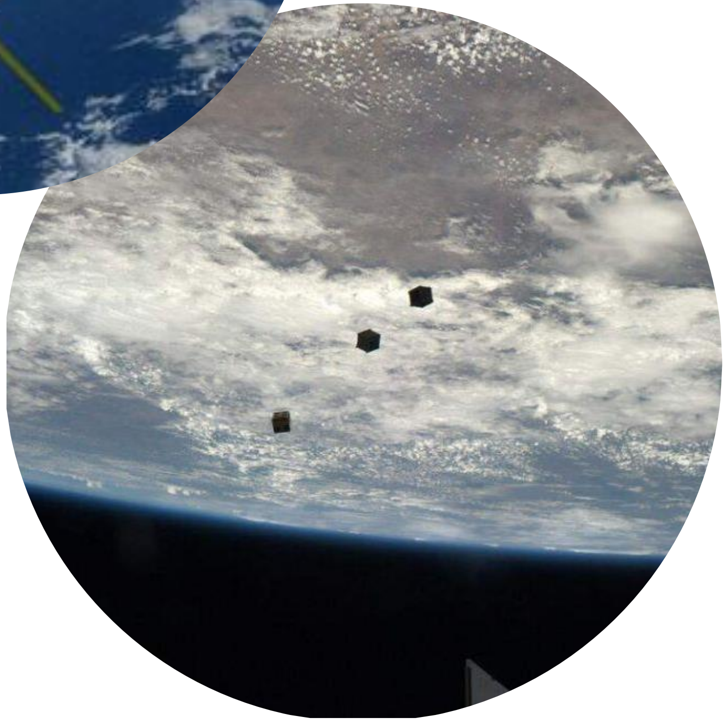
Alfredo Grieco

PhD researcher:

Antonio Petrosino

Index

1. Case of study
2. Structure of Framework
3. Optimization Algorithm
4. Result
5. Conclusion



1. Case of study

The goal is to increase the agility and scalability of satellite networks while reducing costs and simplifying operations.

However, there are some challenges to address, such as the limited resources of CubeSats, high latency in communications, and limited connectivity.

To solve these issues, it is necessary to define the **problem requirements**, **develop** an **appropriate architecture (framework),** create a **simulation environment**, implement an **optimization algorithm**, and **evaluate** it.

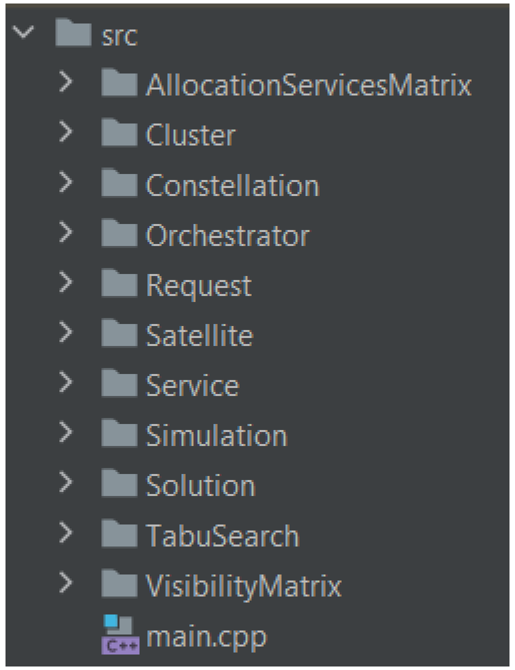
2. Structure of Framework

Constellation: A constellation is a collection of many satellites working together to provide a specific purpose or service.

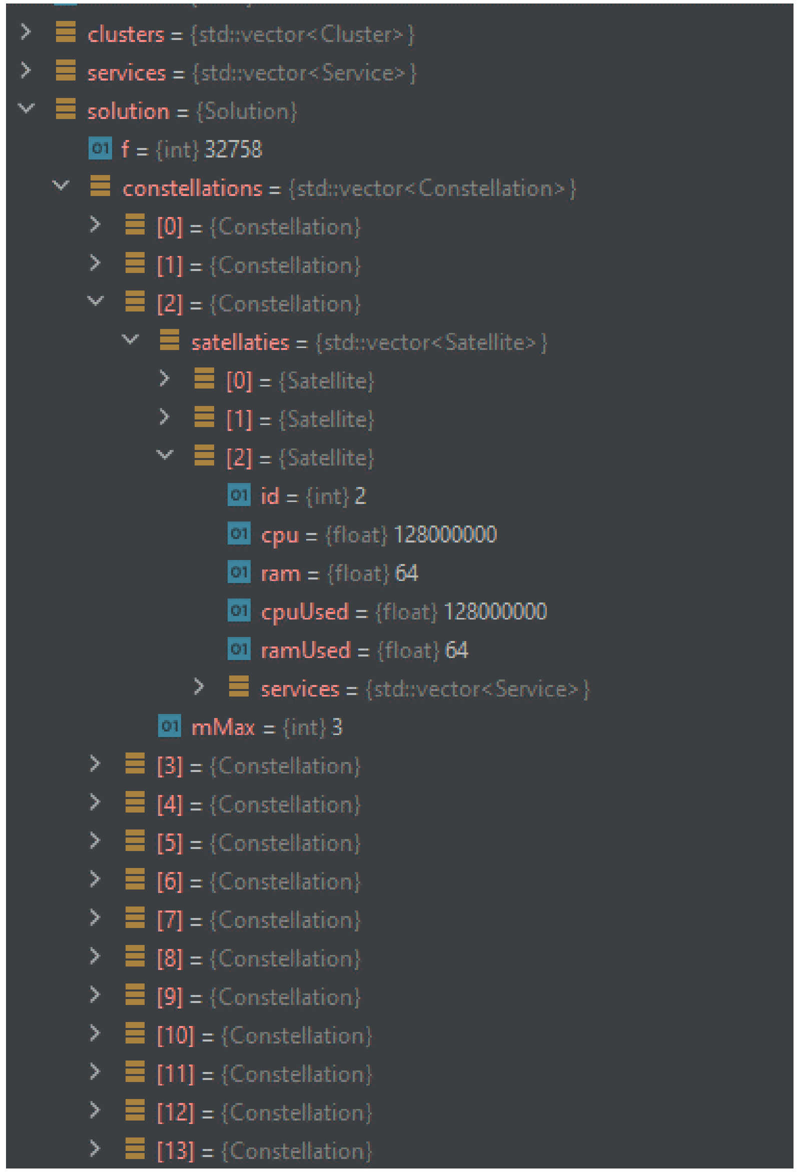
Satellite: Each constellation consists of one or more satellites.

Services: Each satellite within a constellation offers a specific set of services.

Cluster: Is a set of clients that require a service from satellites, the **Requests**.



```
src
  AllocationServicesMatrix
  Cluster
  Constellation
  Orchestrator
  Request
  Satellite
  Service
  Simulation
  Solution
  TabuSearch
  VisibilityMatrix
  main.cpp
```



```
clusters = {std::vector<Cluster>}
services = {std::vector<Service>}
solution = {Solution}
  f = {int} 32758
  constellations = {std::vector<Constellation>}
    [0] = {Constellation}
    [1] = {Constellation}
    [2] = {Constellation}
      satellaties = {std::vector<Satellite>}
        [0] = {Satellite}
        [1] = {Satellite}
        [2] = {Satellite}
          id = {int} 2
          cpu = {float} 128000000
          ram = {float} 64
          cpuUsed = {float} 128000000
          ramUsed = {float} 64
          services = {std::vector<Service>}
      mMax = {int} 3
    [3] = {Constellation}
    [4] = {Constellation}
    [5] = {Constellation}
    [6] = {Constellation}
    [7] = {Constellation}
    [8] = {Constellation}
    [9] = {Constellation}
    [10] = {Constellation}
    [11] = {Constellation}
    [12] = {Constellation}
    [13] = {Constellation}
```

3. Optimization Algorithm

The goal of the algorithm is to optimize the **delay** by **moving services** along satellites, respecting the request deadline and acceptance times. Since this is a real-world-based simulation, only requests up to that time slot are optimized (unit is timeSlot), ignoring future requests.

The algorithm with which the system was tested is **Tabu Search**, a meta-heuristic algorithm. Tabu Search's algorithm explores the solution space iteratively, called neighbors, using a list of "taboo" moves that prevents immediate return to previously visited solutions.

Tabu Search does not guarantee the absolute optimal solution, but offers a good quality solution within predetermined time or resource constraints

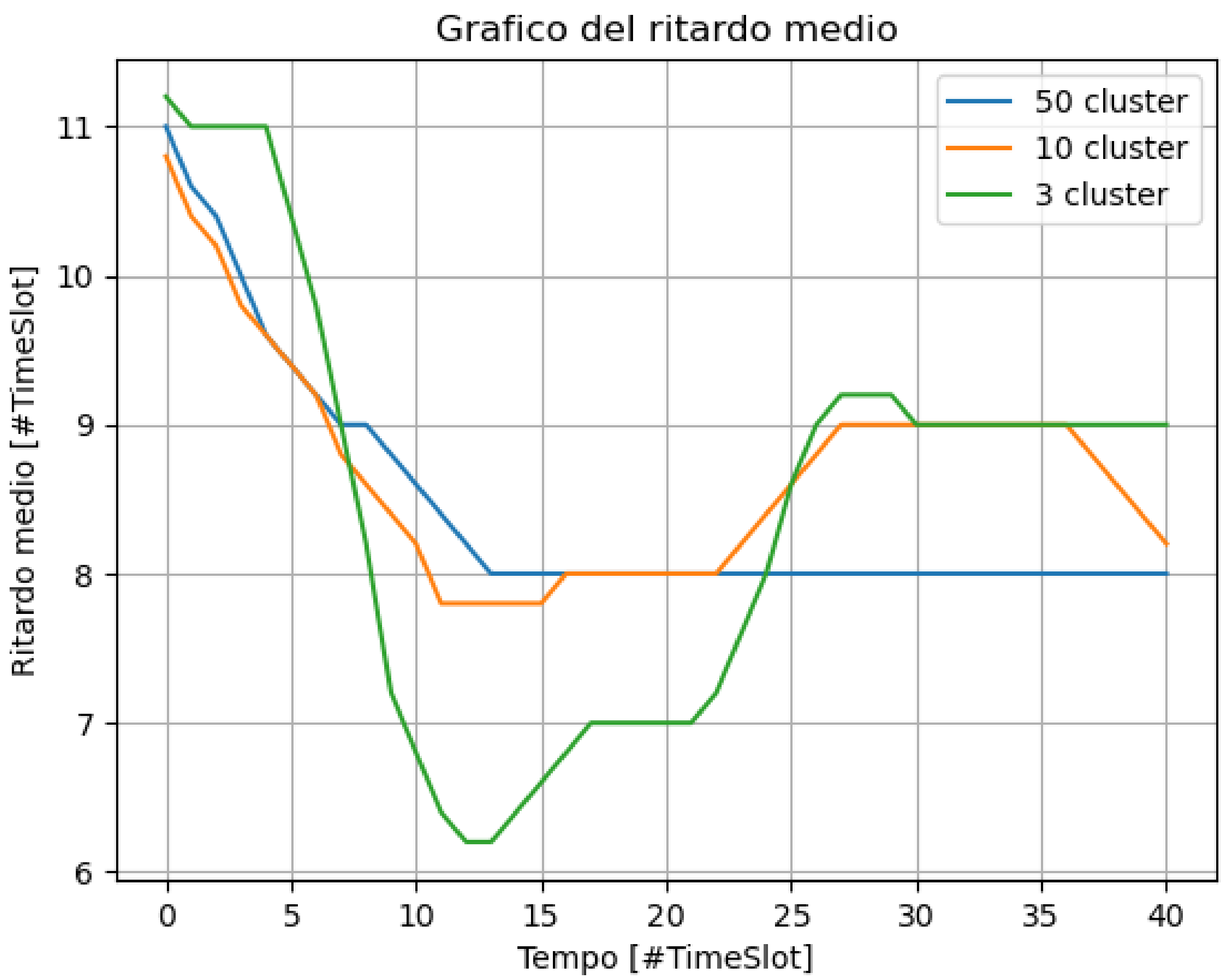
*timeSlot: value based on the number of satellites, of the duration of revolution of individual satellites

4. Result

Questo grafico mostra il ritardo medio che ogni utente percepisce (in media) per ogni richiesta di servizio che fa alla core network. Il ritardo minimo sono 3 slot perché abbiamo (raccolta del servizio, ottimizzazione + deploy del servizio) e il ritardo massimo è la deadline di 6 ore.

Ogni utente genera richieste con una distribuzione di Poisson (6 richieste al giorno). La rete accoglie le richieste senza sovraccarichi anche con 50 utenti

Durante tutta l'intera giornata di simulazione.
Come si nota dal grafico la rete non è ancora in overload



5. Conclusion

The presented **framework** is suitable for real satellite simulation. The system was tested with a single optimization algorithm, which reported good results.

It would be interesting to use the system and use additional optimization algorithms, like simulated annealing or genetic algorithm.

Thanks for listening