Paper 034 / Application 011

Orbis Edge: Latency-Tolerant Artifact, Command, and Telemetry Networks for Orbital Autonomy

A local-first distributed systems architecture using JMP0X1B actors and effects

JMP0X1B Research Group

Working paper draft / June 13, 2026

Abstract

Orbis Edge is a proposed architecture for orbital and off-world autonomy where connectivity is intermittent, latency is unavoidable, and local decisions must remain safe. It combines actor supervision, typed command sequences, telemetry compression, artifact synchronization, fault detection, and provenance. The same pattern can support Earth disaster zones and remote agricultural infrastructure before orbital deployment.

Status. Draft manuscript for review and revision. This paper is not an empirical claim of new physics or field-ready technology. It is a structured proposal, theory note, protocol, or application architecture intended to be auditable, falsifiable, and publishable with source.

Contents

1 Problem

Cloud-native assumptions break under orbital latency, weak links, and partial failure. Systems still need command authority, telemetry, updates, and evidence records. The problem is to coordinate autonomous agents that can operate locally while later reconciling with ground systems.

2 Model

Nodes maintain local command queues, safety policies, state estimates, and artifact stores. Synchronization is digest-based and conflict-aware. Commands carry authority, expiry, preconditions, and rollback or safing behavior.

$$\text{Command} = \{a, \text{authority}, t_{\text{exp}}, P_{\text{pre}}, P_{\text{safe}}, H\}.$$

$$\text{Sync}(A, B) = \Delta(H_A, H_B) \rightarrow \text{merge or quarantine.}$$

3 Claims

1. **Claim 1.** Orbital autonomy should be designed as degraded-normal operation rather than exceptional failure handling.
2. **Claim 2.** Command expiry and authority labels are essential in delayed networks.
3. **Claim 3.** Digest-based artifact synchronization unifies telemetry, software releases, and research evidence bundles.
4. **Claim 4.** Earth deployments are valuable rehearsals for off-world constraints.

4 Evidence Plan

Prototype evidence should simulate orbital contact windows, corrupted packets, delayed commands, clock drift, and node isolation. Success is measured by safe command refusal, eventual artifact reconciliation, and explainable operator reports.

5 JMP0X1B Implementation Surface

JMP0X1B packages for command sequencing, telemetry, time, provenance, CCSDS/CFDP-style data handling, and actor supervision define the technology surface.

Illustrative JMP0X1B-style contract

```
record Command { action: Action, authority: TrustLabel, expires: Time, preconditions: List<Prec
actor EdgeNode {
    fn accept(cmd: Command) -> Gate effects{Authority, Time, Safety}
    fn sync(peer: NodeId) -> SyncReport effects{Network, Provenance}
}
```

6 Release And Review Plan

Release network simulations, command schemas, failure traces, and a reference safing-policy library.

7 Open Questions

- Which CCSDS-inspired behaviors should be built into libraries versus applications?
- How should time uncertainty affect command expiry?
- Can artifact sync handle classified or private payload data?
- What Earth pilot best validates orbital assumptions?

8 Conclusion

Orbis Edge applies mission-grade distributed systems to the long horizon of autonomous orbital infrastructure. It is practical now and extensible later.

References

References

[1] JMP0X1B. *Technology for Earth, designed with space as the operating condition.* `https://jmp0x1b.com/`

[2] JMP0X1B. *jmp0x1b Package Registry.* `https://libs.jmp0x1b.com/`

[3] JMP0X1B. *jmp0x1b_lang_spec package.* `https://libs.jmp0x1b.com/packages/jmp0x1b_lang_spec`

[4] JMP0X1B. *jmp0x1b_provenance package.* `https://libs.jmp0x1b.com/packages/jmp0x1b_provenance`

[5] Consultative Committee for Space Data Systems. *Reference Architecture for Space Data Systems.* CCSDS informational reports and standards series.

[6] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for artificial intelligence. *Proceedings of IJCAI*, 1973.

[7] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley, 2002.