ESILV
ENGINEERING SCHOOL
DE VINCI PARIS NANTES

DE VINCI
RESEARCH
CENTER

FROM CLOUD TO CONSTELLATION

Orchestrating Services at the Edge of Space

Orbital Computing and Cloud-Native Orchestration — State of the Art

Farah AIT-SALAHT

De Vinci Higher Education, De Vinci Research Center, Paris, France

Seminar WG —Cloud Optimization
May 29, 2026

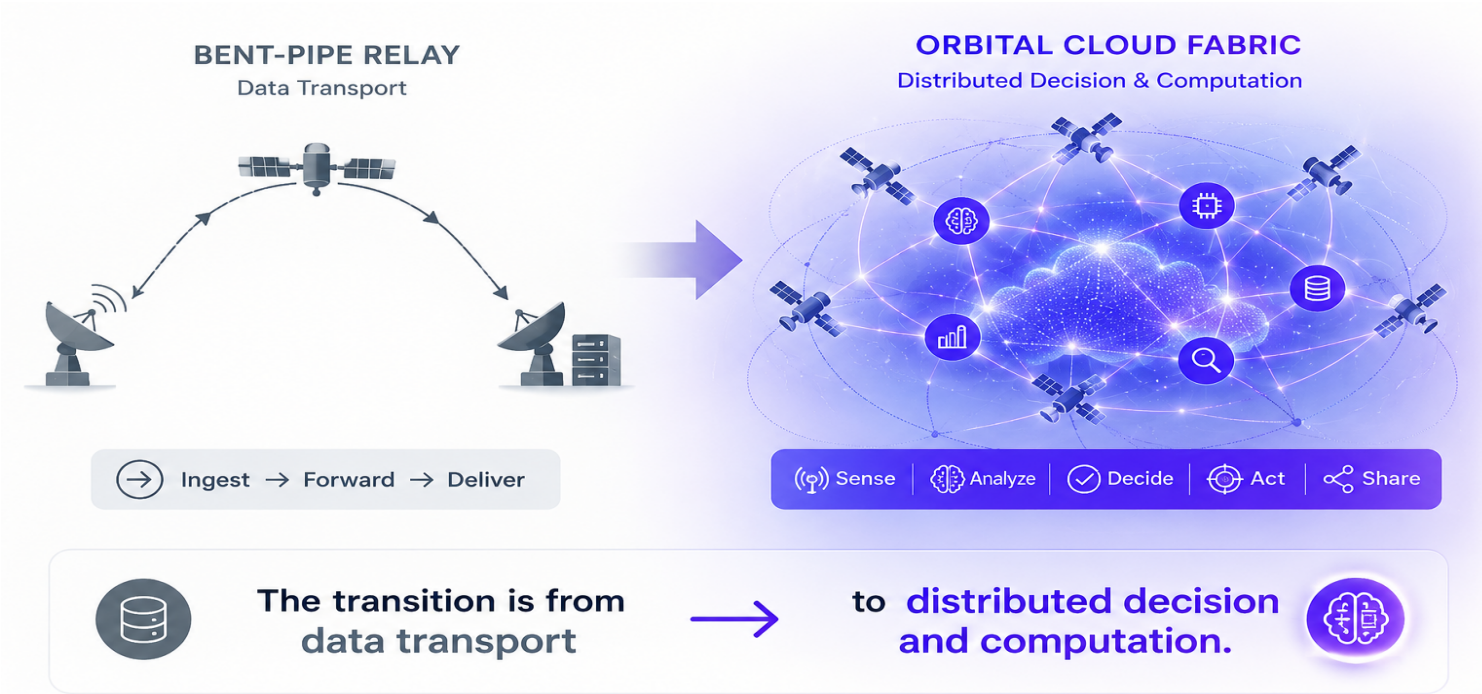
Outline

PART I: Context and stakes

What is the orbital cloud, why now, what does it enable, and what makes it intrinsically hard.

From bent-pipe relay to orbital cloud fabric

The transition is from data transport to distributed decision and computation.



The shift was formalized as a new class of computer system by Denby and Lucia, Orbital Edge Computing, ASPLOS Workshop, 2020.

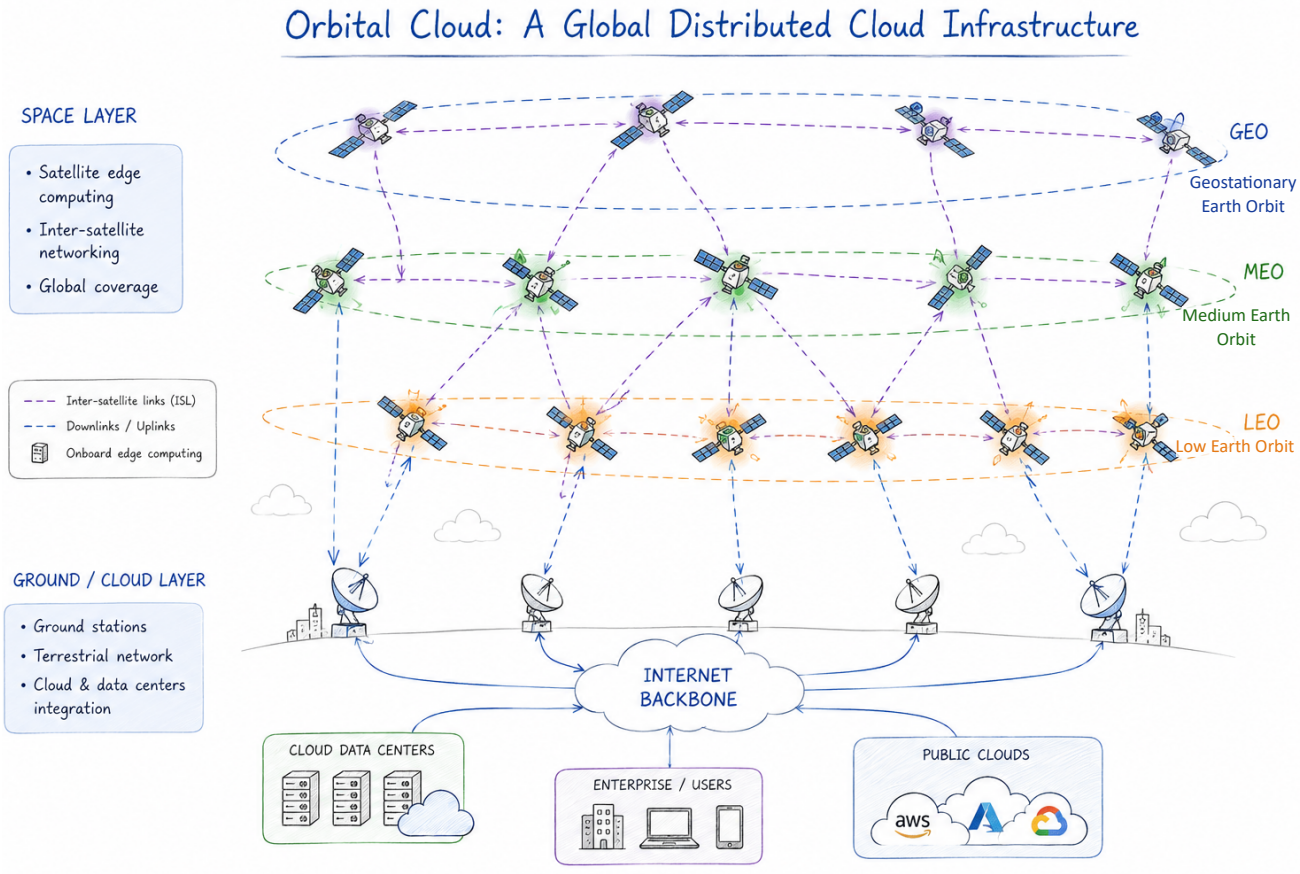
What is an orbital cloud?

Extending cloud computing and data services into space for faster, more resilient missions.

Definition: An orbital cloud is a distributed computing fabric where satellites, ground stations and terrestrial cloud resources jointly host, move and schedule services under orbital constraints.

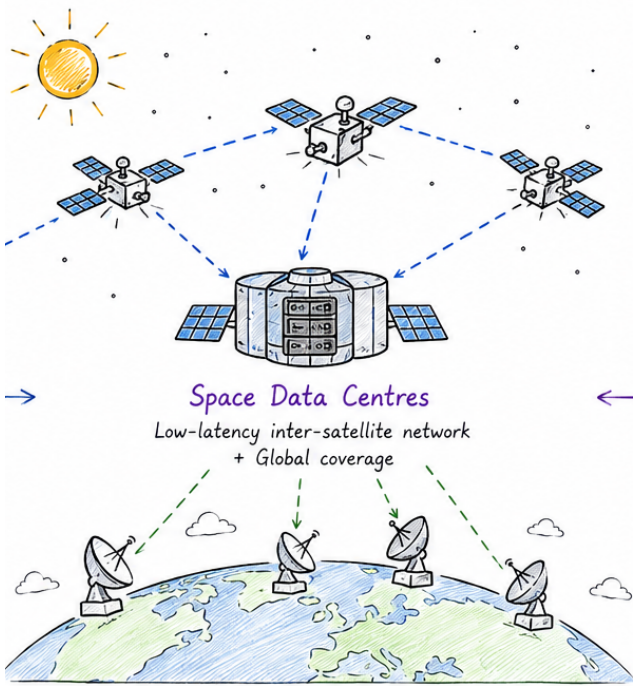
It combines:

- satellite edge computing
- inter-satellite networking
- ground/cloud integration
- service placement and migration
- task offloading and routing



Motivation

Why Data Centres Are Moving to Space ? Why now ?



Why now? Four converging signals

SIGNAL 1

- **LEO constellations at scale**

  More nodes, more contacts, more paths, and more dynamic choices for routing and scheduling.

SIGNAL 2

- **NTN moves beyond relay**

  Regenerative payloads add onboard processing, switching and routing capabilities.

SIGNAL 3

- **Onboard AI becomes real**

  Earth-observation payloads can filter, detect and compress before downlink.

SIGNAL 4

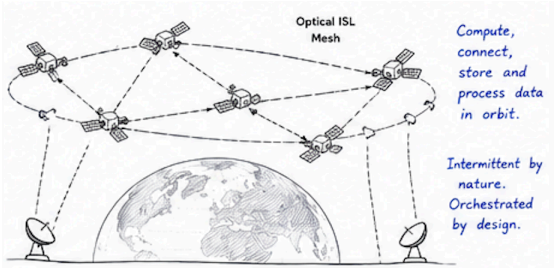
- **Cloud-native services reach constrained environments**

  Containers, functions, lightweight orchestration and model placement become relevant beyond terrestrial edge systems.

Motivation 1: From Raw Data to Timely Decisions

Process data where it is produced, and send only what is useful.

Earth observation satellites generate massive raw data, but the useful operational output is often much smaller and time-critical.



- **Bandwidth Constraint:** Downlinking raw Terabytes of Earth Observation data is physically limited by radio spectrum.
- **Latency Reduction:** Bypassing the Ground Station (GS) hop can reduce latency from hours/minutes to milliseconds for tactical users or emergency workflows (fire detection, collision avoidance, etc.)
- **Orbital edge opportunity:** On-orbit filtering and inference reduce data volume before transmission

  Raw Sensor Data → On-Orbit Inference → Minimal Actionable Downlink « Alert / confidence / location » (<1% volume).

Key message

« Transfer information, not raw bits. »

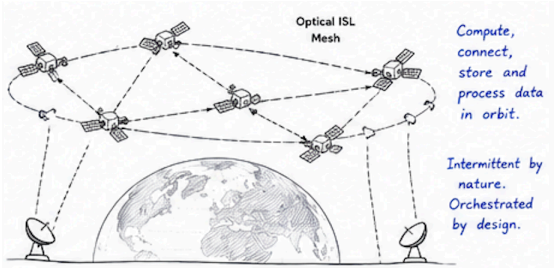
Do not downlink everything.

Downlink what matters.

Motivation 2: orbital compute opportunities

Why compute in orbit?

Orbital compute is becoming realistic because satellites now have more processing, networking and autonomy.



Potential advantages:

- processing closer to sensors and remote users
- near-continuous solar exposure in selected orbit (almost 24/7);
- take advantage of the vacuum of space as an “infinite radiator” to cool GPUs (radiative heat rejection instead of water-based cooling);
- **Reduce dependence on terrestrial constraints:** land availability, grid connection, water for cooling, permits, etc.
- improved resilience when ground infrastructure is degraded

But it is not free:

- launch cost and launch emissions
- radiation and thermal cycling
- limited onboard power
- end-of-life debris risk
- constrained maintenance and upgrades

According to Starcloud, orbital data centers could offer up to around 90% lower electricity costs compared with Earth-based data centers, once the satellite has been launched.

Motivation 3: Sustainability argument

"You get the free energy — but you also remove all of the carbon footprint and the heat generation for this AI work. Get it off Earth."

— Recurring industry framing (Google Project Suncatcher, Axiom ODC, Lonestar)

✳ THREE STRUCTURAL ADVANTAGES IN ORBIT

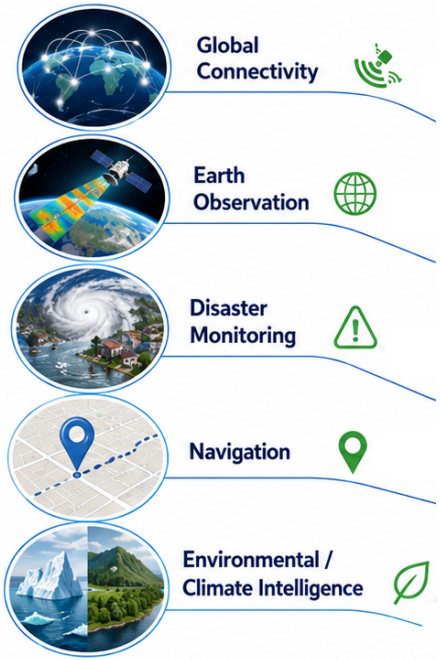
- Continuous or near-continuous solar exposure for selected orbits.
- Radiative heat rejection rather than terrestrial water/HVAC cooling.
- Potential relief for terrestrial grids under AI compute growth.

✳ BUT IT IS NOT FREE

- Launch carbon and end-of-life debris must amortize over years of operation.
- Radiation and thermal cycling degrade hardware faster than terrestrial DCs.
- Energy is still bounded per satellite — orbital cloud trades unlimited grid for hard local budgets.

Applications and Use cases

Use Case	Why orbital matters	Dominant constraint
Real-time EO alerts wildfire · oil spill · vessel	Raw EO image is hundreds of MB; an alert is <1 kB. Onboard filtering collapses 99%+ of the downlink and beats the next contact window.	Latency to first alert · contact window
Defense and ISR sovereign · resilient	Data sovereignty: results never touch foreign infrastructure. Operation continues if gateways are denied.	Trust · isolation · jamming
Connected IoT and vehicles global · low-power	Ubiquitous coverage for ships, planes, remote assets without terrestrial backhaul.	Power budget at the device
Disaster response degraded ground	Ground infrastructure is down. Orbital edge AI fuses sensors and dispatches alerts directly to responders.	Time-criticality · uncertainty
AI inference offload sustainability	Heavy inference moves off Earth — solar energy, radiative cooling, no grid impact.	Compute density · thermal
NTN telecom and 6G Regenerative	Regenerative payloads do baseband processing in orbit, reducing end-to-end latency.	Spectrum · handover



Industry and Institutional Signals

ASCEND

Thales Group • 27 juin 2024

ASCEND : la faisabilité du data center spatial européen confirmée

Consortium Thales Alenia Space, Orange, HPE, CloudFerro, ArianeGroup, Airbus DS, DLR. Objectif : 1 GW orbital d'ici 2050.

airbus defence and space

Press Release | 7 Mar 2024

Airbus to develop space cloud infrastructure for European institutions

Airbus leads industrial consortiums to define and develop space cloud services for secure and resilient European use.

Space Cloud Services

2024

Le CNES et l'Inria veulent traiter les données dans l'espace

EUTELSAT

2024 – 2026+

Eutelsat Group – Space cloud connectivity

Extending OneWeb and GEO assets with cloud-based services in space.

- Edge computing & in-orbit processing
- Network APIs and cloud services
- Partnerships with European cloud providers and institutions

eutelsat.com

esa

FutureEO | 2022–2025

ESA FutureEO programme: autonomous operations and in-orbit data management

FutureEO develops technologies for autonomous satellite operations, in-orbit data handling and distribution.

Autonomous & Smart Operations

Challenge IA

Région IDF + CNES • 15 janvier 2026

Challenge AI for Space — 1 M€ pour l'IA appliquée aux donnée satellitaires

Deux défis : démonstrateur d'analyse auto. d'image service applicatif IA géospatial. Clôture le 9 mars 20

AIRBUS

2024 – 2026+

Space Cloud Infrastructure for Europe

Airbus develops modular payloads and platforms for on-orbit data processing and storage.

- Edge computing in orbit
- Scalable data handling
- Industrial partnerships & demos ongoing

airbus.com

ESA

Payload Space • 4 mai 2026

ESA signe avec Edge Aerospace pour les orbital data centers

Contrat luxembourgeois pour définir l'architecture et la roadmap européenne, dans une logique de souveraineté.

Europe is building today the orbital cloud infrastructure of tomorrow.

Connected. Intelligent. Sovereign.

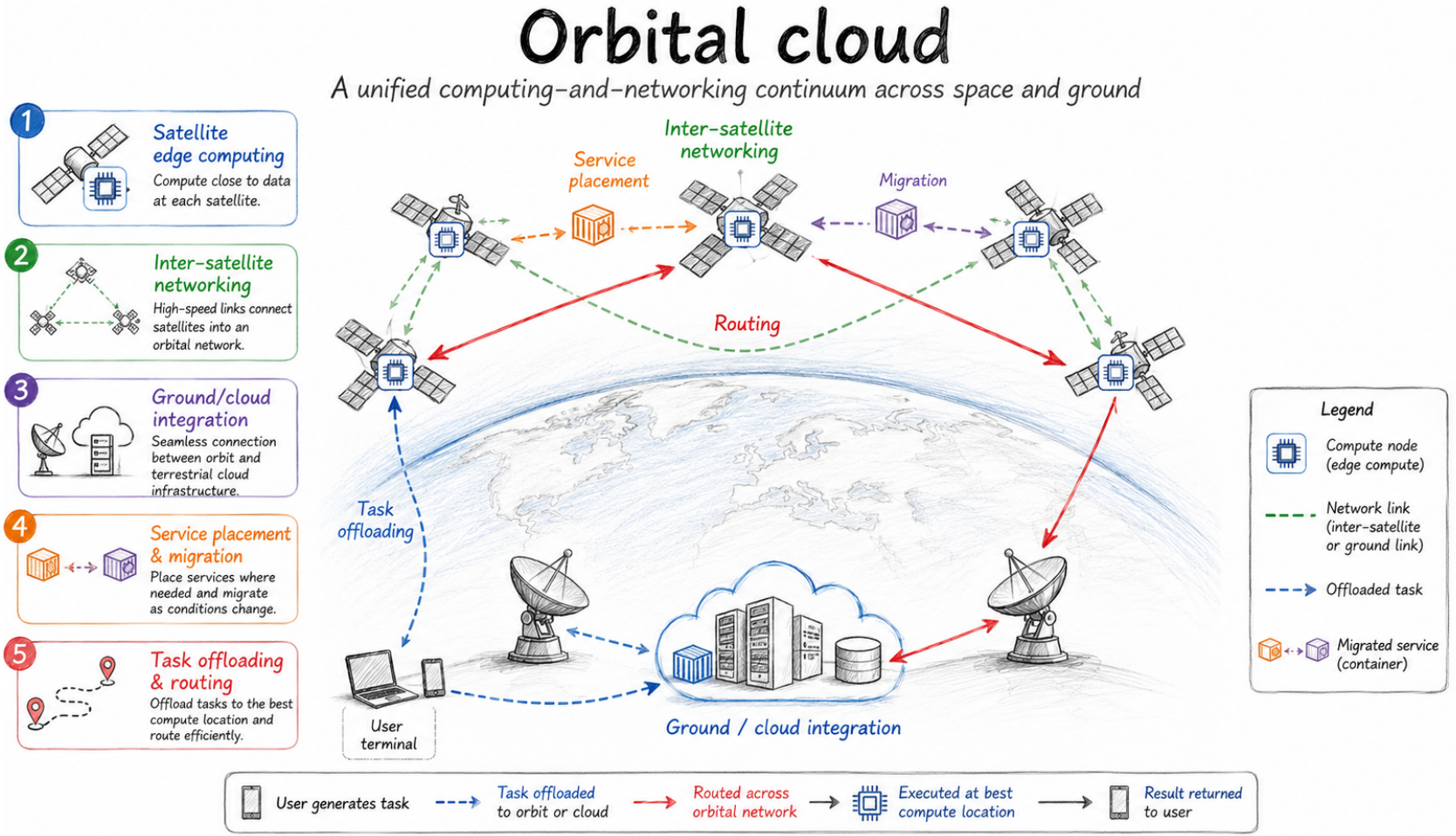
Research Challenges

Space offers new opportunities, but also creates harder constraints.

PART II: What problem are we solving?

From raw data transport to distributed decision-making

Focus of This Presentation



Focus of This Presentation

From orbital compute to orbital orchestration

Once satellites become compute-capable nodes, the key challenge is not only to process data in orbit, but to decide how services should be orchestrated across the constellation.

The Decision Loop

- **Service Placement:** Where should a container, function or AI model be deployed?
- **Task scheduling:** When and where should each workload stage run?
- **Task Offloading:** Should computation run onboard, on a peer satellite, at a gateway or in the cloud?
- **Migration:** Moving runtime state as visibility or resources change.
- **Routing-aware:** How to find a path through the shifting ISL mesh?
- **Constraint-aware decisions:** Can deadlines, energy and thermal limits be respected?

Core question

How can we control over a moving, intermittent and resource-constrained orbital infrastructure?

Why classical cloud assumptions fail in orbit

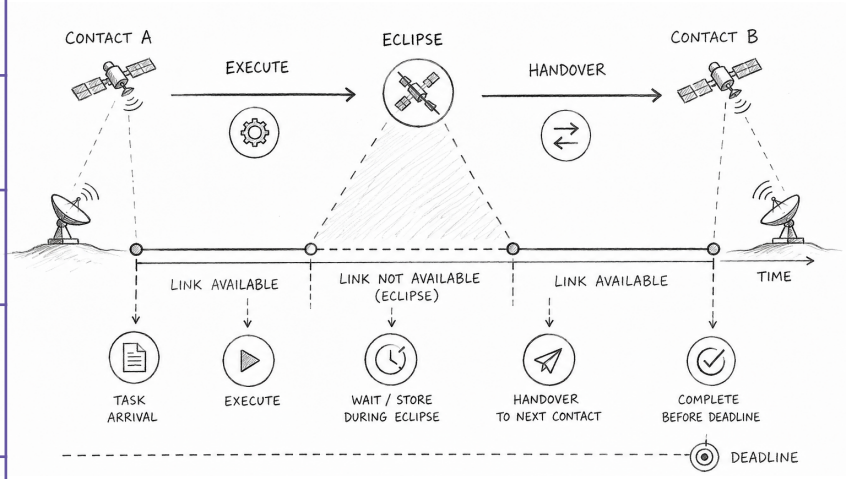
Terrestrial vs. Orbital Edge Computing

Feature	Terrestrial Cloud/Edge	Orbital Edge (LEO)
Reachability	Always reachable	Isolated for long durations
Connectivity	High Bandwidth / Stable	Intermittent / Dynamic (ISL/GSL)
Mobility	Static (Base Stations)	Highly Dynamic (7.6 km/s), predictable ISL handovers
Energy	Grid-powered / elastic	Strictly bounded (Solar/Battery)
Scalability	Elastic (Auto-scaling)	Finite (Fixed Hardware Onboard)
Topological State	Known & centralized	Time-varying & Delayed State
Cooling	Active Convection (HVAC)	Radiative cooling only (Vacuum)

Takeaway: orchestration must be physics-aware — it must reason over time, motion, energy and uncertainty.

What makes orbital scheduling different?

Constraint	Scheduling implication
Orbital motion	The best node now may disappear before the task finishes.
Visibility window	Admission and offloading must anticipate eclipse, visibility loss and handover.
ISL dynamics	Routing and compute placement are coupled.
Energy cycle	Batteries and eclipse periods constrain long-running workloads.
Partial observability	Decisions rely on delayed and local state.
Harsh environment	Faults, radiation and recovery policies become part of scheduling.



In orbital cloud, infrastructure is a moving graph. Scheduling becomes a time-dependent graph optimization problem.

Challenge map: what makes the problem hard

Orbital dynamics

Contacts and topology vary

Resource scarcity

Compute, network, energy, storage and bandwidth are limited.

Partial observability

State is local, delayed and uncertain.

Mission criticality

Deadlines, safety, reliability

Heterogeneity

Different payloads, accelerators, batteries and trust levels.

Deployment realism

Thermal effects, radiation, faults and benchmarks are often missing.

Coupled layers

Compute, routing, storage and energy interact.

Research challenge: exploit predictable orbital motion while remaining robust to unpredictable traffic, queues, channels and failures.

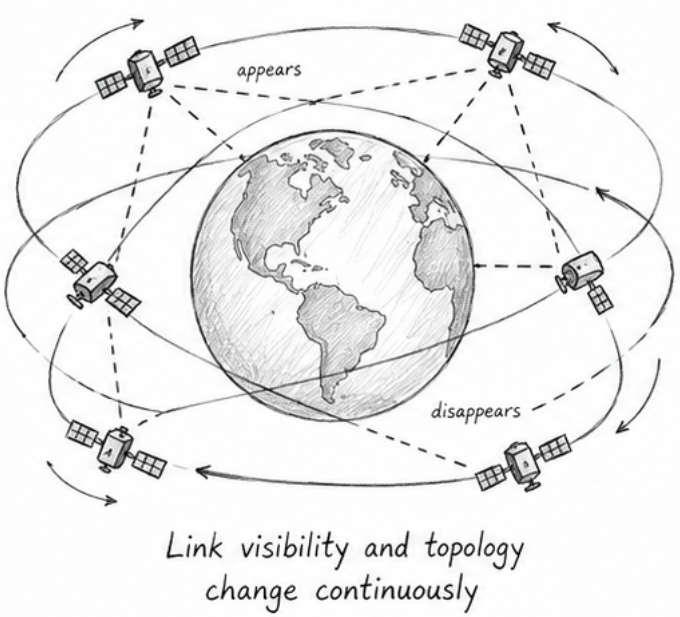
PART III: Problem modeling

Architecture, workloads, resources, dynamic topology, time-expanded graphs, compact formulation.

Modeling the moving infrastructure

To optimize orbital cloud services, we need to model:

- moving nodes
- intermittent contacts
- constrained resources
- multi-stage workloads
- deadlines and mission value
- uncertainty and partial information

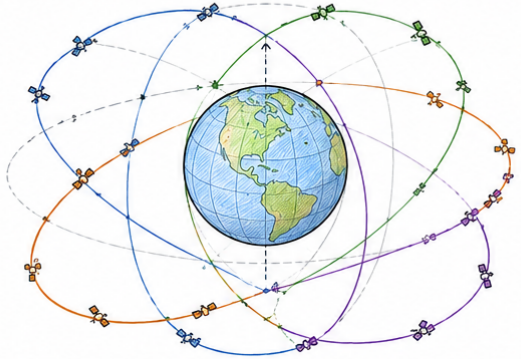


Constellation geometry: why topology is predictable

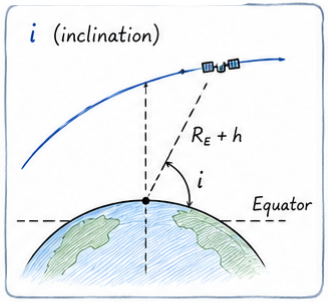
LEO topology is dynamic, but not arbitrary.

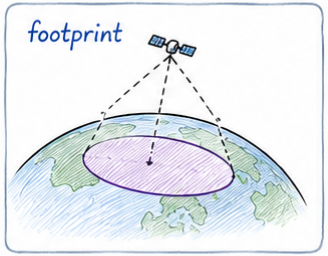
It is largely shaped by the constellation geometry.

Walker Δ — 4 of P orbital planes shown
Each colour = one orbital plane



Walker Δ notation: i° / T / P / F
inclination, total sats, planes, phasing.





Constellation	i / T / P	Alt.	Period
IRIDIUM NEXT	86.4° / 66 / 6	780 km	≈ 100 min
ONEWEB	87.9° / 648 / 12	1 200 km	≈ 109 min
STARLINK (Sh.1)	53° / 1 584 / 72	550 km	≈ 95 min
amazon | KUIPER	33–51° / 3 236 / 98	590–630 km	≈ 97 min

WHY GEOMETRY DICTATES EVERYTHING DOWNSTREAM

Plane count P →
coverage redundancy & cross-plane ISL density.

Inclination i →
which latitudes are visited and how often.

Phasing F →
relative position across planes ⇒ ISL handover timing.

Altitude h →
orbital period, contact length, latency, eclipse fraction.

Take-away.
A scheduler that ignores (i, P, F, h) ignores the very structure that makes the topology predictable.

Visibility windows and contact plans

Intermittency is not random; it is structured.

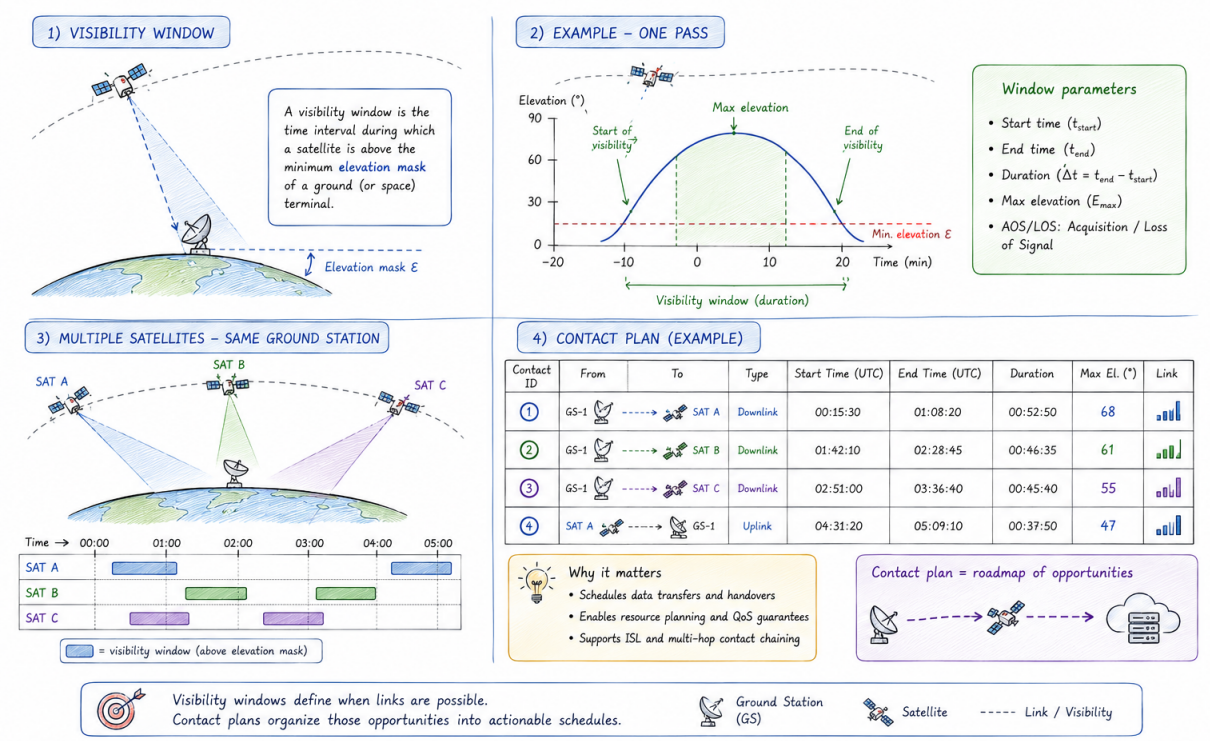
LEO satellites move relative to:

- ground stations
- users
- other satellites
- regions of interest

This creates time-limited contact windows.

Contact plan:

A predicted schedule of future communication opportunities.



The scheduler must plan ahead and execute within the window.

Modeling approaches in the literature

LEO networks are highly dynamic, but their motion is partly predictable.
Most models exploit this predictability by precomputing future connectivity.

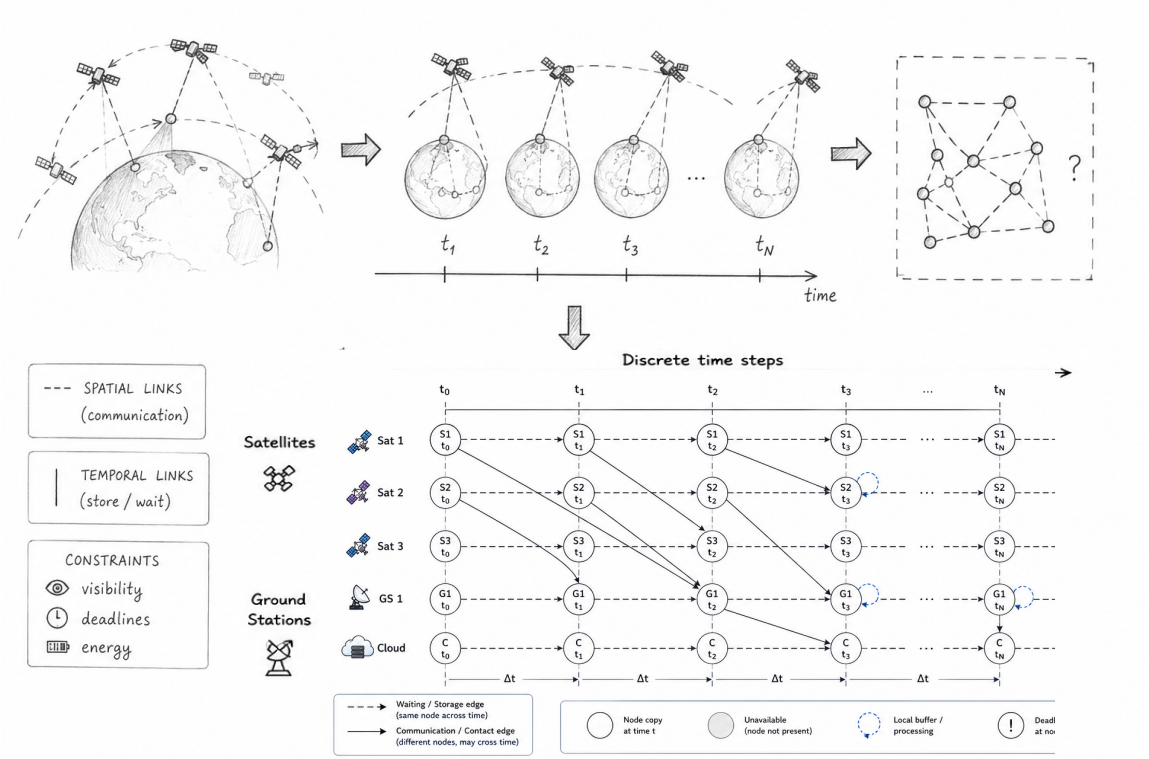
Two abstractions appear frequently:

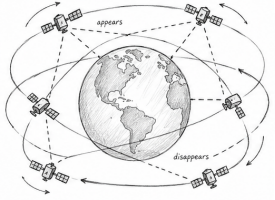
1. **Time-expanded orbital graph (TEG)**
2. **Contact Graph Routing (CGR)**

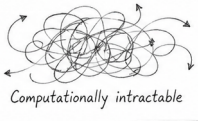
Core idea: These models transform orbital mobility into a time-dependent graph problem.

Modeling I: Time-expanded orbital graph (TEG)

Continuous network tracking is computationally intractable.







Idea: Converting Dynamics to Statics

The most common approach for LEO networks:

1. Discretize time into (Δt) snapshots.
2. Duplicate each node at each time interval.
3. Add spatial edges for communication links
4. Add temporal edges for storage / waiting
5. Encode visibility, deadlines and energy as constraints

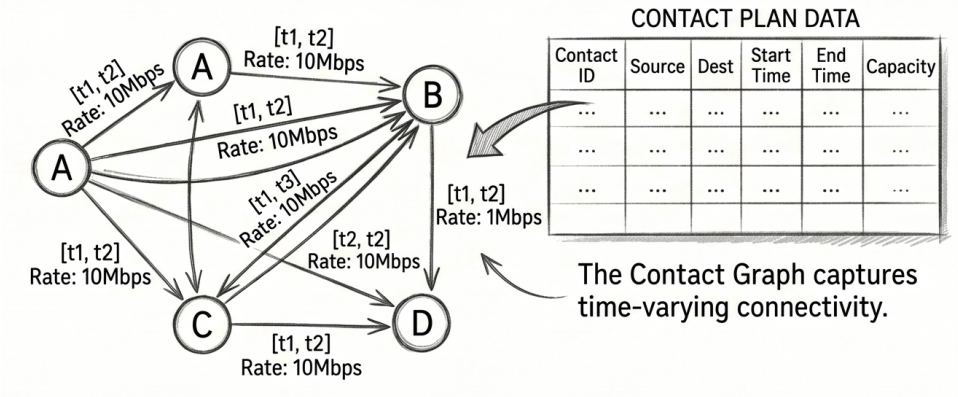
Strength: Good for joint scheduling, routing and storage decisions.

Limitation: State space grows rapidly with satellites × time slots × tasks.

Modeling II: Contact Graph Routing (CGR)

Instead of snapshots, CGR focuses on the **Contact Schedule.**

- Standard IP routing fails without stable end-to-end paths.

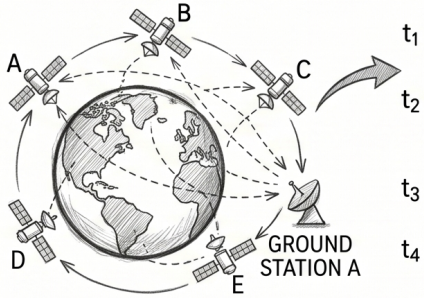


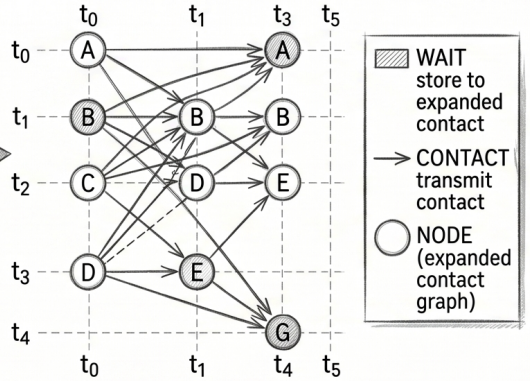
- **Predictable paths:** Orbits allow earliest-delivery paths to be computed over future contacts.
- **Contact Plan:** Future links are precomputed as contacts: source, destination, start, end, capacity.
- **Store-and-Forward:** Data is buffered until the next scheduled contact opens.
- **Multigraph Model:** Nodes have multiple edges between them, each representing a future communication window.
- **DTN-oriented model:** Multiple future contacts between nodes form a multigraph for Delay-Tolerant Network (DTN) routing.

Modeling II: Contact Graph Routing (CGR) - A DTN Perspective

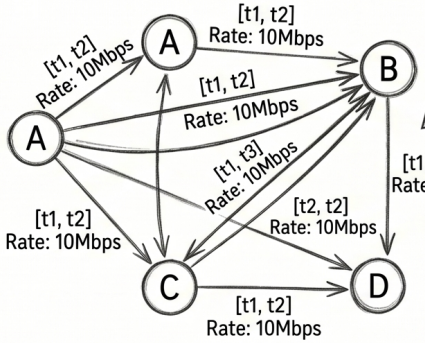
1 NETWORK TOPOLOGY & INTERMITTENT CONNECTIVITY

Nodes have scheduled contact intervals (Contacts).





2 CONTACT GRAPH GENERATION

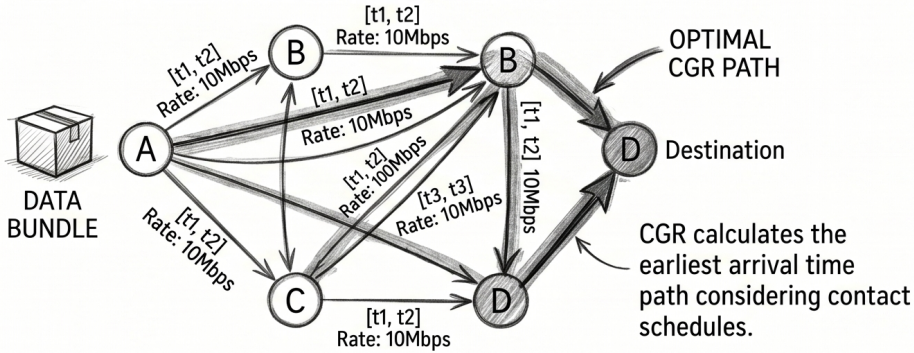


CONTACT PLAN DATA

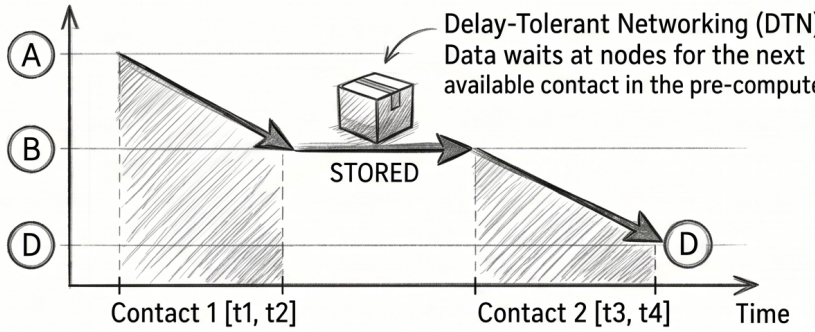
Contact ID	Source	Dest	Start Time	End Time	Capacity
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…

The Contact Graph captures time-varying connectivity.

3 ROUTING & PATH COMPUTATION



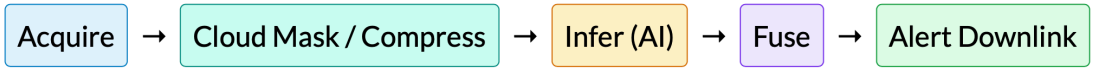
4 STORE-AND-FORWARD DELIVERY

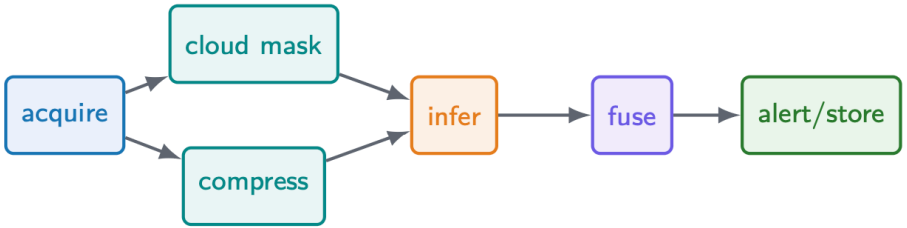


Workload Reality: DAG Pipelines

Tasks are rarely independent.

Real workloads are often DAGs (Directed Acyclic Graphs)





- Each stage has different input/output size, distinct data sizes, compute cycles, cost, and deadlines.
- **Implication for Scheduling:** Offloading a late stage (like inference) may be drastically cheaper energetically than offloading the raw acquisition.

Why this matters for scheduling

- Offloading a late stage may be cheaper than offloading the whole task.
- Precedence constraints interact with contact windows.
- The most valuable result may be an alert, not a full data product.

Compact problem formulation

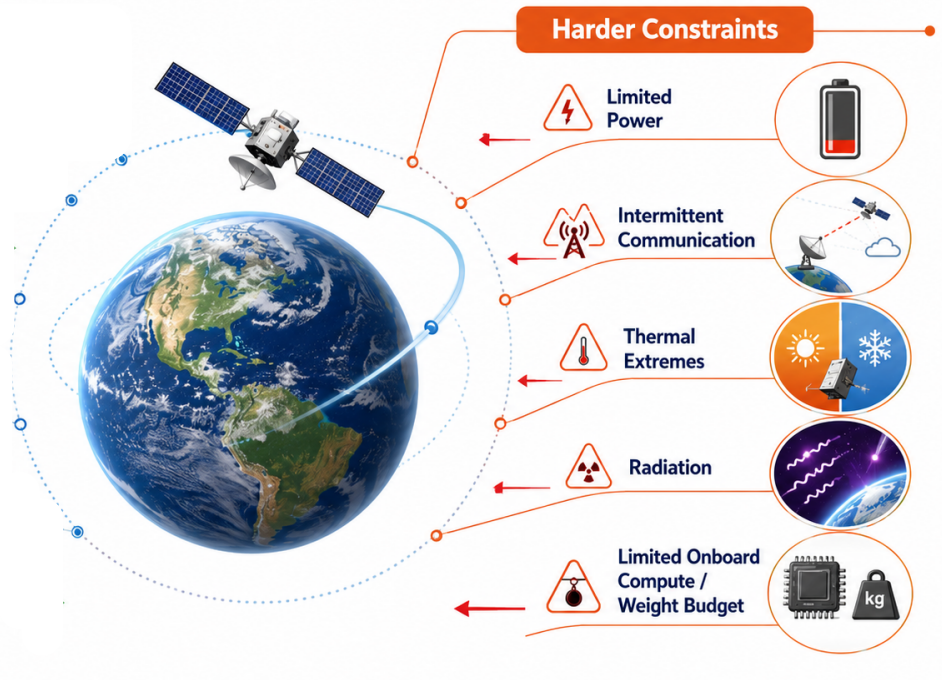
A compact orchestration model

For each task or DAG to deploy on the infrastructure graph, we need to decide:

- execution node
- start time
- bandwidth allocation
- storage/cache placement
- route
- CPU/GPU allocation
- fallback option

Subject to:

- visibility windows
- precedence constraints
- deadline constraints
- energy and thermal limits
- link capacity
- service availability
- trust and safety policies



Objective:

Maximize mission utility while minimizing delay, energy cost, risk and SLA violations.

The Control Plane Dilemma

Where should orchestration decisions be made?

- **Ground-based control (Centralized):**
  - better global view
  - easier optimization
  - but telemetry may be stale
  - vulnerable to gateway loss
- **In-orbit distributed control:**
  - fast local reaction
  - robust to ground disconnection
  - but partial and myopic view
- **Hierarchical orchestration:** ground planning + in-orbit adaptation.

PART IV: Optimization and orchestration approaches

Resolution & optimization approaches: exact, metaheuristic, learning, etc.

Method landscape

Main families:

1. exact optimization
2. decomposition and MPC
3. queueing and Lyapunov control
4. heuristics and metaheuristics
5. DRL, MARL and GNN-based policies
6. hybrid safe orchestration

No single family solves the full problem.

Method families overview

FAMILY	TYPICAL TOOLS	STRENGTH	LIMITATION
Exact optimization	MILP, MINLP, convex relaxation, CSP	clear formulation, global Optimality	scalability (NP-Hard) and centralization
Decomposition / MPC	ADMM, Lagrangian, receding horizon	uses predictable orbital dynamics	model accuracy and online updates
Queue / Lyapunov	drift-plus-penalty, stability control	online decisions and stability	hard to encode mission semantics
Heuristics	EDF, greedy, PSO, GA, ACO	fast, deployable baselines	weak guarantees, scenario tuning
DRL / MARL	DQN, PPO, TD3, CTDE, GNN, attention	adaptation under uncertainty	training complexity, safety, stability, explainability

The frontier is hybrid: optimization for feasibility, learning for adaptation, safety shields for deployment.

Exact optimization: useful but not enough

Tools: MILP, MINLP, convex relaxation, CSP

Why it matters

- Forces a precise model.
- Gives optimal or bounded solutions on small cases.
- Excellent baseline for algorithm evaluation.

Decision variables

$x_{kjt} \in \{0,1\}$ task k assigned to satellite j at t

$y_{k\ell t} \in \{0,1\}$ link ℓ used by task k at t

$f_{kjt} \geq 0$ compute allocation

$b_{k\ell t}, p_{kt} \geq 0$ bandwidth / transmit power

Multi-objective cost

$$\min \sum_k \left[\alpha L_k + \beta E_k + \gamma v_k + \delta AoI_k + \eta R_k - \rho U_k\right]$$

where $v_k = \mathbf{1}[L_k > D_k]$.

Typical constraints

$$\sum_k f_{kjt} \leq C_j(t)$$

$$\sum_k b_{k\ell t} \leq B_\ell(t)$$

$$E_j(t+1) = E_j(t) + H_j(t) - P_j(t)$$

$$x_{kjt} = 0 \quad \text{if } j \text{ is not reachable.}$$

Why it breaks

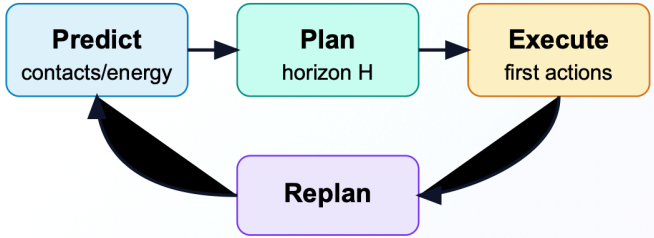
- Integer variables explode with satellites × tasks × time slots.
- Often assumes central state and simplified topology.
- Limited ability to react to unplanned queues or faults.

Use exact optimization as a microscope, not as the full operational orchestrator.

Model Predictive Control (MPC) and decomposition

MPC: exploit what is predictable

Orbital motion and contact windows are predictable.
Queues, traffic and link quality are not fully predictable.



Why it fits orbital cloud:

- short-horizon contact planning
- online updates
- energy-aware scheduling
- feasible fallback planning

Limitation:

Performance depends on prediction accuracy and solver speed.

Queueing and Lyapunov control

Queueing and Lyapunov: online stability

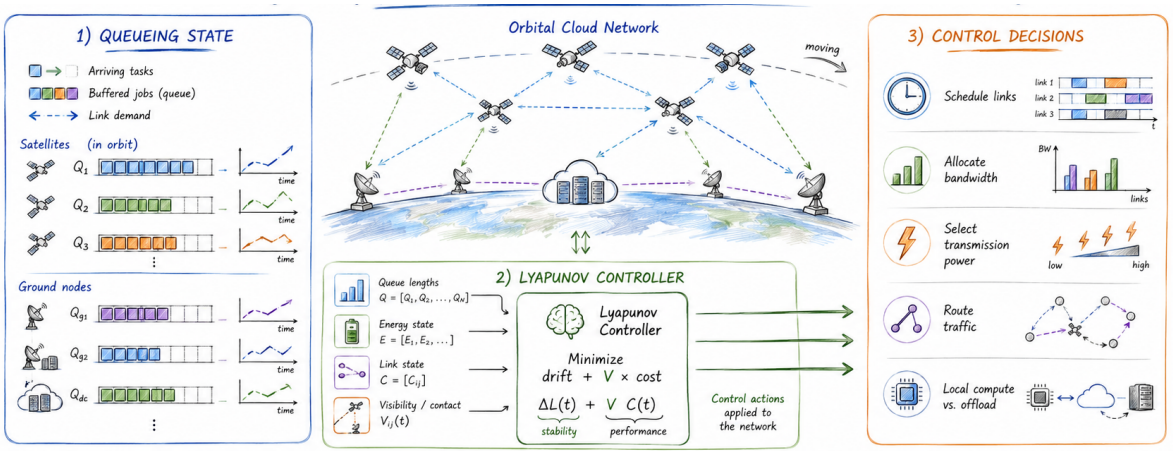
What it does well

- Controls queues without full future knowledge.
- Balances delay and energy over time.
- Supports online decision-making.

Where it struggles

- Mission value is difficult to encode
- DAG dependencies are harder to represent
- Deadline-critical semantics may be weak

Queue stability is necessary, but not sufficient for mission-aware orbital cloud.



Heuristics and metaheuristics

Heuristics: strong baselines

Rule-based heuristics:

- earliest-deadline-first
- nearest visible node
- battery-aware thresholds
- greedy placement

Metaheuristics:

- genetic algorithms
- particle swarm optimization
- ant colony optimization

Hybrid heuristics:

- local repair
- fallback rules

Why useful

Fast, simple, deployable, good baselines

Limitation

Scenario-specific tuning and weak guarantees.

DRL, MARL and GNNs

Learning-based orchestration

Useful for:

- Handles partial observability and changing traffic
- Manages Uncertain link quality (solar interference).
- Distributed agents match the constellation structure.
- GNNs can encode dynamic topology.
- RNNs can use histories and contact forecasts.
- The Agent learns the orbital pattern to predict when to buffer vs. when to transmit.

But deployment is hard

- Safety violations during exploration are unacceptable.
- Policies may fail under out-of-distribution traffic or faults.
- Explanations and certification are difficult.
- Sim-to-real gap is severe.

Learning can propose actions; feasibility checks should decide what is allowed.

The hybrid frontier

The most credible direction is hybrid.

A practical orbital orchestrator should combine:

- predictive models for contact and energy
- optimization for feasibility
- heuristics for fast fallback
- learning for adaptation
- safety shields for hard constraints
- distributed execution for resilience

Hybrid pattern	Example in literature	What is combined
Hybrid metaheuristic	Hu & Gong 2023	GA + Binary PSO
Optimization + MARL	Li et al. 2024	Optimization formulation + decentralized learning
GNN + DRL	Hu et al. 2024	Graph topology encoding + learned policy
Decision-assisted DRL	Huang et al. 2024	RL + feasibility guidance / hybrid action space

Policy pipeline:
Proposal → Feasibility projection → Safety shield → Deterministic fallback

Adaptivity without safety is not deployable.
Safety without adaptivity is too rigid.

PART V: STATE-OF-THE-ART

From isolated decisions to joint orbital cloud orchestration

State of the art and gaps

Existing works address important pieces of the problem: **offloading, routing, scheduling, resource allocation, learning and cloud-native platforms**.

However, full orbital cloud orchestration requires these decisions to be treated **jointly**, under orbital mobility, intermittent contacts, partial state and hard resource constraints.

The field is moving from isolated optimization problems toward joint orbital cloud orchestration.

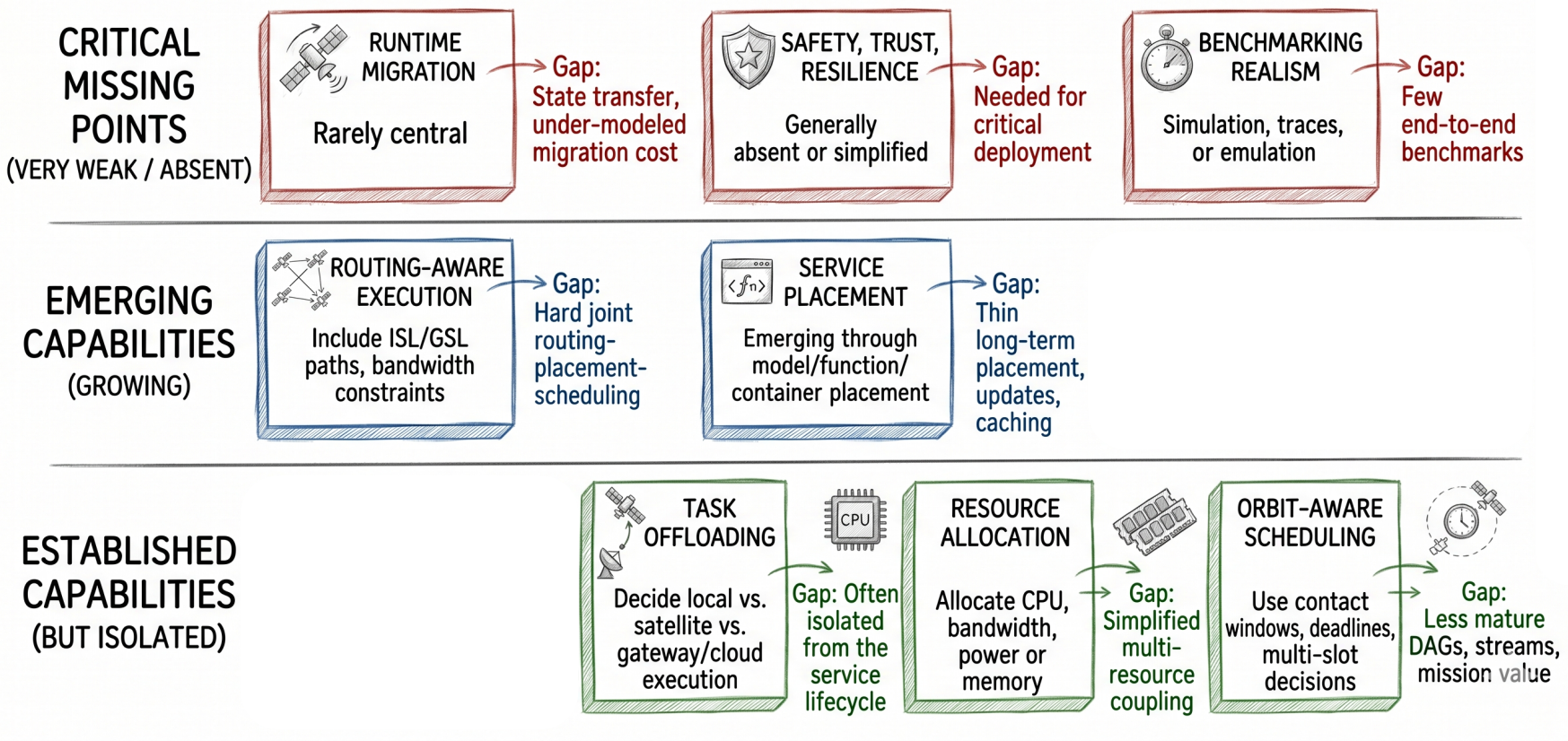
Evaluation lens: what do we compare?

How to classify a paper quickly

Question	Why it matters
What is the workload model?	Independent tasks, DAGs, streams and AI services require different schedulers.
What is the architecture?	Single satellite, multi-satellite, ISL, gateway and cloud layers define different problems.
Which orchestration decisions are included?	Placement, scheduling, offloading, migration, routing and resource allocation are often studied separately.
What information is available?	Global state, local state and partial observations lead to different control strategies.
Which resources are modeled?	CPU-only models are simpler; CPU, bandwidth, storage, energy and thermal constraints are closer to reality.
Which metrics are optimized?	Average latency can hide deadline failures, energy risk, unfairness and mission loss.
How is it validated?	Synthetic simulation, orbital traces, emulation and on-orbit evidence have different credibility.

A paper is not only defined by its algorithm, but also by its assumptions, decisions, metrics and validation setup.

Literature landscape by orchestration scope



Modeling realism: what is still simplified?

Modeling assumptions in current works

Modeling aspect	Coverage	Main simplification	Why it matters
Future contacts	◐	Contacts are predicted or partially known	Real execution sees errors, queues and failures
DAG workflows	◐	Many works still use independent tasks	EO/AI pipelines are multi-stage
Output flow	◐	Output size is simplified	Inference can shrink raw data into small alerts
Partial and stale state	◐	Simplified local/delayed telemetry.	Decisions rely on incomplete or outdated state.
Energy / thermal realism	◐	Static energy budget; simplified heat and battery dynamics.	Feasibility depends on power, heat, battery and eclipse cycles.
Faults / recovery	○	Mostly ignored	Critical missions need fallback policies

Metrics: from performance to mission value

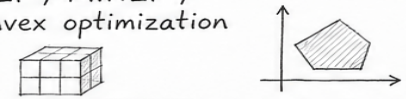
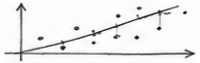
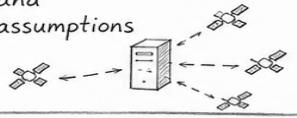
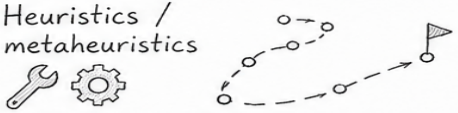
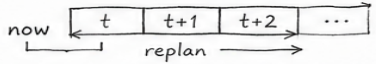
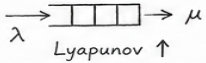
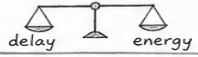
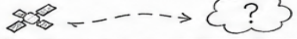
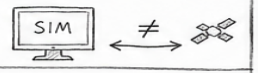
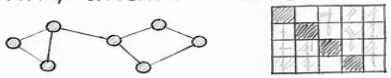
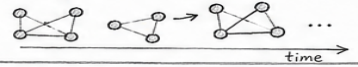
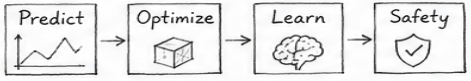
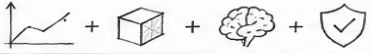
What gets optimized?

- ● well covered
- ◐ partially covered
- ○ under-covered

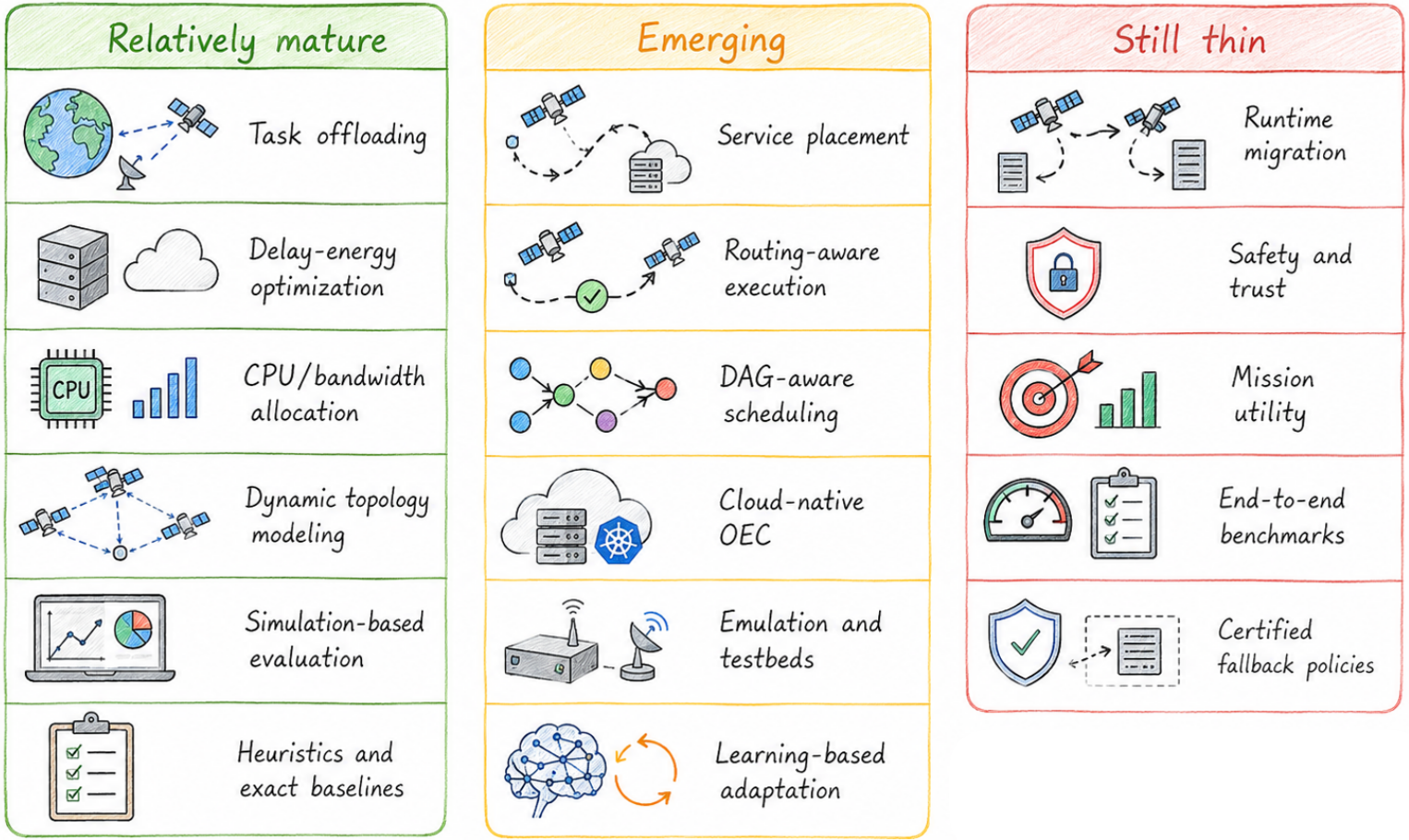
Metric category	Coverage	Common in literature	Still needed
Delay / latency	●	Mean latency, completion time	Tail latency and deadline satisfaction
Energy	●	Computation + transmission energy	Battery state, eclipse and thermal risk
Resource utilization	●	CPU, bandwidth, power	Joint compute-network-storage-energy trade-offs
Routing cost	◐	Path cost, ISL/GSL availability	Coupling with placement and migration
Deadline constraints	◐	Completion before due time	Mission-specific value of time
Freshness / AoI	○	Rarely central	Value decay and timeliness
Risk / safety	○	Usually absent	Hard feasibility and certified fallback
SLA / fairness / cost	○	Rarely modeled	Needed for multi-tenant orbital cloud

Performance-centric → Mission-value-centric

Method families: what they are good for

Method family	Maturity	What it is good for	Main limitation
MILP / MINLP / convex optimization 	●	Precise formulation and small-scale optimal baselines 	Scalability and centralized assumptions 
Heuristics / metaheuristics 	◕	Fast baselines and local repair 	Weak guarantees 
MPC / receding horizon 	◑	Predict-plan-replan under known contacts 	Solver speed and model accuracy 
Queueing / Lyapunov control 	◑	Online stability and delay-energy trade-offs 	Weak mission semantics 
DRL / MARL 	◑	Adaptation under uncertainty and local state 	Safety and sim-to-real gap 
GNN / attention models 	◑	Dynamic topology and history-aware decisions 	Training and certification complexity 
Hybrid safe orchestration 	○/◑	Prediction + optimization + learning + safety 	Still an open integration challenge 

Synthesis: what is mature, emerging, thin



PART VI: Research Directions & Open Challenges

Where the field is mature, where it is thin, and where a credible contribution can be made.

What the literature still under-covers

Workloads

Too many independent synthetic tasks; not enough DAGs, streams and mission value.

Resources

Compute-only models often ignore storage, accelerators, battery degradation and thermal effects.

Topology

Static or simplified contacts hide feasibility issues created by mobility.

Cloud ops

Service placement, updates, rollback, isolation and multi-tenancy are under-modeled.

Safety

DRL/MARL papers often lack hard feasibility checks and certified fallback.

Evaluation

No common benchmark combining TLE/SGP4, workloads, energy, traffic and baselines.

The opportunity is not only a new algorithm; it is a credible orchestration model plus a reproducible evaluation protocol.

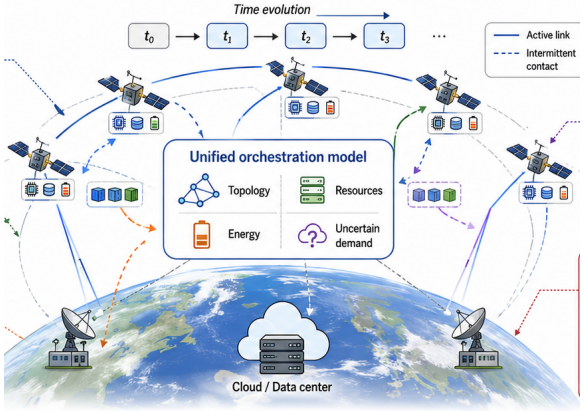
Candidate research questions

Research question	Possible approach	Expected contribution
How to jointly place, schedule, offload and route services under intermittent contacts?	Time-expanded graph + receding-horizon optimization	Feasible plans under moving topology
How to manage service placement and migration over time?	Multi-timescale placement + migration policy	Cloud-native orchestration across moving nodes
How to schedule DAG workflows with mission deadlines?	DAG-aware MPC + freshness/ value objective	Better deadline and mission-value guarantees
How to offload safely with local and delayed state?	Constrained MARL + safety shield	Adaptive but bounded decisions
How to trade mission utility against energy and downlink?	Semantic utility + multi-objective optimization	Transmit useful information, not raw bits
How to evaluate algorithms fairly?	Open benchmark with orbital traces and workloads	Reproducible comparison protocol

Research direction 1: Unified VLEO orchestration model

CORE IDEA

Model the constellation as a time-varying decision system coupling topology, resources, energy and uncertain demand.



Key ingredients

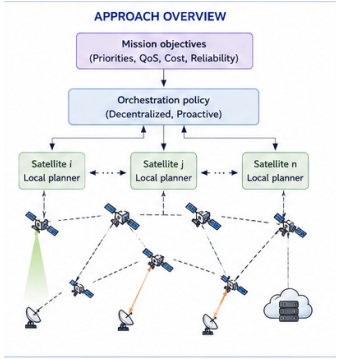
- contact graph / temporal dynamic graph
- onboard compute, storage, bandwidth and energy constraints
- service placement, task scheduling and migration variables
- partial and delayed state
- mission-driven workloads and priorities

Contribution. A unified modeling framework to reason about feasibility, autonomy and orchestration cost in VLEO constellations..

Research direction 2: Decentralized and proactive service orchestration

CORE IDEA

Design onboard and distributed mechanisms for **service placement, task scheduling and migration**, exploiting predictable orbital dynamics without relying on continuous ground supervision.



Decisions

- where to place services
- when to schedule tasks
- when to migrate runtime state
- how to coordinate neighboring satellites
- when to fallback or defer execution

Approach

- centralized baselines for reference
- short-horizon contact-aware planning
- decentralized / lightweight decision rules
- MPC or hybrid model-based mechanisms

Contribution. Embeddable orchestration algorithms for VLEO constellations under partial state and limited resources.

Research direction 3: Resilient orchestration without continuous ground supervision

CORE IDEA

Maintain useful service execution when ground access, links or satellites are degraded.

Failure modes

- unavailable or contested ground segment
- ISL disruption
- satellite failure
- demand spike
- degraded energy or communication conditions

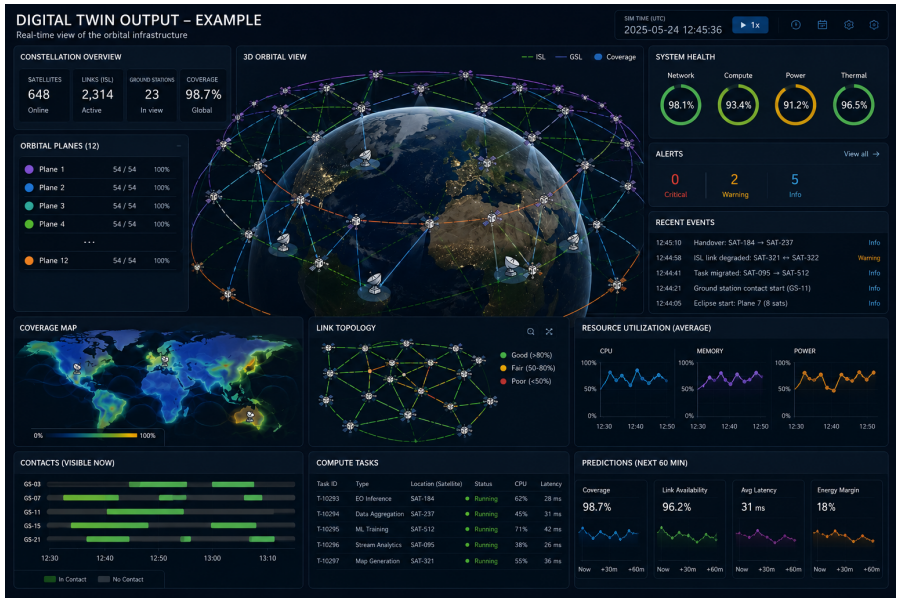
Approach

- topology-aware vulnerability analysis
- local re-placement and re-routing
- task re-prioritization
- fallback and degraded-mode policies
- explicit autonomy limits

Contribution. A resilience-aware orchestration strategy able to maintain mission utility under degraded VLEO conditions.

Research direction 4: Sim-to-real validation and digital twins

Developing directly in orbit is expensive and risky.
Orbital orchestration needs realistic test environments before deployment.



Validation stack

- **Orbital traces**
  TLE/SGP4, contacts, eclipse periods
- **Network simulation**
  ISL/GSL links, routing, bandwidth, delays
- **Resource and energy models**
  CPU, memory, storage, battery and thermal constraints
- **Workload generators**
  EO DAGs, streaming tasks, AI inference pipelines
- **Emulation / hardware-in-the-loop**
  Real software stacks on ground-based hardware or microVMs

A credible contribution needs not only an algorithm, but also a reproducible evaluation protocol.

VII. Conclusion

Final takeaways

Conclusion / Final takeaways

- **Orbital cloud is not “cloud in space”.**
  It is cloud-native orchestration under orbital physics.

- **The problem is joint orchestration.**
  Placement, scheduling, offloading, routing and migration must be optimized together.

- **The field is progressing, but still fragmented.**
  Current works address important pieces, but service lifecycle, safety, mission value and validation remain weak.

- **A credible contribution needs two things:**
  a realistic orchestration model and a reproducible evaluation protocol.

Thank You
For Your Atten

References

[1] 3GPP, “Non-Terrestrial Networks (NTN),” 3GPP Technology Overview, May 2024, updated Jul. 2025.

[2] Axiom Space, “Axiom Space to Launch Orbital Data Center Nodes to Support National Security, Commercial, International Customers,” Press Release, Apr. 7, 2025.

[3] B. Denby and B. Lucia, “Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System,” in Proceedings of the 25th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’20), Lausanne, Switzerland, 2020, pp. 939–954.

[4] European Space Agency, “Φsat: Artificial Intelligence for Earth Observation,” ESA Mission Information Page.

[5] Google Research, “Exploring a Space-Based, Scalable AI Infrastructure System Design,” Google Research Blog, Nov. 4, 2025.

[6] Y. Hu and W. Gong, “An On-Orbit Task-Offloading Strategy Based on Satellite Edge Computing,” Sensors, vol. 23, no. 9, Art. no. 4271, 2023.

[7] H. Li, J. Yu, L. Cao, Q. Zhang, Z. Song, and S. Hou, “Multi-Agent Reinforcement Learning Based Computation Offloading and Resource Allocation for LEO Satellite Edge Computing Networks,” Computer Communications, vol. 222, pp. 268–276, 2024.

[8] D. Magliarisi, E. Casalicchio, and V. Salvatore, “Orbit-Aware Task Scheduling in Satellite Edge Computing,” Cluster Computing, vol. 28, Art. no. 1013, 2025.

[9] Y. Qu, T. Zhang, Y. Feng, T. Xu, and Z. Guo, “Computation Offloading and Resource Allocation for E2E Tasks in Satellite Edge Computing Networks,” Space: Science & Technology, vol. 4, Art. no. 0144, 2024.

[10] M. Sun, J. Hou, K. Qiu, K. Wang, X. Chu, and Z. Zhang, “LLM-Based Task Offloading and Resource Allocation in Satellite Edge Computing Networks,” IEEE Transactions on Vehicular Technology, vol. 75, no. 3, pp. 5262–5267, Mar. 2026.

[11] Q. Tang et al. “Joint Service Deployment and Task Scheduling for Satellite Edge Computing: A Two-Timescale Hierarchical Approach,” IEEE Journal on Selected Areas in Communications, vol. 42, no. 5, pp. 1063–1079, 2024.

References

[12] Q. Xiu, J. Liu, X. Liu, and J. Wang, “Computation Offloading and Resource Allocation in Satellite Edge Computing Networks: A Multi-Agent Reinforcement Learning Approach,” Computer Networks, vol. 272, Art. no. 111680, Nov. 2025, doi: 10.1016/j.comnet.2025.111680.

[13] H. Yan, H. Huang, Z. Zhao, Z. Wang, and Z. Zhao, “Accuracy-Aware MLLM Task Offloading and Resource Allocation in UAV-Assisted Satellite Edge Computing,” Drones, vol. 9, no. 7, Art. no. 500, 2025, doi: 10.3390/drones9070500.

[14] Z. Yin, C. Wu, C. Guo, Y. Li, M. Xu, W. Gao, and C. Chi, “A Comprehensive Survey of Orbital Edge Computing: Systems, Applications, and Algorithms,” Chinese Journal of Aeronautics, vol. 38, no. 7, Art. no. 103316, Jul. 2025, doi: 10.1016/j.cja.2024.11.026.

[15] R. Zhang, Y. Feng, Y. Yang, X. Li, and H. Li, “Dynamic Delay-Sensitive Observation-Data-Processing Task Offloading for Satellite Edge Computing: A Fully-Decentralized Approach,” Remote Sensing, vol. 16, no. 12, Art. no. 2184, 2024, doi: 10.3390/rs16122184.

[16] L. Zhong, Y. Li, M.-F. Ge, M. Feng, and S. Mao, “Joint Task Offloading and Resource Allocation for LEO Satellite-Based Mobile Edge Computing Systems With Heterogeneous Task Demands,” IEEE Transactions on Vehicular Technology, vol. 74, no. 7, pp. 11337–11352, 2025, doi: 10.1109/TVT.2025.3549119.