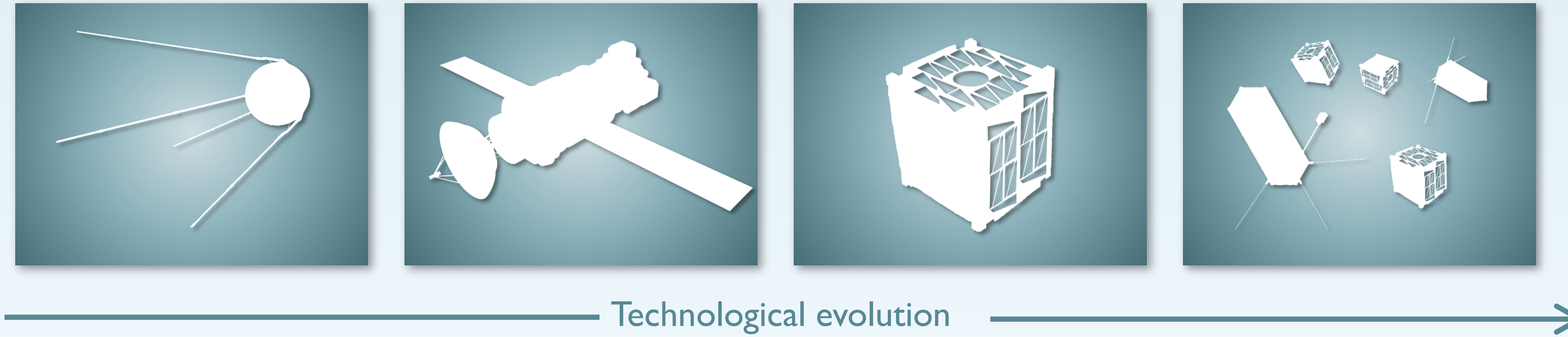
2015 IEEE Aerospace Conference
March 14th, Montana, US.

On Software Architectures for Distributed Spacecraft: A Local-Global Policy

Carles Araguz, Angel Alvaro, Kenny Root,, Iñigo del Portillo
Eduard Alarcón, Elisenda Bou-Balust

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Introduction



→ **New trend: distributed spacecraft**

- Swarms, Fractionated Satellites, Federated Systems, Constellations, Satellite trains…

→ **Challenges / enabling technologies**

- Networking and Communications → Inter-Satellite Links, Protocols, DTN, Phy…
- Wireless Power Transfer → Service areas, distributed power, …
- Cluster flight → Collision avoidance, Flight formation, …
- Distributed computing → distributed algorithms, decentralized management, …

Introduction

→ Software remains in the background

- The design of suitable software architectures needs to be addressed.
  - Resource management and exchange.
  - Autonomous task allocation.
  - Designed to mitigate technical constraints.
  - Empower new functionalities (new mission concepts).
  - Mission operability, security and robustness.

→ How to conceive software architectures for current mission architectures?

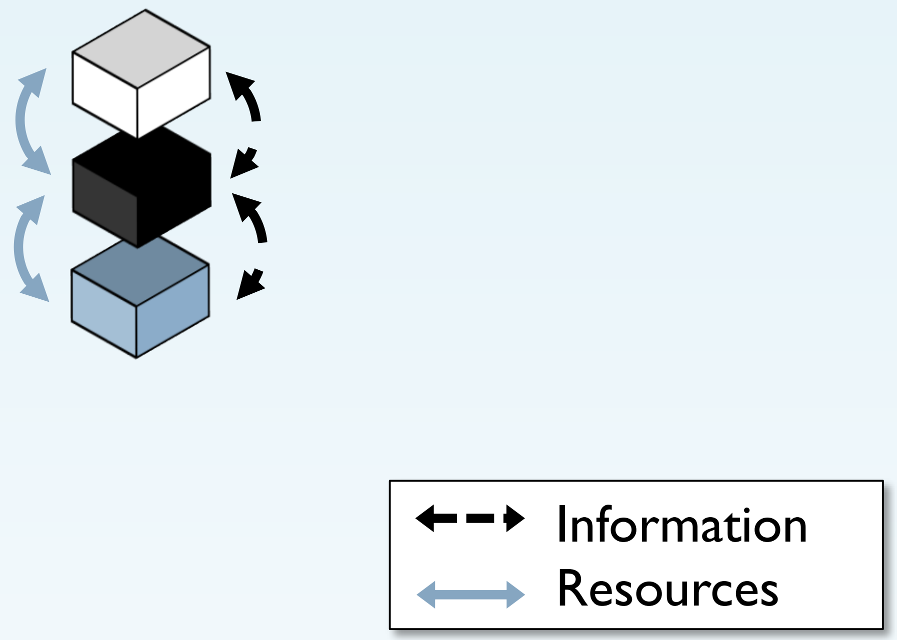
- What are the key characteristics in distributed spacecraft?
- What are the missing features in current designs?
- Can new software architectures be accommodated to all mission topologies?

Taxonomy of Distributed Satellite Systems

→ Mission archetypes:

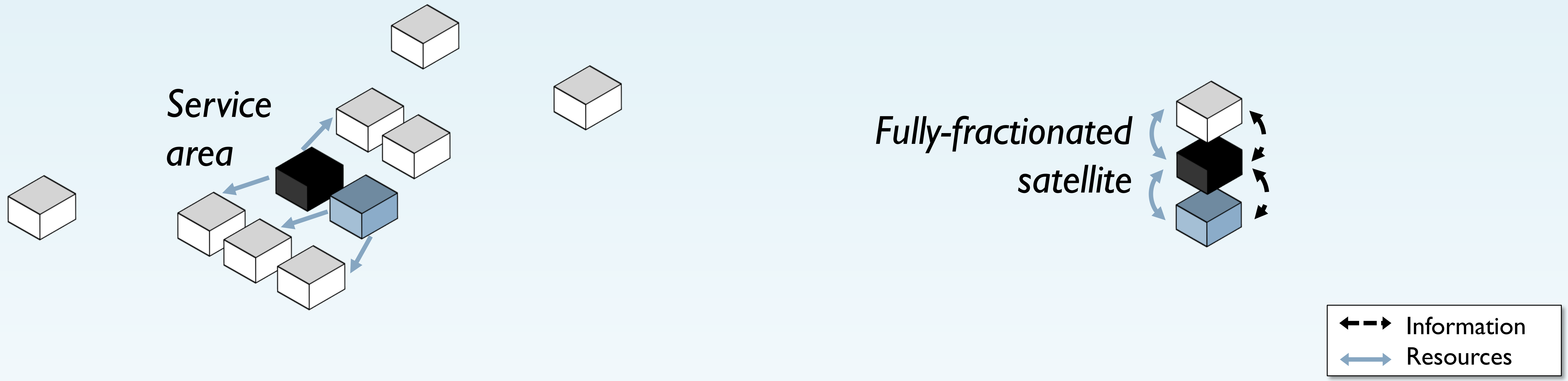
- Fractionated Satellites:
  - Fractions are devoted to specific purposes (power, energy storage, data processing, ground link, …)
  - Both ***management information*** and ***resources*** (power, bandwidth, …) are exchanged among modules.
  - Fully functional symbiosis.

Fully-fractionated satellite



Taxonomy of Distributed Satellite Systems

→ Mission archetypes:



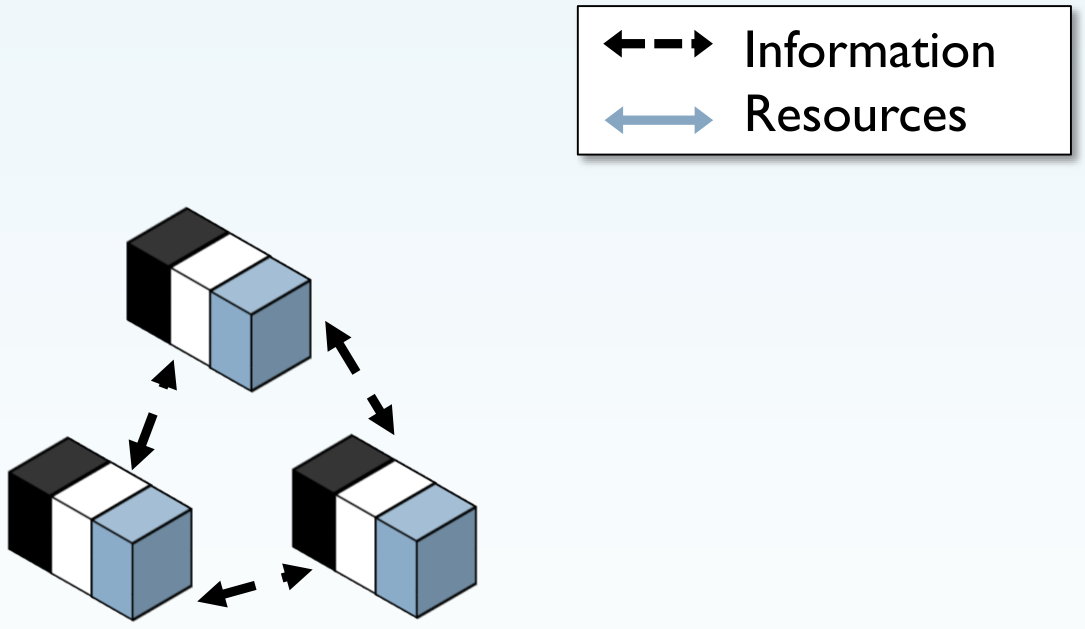
- New concepts have also appeared, such as service areas:
  - Satellite modules provide a specific resource to other modules sporadically.
  - There is exchange of resources (power, bandwidth, processing time, data storage, …)
  - Service providers and consumers do not share common goals (i.e. are not part of the same mission).
  - Consumers are fractions of a distributed satellite.

Taxonomy of Distributed Satellite Systems

→ Mission archetypes:

- Federated Satellite Systems:
  - Collaborative (opportunistic) missions.
  - Each module is a complete satellite (can operate <u>independently</u>).
  - ***Distributed management***/collaboration implies a certain exchange of management information.

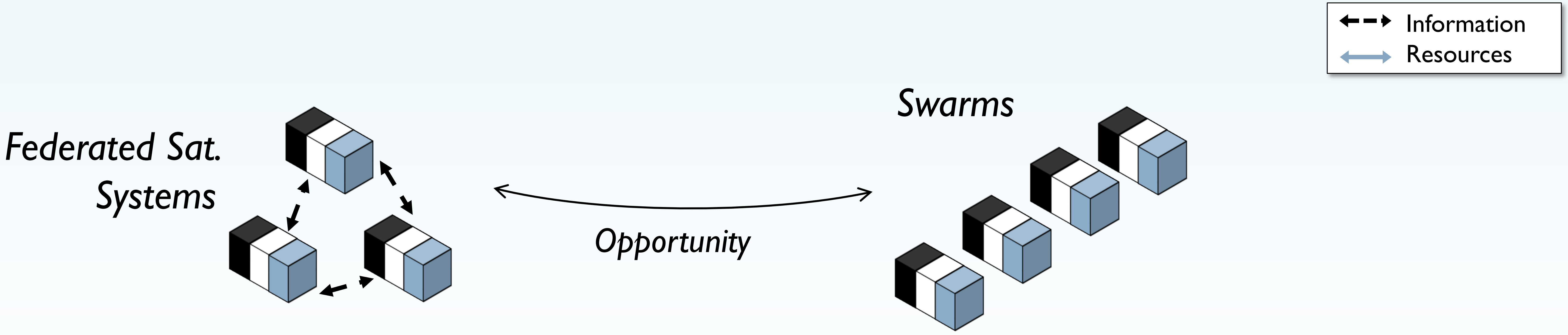
Federated Sat. Systems



Taxonomy of Distributed Satellite Systems

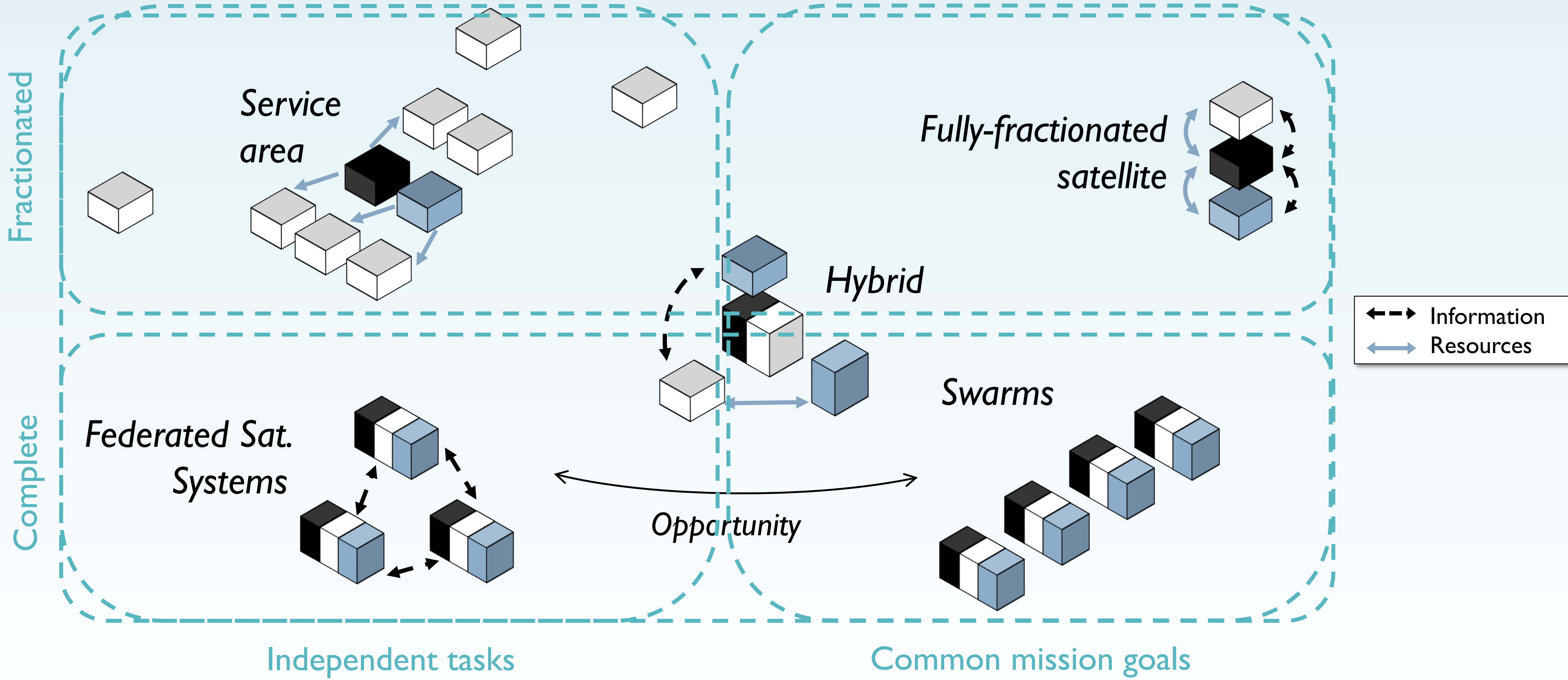
→ Mission archetypes:

- Satellite swarms / constellations
  - Independent satellites (usually homogeneous)
  - Each module performs its own tasks. The swarm is managed by ground operators.
  - No resource nor management information is exchanged among modules.



Taxonomy of Distributed Satellite Systems

→ Mission archetypes:



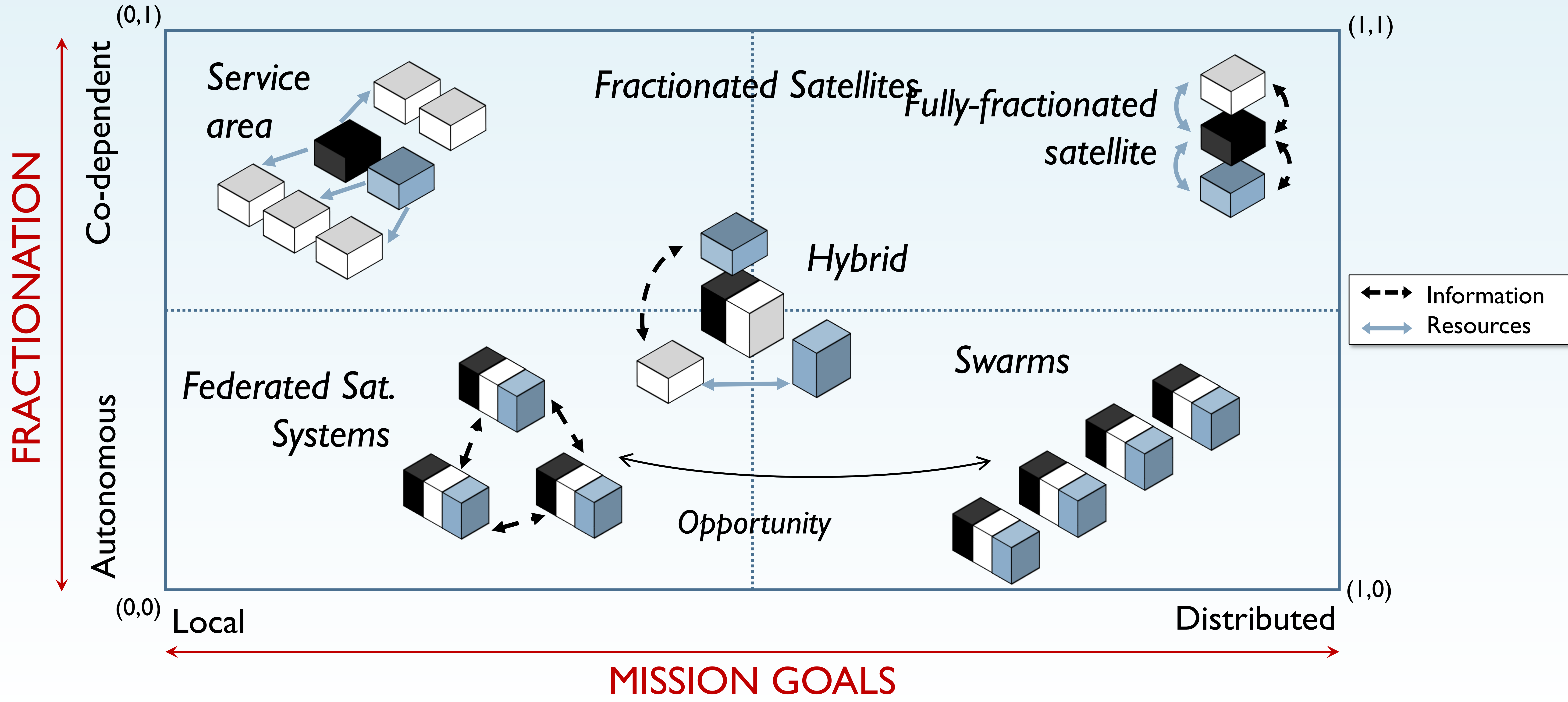
Taxonomy of Distributed Satellite Systems

→ Two parameters can be used to classify distributed satellite missions:

- **Degree of fractionation**: ***resource interdependence*** between modules.
  - How autonomous the modules are.
  - Exchanged resources (power, data storage, bandwidth, …)
  - Particularities of the exchange (continuous, intermittent, opportunistic)
  - 0: Fully-fractionated satellites (co-dependent) ↔ 1: autonomous (independent)
- **Mission goals**: ***local*** to the modules, or ***global*** to a distributed infrastructure.
  - Whether there is a distributed mission management.
  - 0: Each satellite module performs a set of activities which seek to accomplish a local objective.
  - 1: Satellite modules develop small portions of a global objective. (e.g. multi-spectral measurement where each sensor is located at a different satellite module)

Taxonomy of Distributed Satellite Systems

→ Classifying distributed spacecraft:



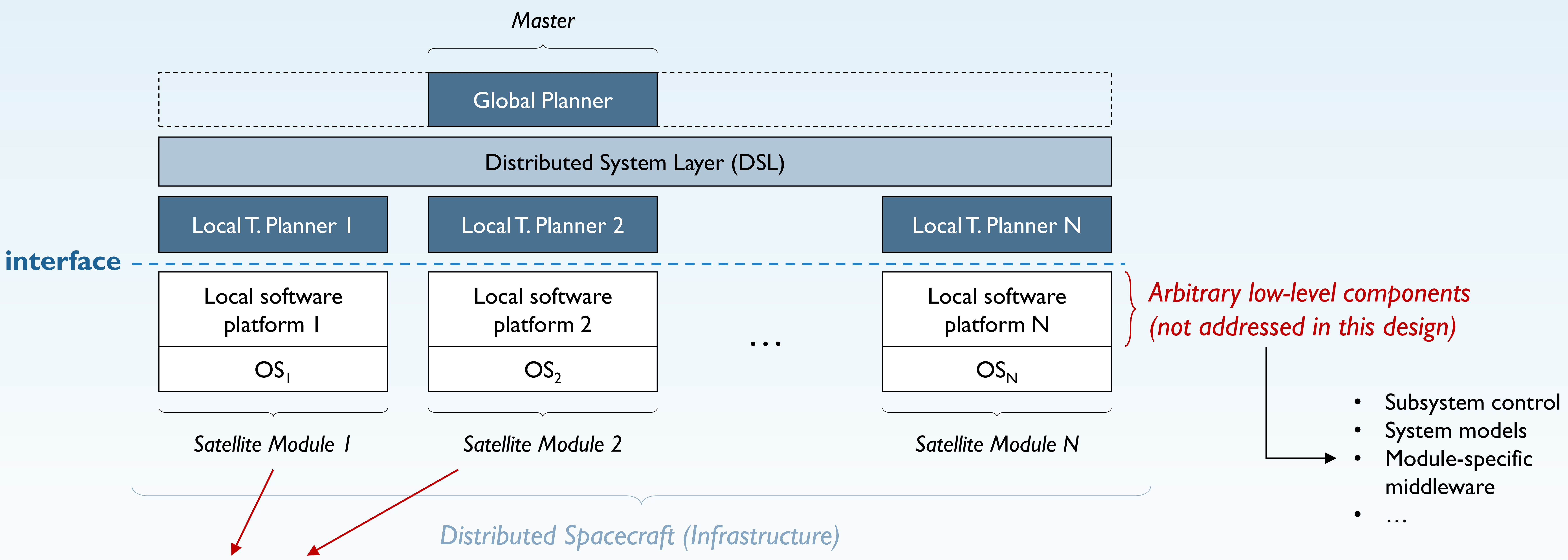
An Autonomous Software Arch. for Distributed Spacecraft

→ Software architecture: preliminary design

- Based on the features of these mission archetypes, a suitable software architecture has been designed.
- Instead of addressing low-level components (OS, middleware, models, mission-specific components), the software design is approached as a <u>top-level description</u>.
  - Encapsulation of systems.
- Software architecture for autonomous distributed spacecraft.
  - Components interact to autonomously operate the system:
    - Distribute tasks among satellite modules / nodes.
    - Allocate infrastructure resources for these tasks.
    - A policy is defined to perform task scheduling in a distributed manner.
- Currently in its prototyping phase at UPC's NanoSat Lab.

An Autonomous Software Arch. for Distributed Spacecraft

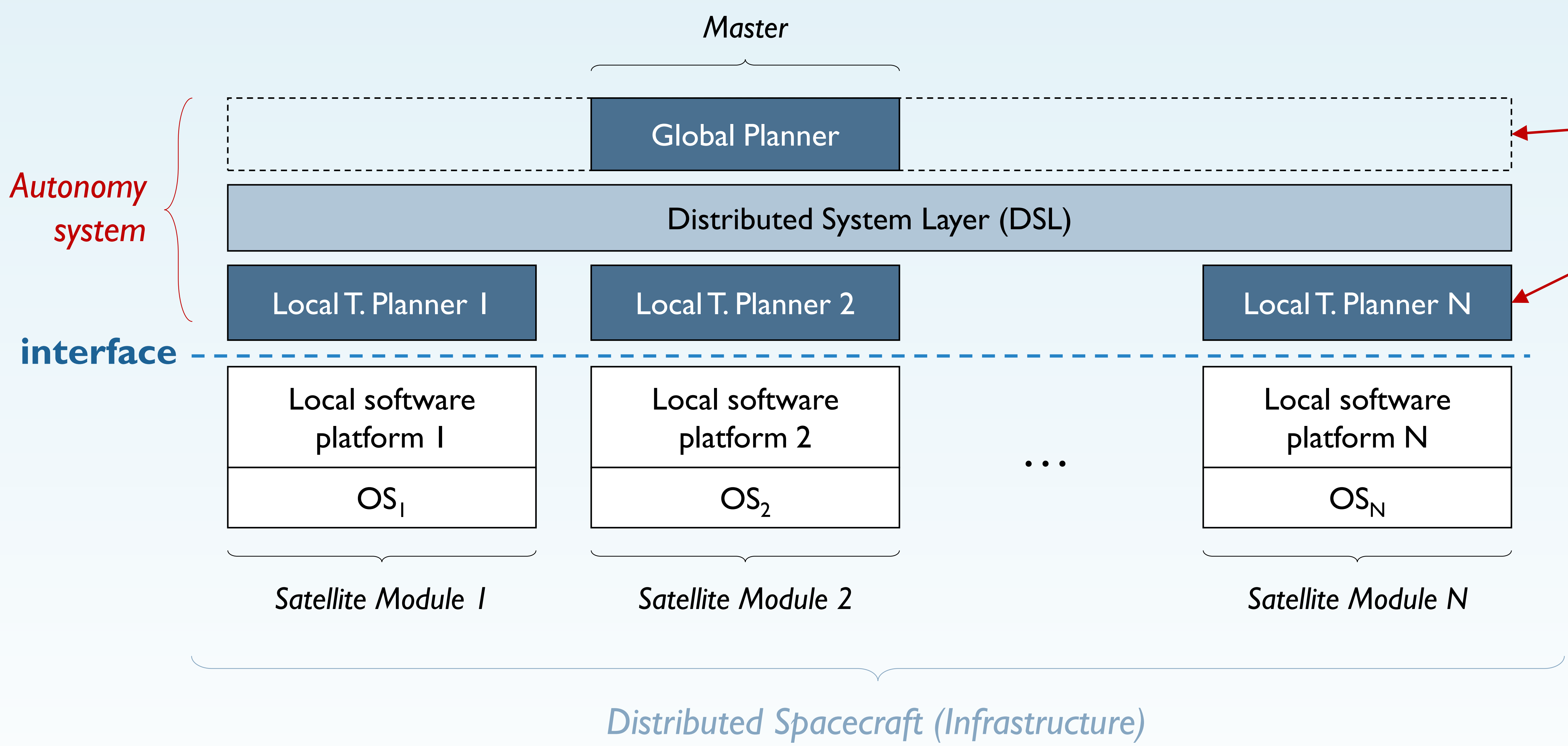
→ Structural view:



- Modules need not be homogeneous (i.e. different computational capabilities, payloads, subsystem availability/capabilities…) → System encapsulation.

An Autonomous Software Arch. for Distributed Spacecraft

→ Structural view:



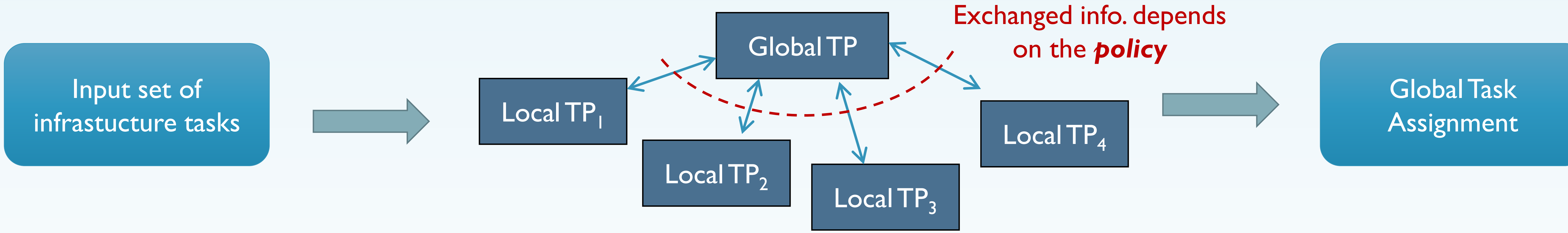
- Two control levels:
  - **Global:** relative to the infrastructure domain.
  - **Local:** relative to each module domain.
- Hierarchical relation similar to master-slave.

- The autonomy system is composed of *Autonomy Management Entities* (i.e. task planners) which interact to operate the spacecraft autonomously.
- DSL provides a transparent communication channel (through ISL) between global and local entities.

An Autonomous Software Arch. for Distributed Spacecraft

→ Functional view:

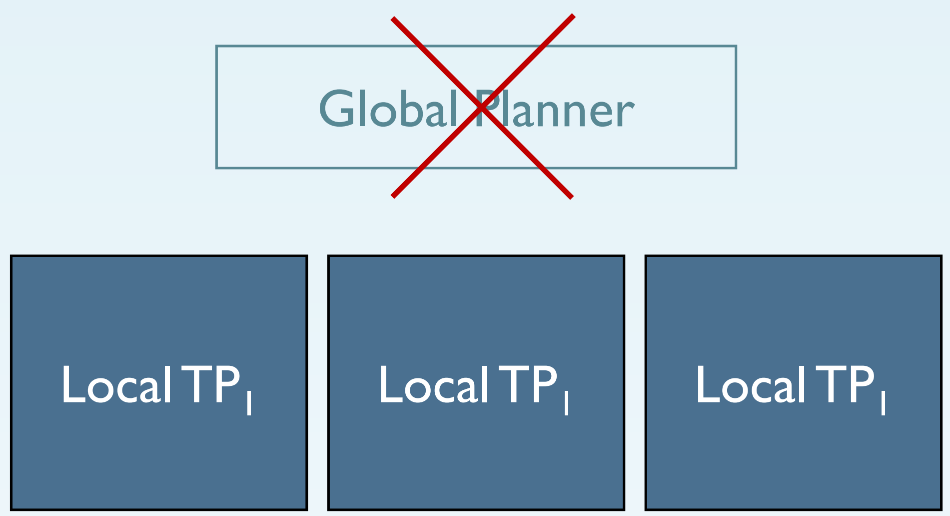
- Locally managed activities are hidden to the autonomy system.
  - Activities/tasks performed by local software platforms. E.g.:
    - Maintenance tasks.
    - Flight formation.
    - Functionalities/tasks extrinsic to the infrastructure.
- Global tasks are scheduled by the autonomy system.
  - Activities/tasks that could be executed, a priori, by any node in the infrastructure.



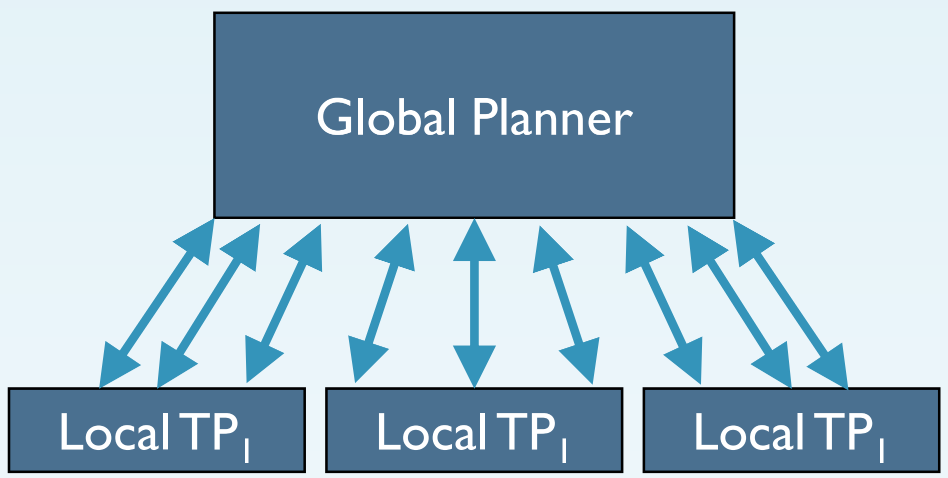
- "Policy" as the architecture's functional behavior/model.
  - Establishes the exchange of information between the Global and Local control levels.
  - Provides a mechanism to perform distributed assignment of ***global*** tasks, for each node and period of time.
  - Compendium of algorithms.
- Software architecture for dynamic contexts → ***dynamic*** management policies.

An Autonomous Software Arch. for Distributed Spacecraft

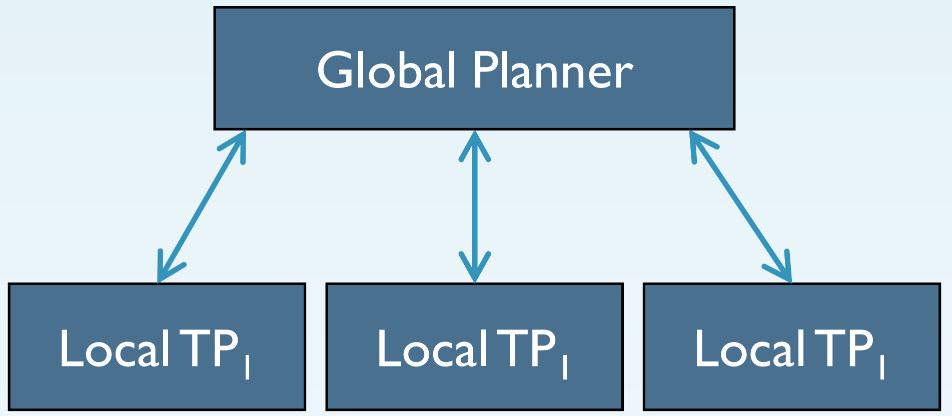
→ Possible scenarios (management policy types):



- ***Local management***: most information is processed at the local level.
- E.g. Swarms



- ***Global management***: utterly centralized management (requires *all* local information to be transferred to the global entity, which processes it)
- E.g. FracSats.



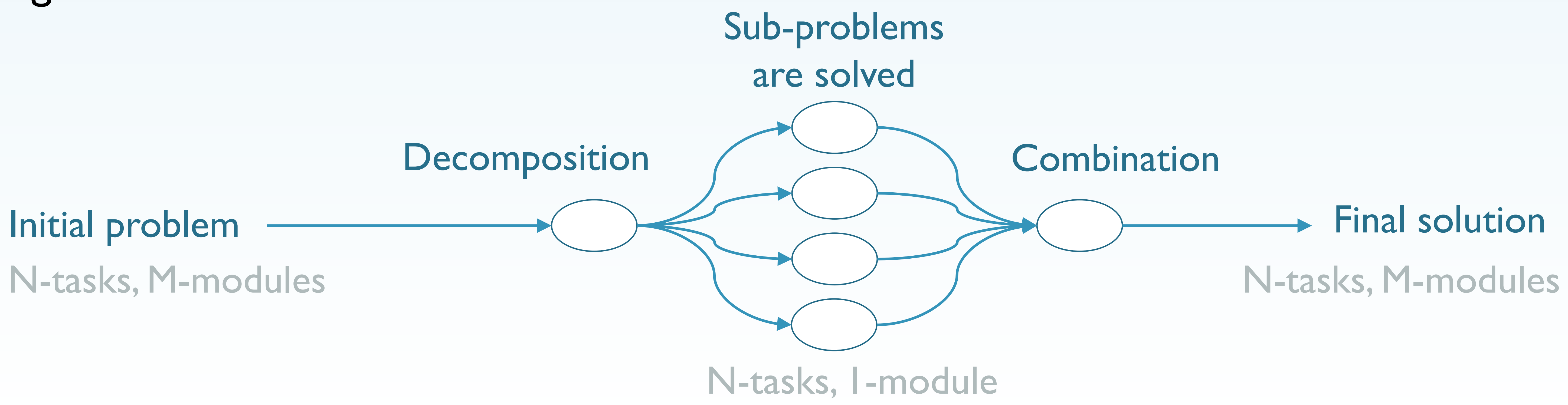
- ***Mixed management***: information exchange and process is balanced between local and global entities.
- E.g. FSS

→ Dynamic: the policy changes with mission opportunities.

The Local-Global Policy

→ Local-Global approach (L-G):

- Mixed management policy.
- Aimed at providing an ***adaptive*** planning solution for a ***distributed*** spacecraft with an arbitrary number of ***heterogeneous*** modules (i.e. different platforms, hardware, payloads, computational capabilities, ISL bandwidth, …)
- Adapts to the number of modules present in the infrastructure.
- Balances the amount of information processed by the master node.
- Based on decomposing the "multiple-tasks multiple-modules" problem into "multiple-tasks single-module".



The Local-Global Policy

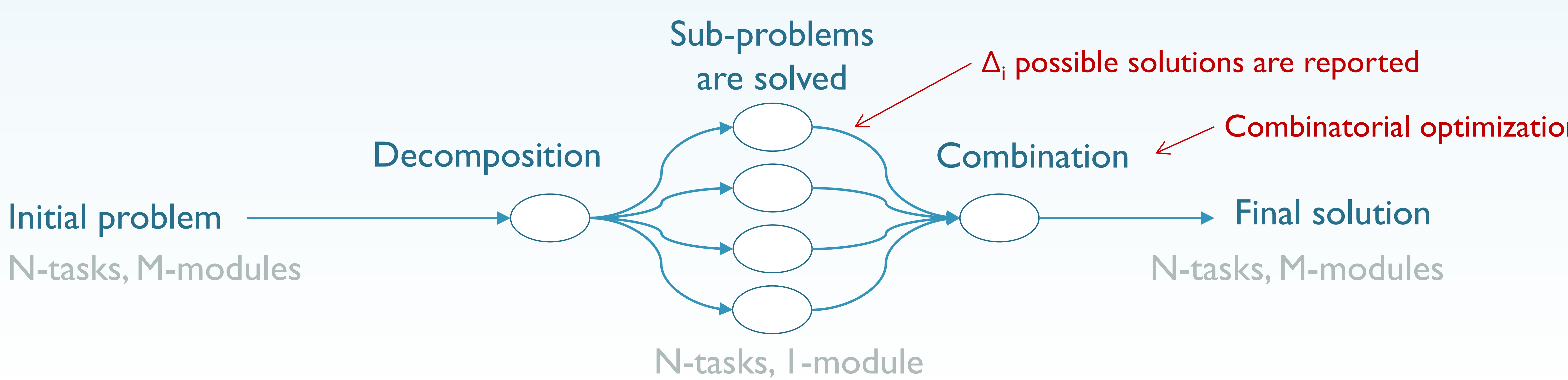
→ Local-Global approach (L-G):

Golden index (Δ, integer)

- Amount of reported sub-solutions by every local planner.
- Calculated a priori with the computational capacity of the master and network features.
- Tries to mitigate heterogeneity problems.

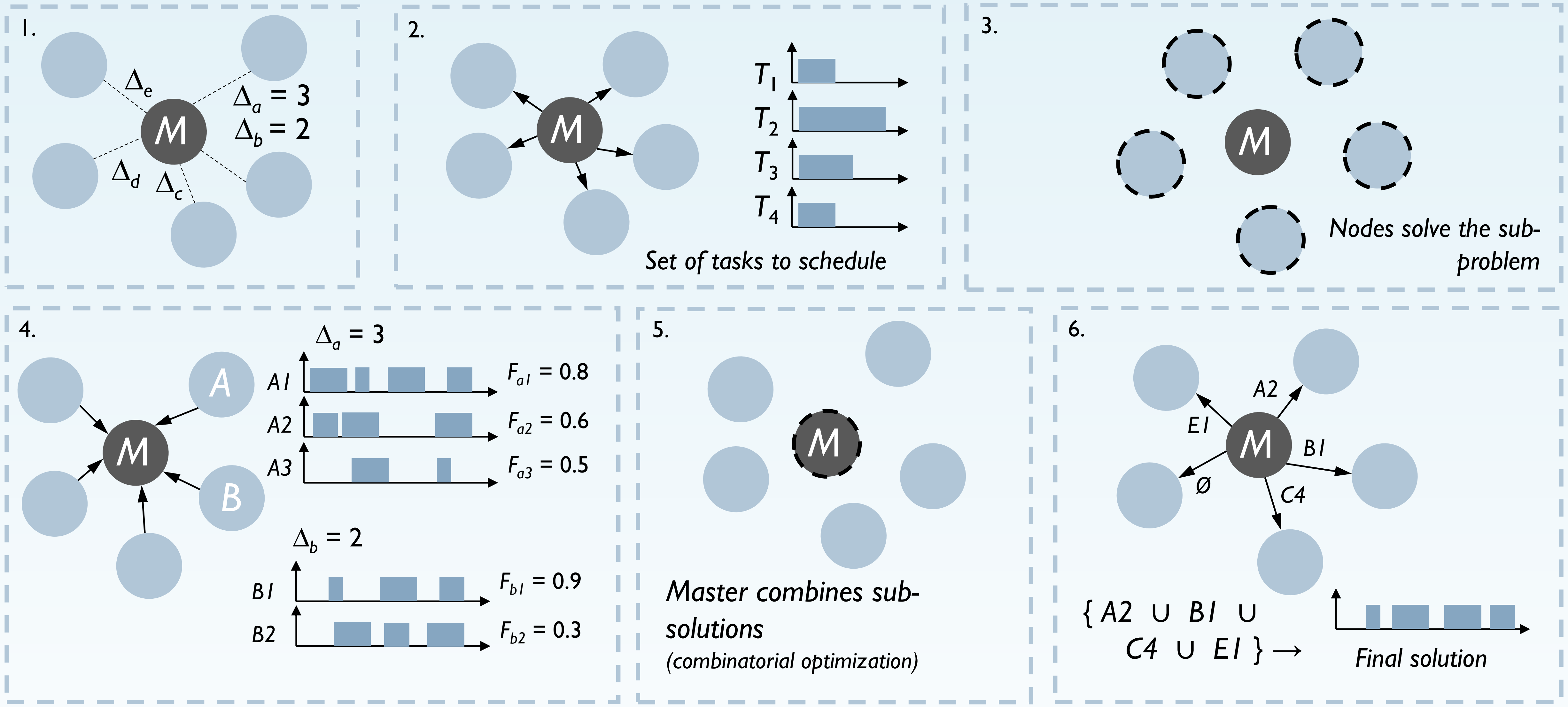
Figure of merit (F)

- Encompasses a set of variables that state the goodness for each sub-solution reported to the master.
- Optimality criteria.



The Local-Global Policy

→ L-G policy steps:

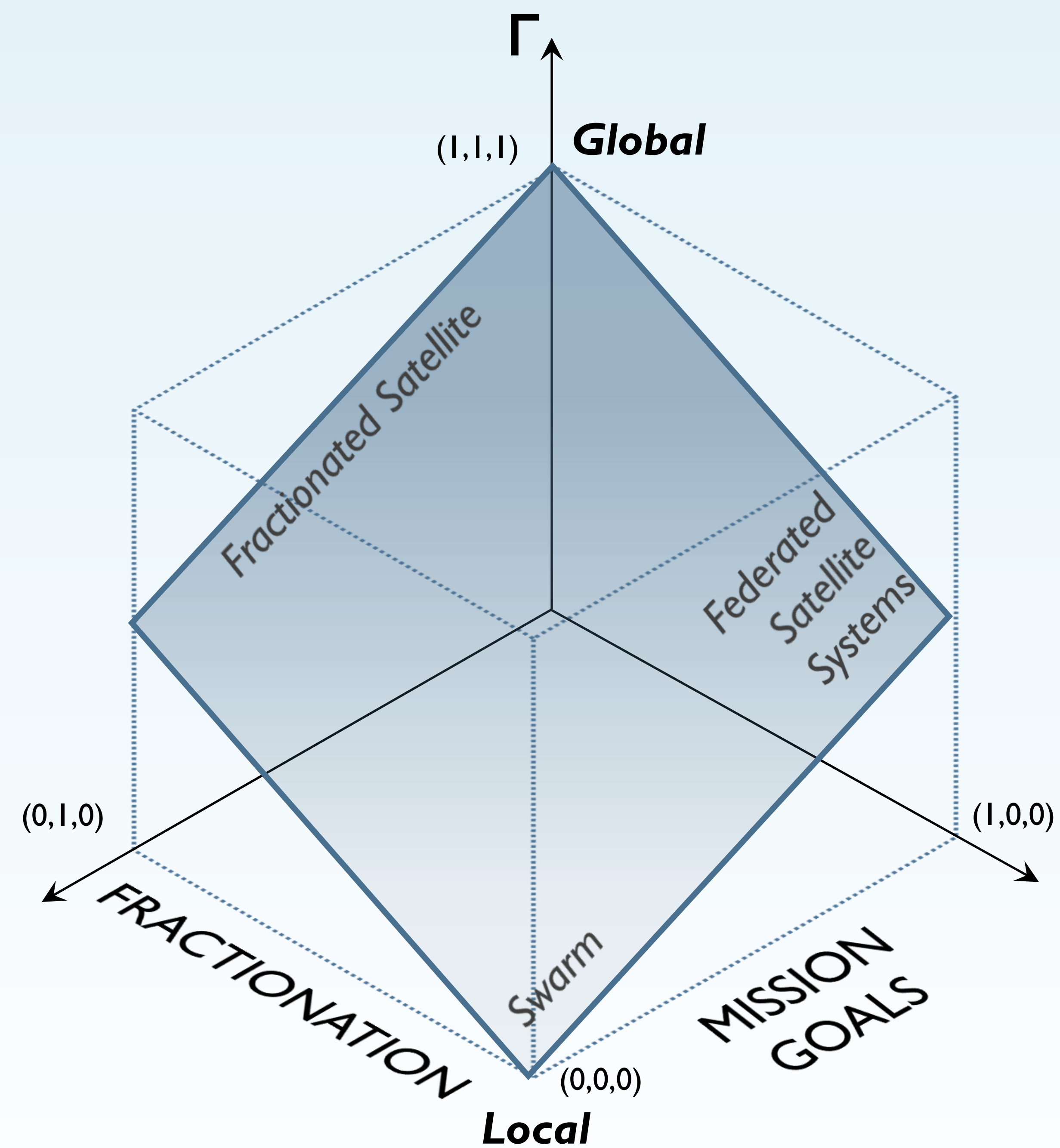


1. **Characterization**: Δ is set for every module.
2. **Task delivery**: determine scheduling window and distribute all tasks to all modules.
3. **Local evaluation**: potential sub-solutions are sorted by each local planner.
4. **Submission of solutions**: sub-solutions are reported in a simplified manner.
5. **Global selection and combination**: accepts and discards sub-solutions to meet some mission metrics (i.e. utility, agility, throughput).
6. **Distribution of solution**: the final solution is reported back to each module.

The Local-Global Policy

→ Parameter adjustment

- Adjusting the policy's parameters (Δ and |F|) allows to modify the amount of information processed by the master node:
  - $I_m = \sum I_i \cdot \gamma_i$
  - With $\gamma_i = f(\Delta, |F|)$
  - Static management policies + archetypes:
    - $\gamma_i = 0 \ \forall i$ → Swarms (no global computation).
    - $\gamma_i = 1 \ \forall i$ → Fully-Fractionated Satellites (no local computation).
    - $0 < \gamma_i < 1$ → FSS (both local and global computation).
  - Heterogeneous distributed spacecraft:
    - Aggregated ratio: $\Gamma = {}^{I_m}/_{\sum I_i}$



Conclusion

- Distributed spacecraft is an emerging paradigm which requires novel techniques to empower the mission development and operation of such missions.
- Presented distributed software architecture: valid for any kind of distr. mission.
- High-level generic architecture which encapsulates module's flight platforms:
  - Need to define standard interfaces.
- Resource-aware autonomy system:
  - Scalable scheduling policy based on a parametrized (Δ, $|F|$) collaborative procedure.
  - Computational burden is balanced among nodes.
  - Information exchanged through ISL is minimized.
  - Suboptimal solutions are produced:
    - Optimality is influenced by quality and variety of local sub-solutions.
- Resource exchange management not considered, could be performed through an additional step in the scheduling policy.

Thanks for you attention

Carles Araguz – ***carles.araguz@upc.edu***

Elisenda Bou-Balust – ***elisenda.bou@upc.edu***

Eduard Alarcón – ***eduard.alarcon@upc.edu***

Iñigo del Portillo – ***portillo@mit.edu***