StarLoom: Resilient Control Plane for LEO Satellite Constellations

Vaibhav Bhosale, Ada Gavrilovska, Ketan Bhardwaj
Georgia Institute of Technology

Under submission at ACM Eurosys 2026 – Do not distribute

ABSTRACT

Low Earth Orbit (LEO) satellite networks are emerging as the next-generation communication and compute infrastructure. Despite their wide utility, they are fragile. The July 2025 Starlink outage exposed that a single failure on its centralized control plane has a constellation-wide blast radius and long recovery times. Resilient designs for terrestrial control planes are not suitable for LEO networks due to fundamental differences in the infrastructure. In LEO network infrastructure, control ground station (GS) are geodistributed and satellites are constantly moving, requiring them to change their configuration every few minutes. If not done correctly, it leads to loss of connectivity and can take down the entire cluster.

To address this, we present StarLoom, a resilient control plane design for LEO networks that minimizes the blast radius of control plane failures by leveraging the principle of disaggregated federation – a new principle we formulate based on observed properties of the state manipulated in the LEO control plane operations. StarLoom realizes the principle through a disaggregated coordination service that separates global liveness tracking from local configuration management, making it feasible to ensure resilience (consistency in the control plane) while avoiding the need for untenable consensus across all GSs. In addition, StarLoom introduces a configuration management layer spanning both GSs and satellites, which enables seamless configuration changes such as updating satellite-GS mappings as satellites move. Together, these mechanisms reduce the blast radius from the entire constellation with 1 failure to less than one-third with 10 concurrent failures while ensuring recovery can happen up to 300× faster than the current centralized baseline.

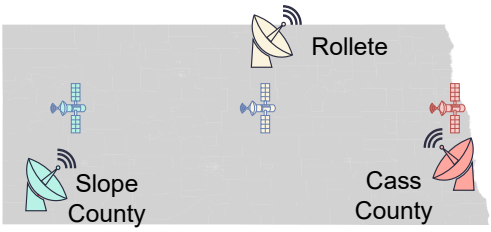


Figure 1: Map of North Dakota with the three Starlink ground stations. The ground stations and satellites are color coded to depict the ground stations managing the satellites.

1 INTRODUCTION

In July 2025, Starlink suffered a global multi-hour outage that disrupted broadband and cellular backhaul services worldwide. Users experienced packet loss, high latency, and session resets across regions, effectively rendering the system unreachable [40, 70]. The severity of this incident is amplified by the fact that Low Earth Orbit (LEO) satellite megaconstellations [7–10, 13] are no longer just communication infrastructures, but are rapidly evolving into a *LEO Compute Cloud*. These constellations now support not only broadband Internet and cellular connectivity [33, 42, 49, 52, 54, 73, 74], but also IoT services [5], machine learning inference [26, 27], with the goal of eventual space datacenters [6, 21, 56].

Despite its wide utility, the infrastructure remains fragile. The attribution of the July 2025 outage to a faulty update to the centralized control plane by both Starlink [50] and an independent third-party analysis [70] exposes the lack of resilient control plane systems that underpin this new generation of LEO infrastructure, a problem that has been explored across multiple contexts in terrestrial systems.

The currently employed centralized control plane has its roots in the legacy mode of *mission-specific, a priori established control plane* [34, 63] in traditional satellite systems. They were designed for scheduling satellite-ground station (GS) contacts (time windows when a satellite can communicate with a GS) with a handful of satellites and GSs in specific geographical regions. The same approach is now being retrofitted to handle thousands of satellites, hundreds of GSs and millions of user terminals (UTs), worldwide. Current LEO networks rely on automation while keeping the same centralized control plane. A concrete example is Starlink employing centralized coordination to manage satellite-GS contacts across the constellation [37, 53, 67]. Prior work on scheduling applications [19] and their state [60] also assume a centralized control plane. While such a design has worked so far, a single failure in the centralized control plane leads to a global outage like what we observed in the 2025 outage.

Terrestrial systems have explored multiple approaches to mitigate the shortcomings of centralized control planes at scales that are multiple orders of magnitude larger than LEO

networks. Resilience in terrestrial control planes is achieved by dividing the cluster management functionality among multiple control nodes. For instance, systems such as Twine [66] and Kubernetes federation [28] operate statically-defined disjointed, federated clusters and thus limit the blast radius of failures. This assumption fails for LEO networks where satellites change their control GS every few minutes/seconds. Current systems would treat each such change as a join/leave event resulting in multiple seconds of downtime for every satellite. While prior work [18, 19] has looked at addressing this downtime for applications in LEO network nodes using handovers, satellite configuration changes such as changing the control GS for a satellite are much different. Unlike application handovers that rely on a stable underlying network topology and control plane, configuration update of a satellite's control node requires physically changing the control node that results in an updated topology and rearchitected control plane. These changes make the approaches taken to hide application unavailability such as provisioning a new application instance infeasible. Further, the availability impact in the application scenario is restricted to just that particular application. Changing the control node of a satellite affects all the network functions running on the satellite. This makes existing solutions unsuitable for LEO networks.

The lack of a suitable approach is due to the drastically different infrastructure of a LEO network. Unlike terrestrial cloud systems, LEO satellites are moving at speeds over 27,000 km/h with their control nodes (GSs) geodistributed across the Earth, i.e., not collocated with satellite – a key design assumption in terrestrial systems. Consider the example illustrated in Figure 1 with three Starlink GSs in North Dakota. As the satellite moves over the state, it is managed by different GSs starting with the Slope Country GS, then the Rollette GS and finally the Cass County GS. However, in the intermediate cases, the satellite can very well be managed by either of the two nearby GSs, requiring coordination between them to avoid any conflicting decisions. Consensus protocols such as Raft, Paxos, or even optimized approaches like TAPIR [76] or EPaxos [55] incur significant coordination delays making them unsuitable for LEO networks where these configurations change in minutes (or even seconds).

Addressing this requires a new *control plane design* used to manage a LEO network that allows seamless configuration updates enabling the satellite compute nodes to dynamically change their control GS nodes. A key insight is to *align control plane configuration updates with application and state handovers*. Starlink, for example, reevaluates scheduling decisions for applications (and their state) approximately every 15 seconds [37, 67]. Synchronizing configuration updates to this boundary avoids replication of the dynamic application and satellite control state, since it gets flushed and rebuilt at this interval, thus providing a natural checkpoint for deterministic reconfiguration. However, this necessitates the existence of uniform configuration management so that the configuration across all the GSs is consistent to avoid any conflicting control decisions being sent to the satellites. We observe that the frequently changing satellite-GS mapping can be computed locally and deterministically leveraging the predictable orbital motion and thus avoiding replication. On the other hand, the less frequently changing state such as GS liveness can be centrally replicated, using a ZooKeeper [35] like service. Further, even if this uniform configuration is achieved, such a control plane requires mechanisms that enables satellites to physically update their control nodes without any downtime for the network functions running on the satellite.

In response, we present StarLoom, a resilient control plane system specifically designed for the LEO network infrastructure. StarLoom introduces a novel *disaggregated federation* principle and a new LEO control plane architecture that reduces the blast radius of control plane failures by dynamically distributing the satellite nodes among all the GSs and dynamically changing their affinity to GSs along with satellite motion. StarLoom uses a *disaggregated configuration service* made up of a globally replicated GS liveness service tracking (and sharing) GS availability and a local deterministic configuration mapper computing the same satellite-GS mapping everywhere without any coordination. Finally, StarLoom incorporates a *configuration management layer* that enables satellites to seamlessly switch their control node without any downtime to the satellite or the onboard network functions. StarLoom achieves this using a lightweight configuration manager that runs on every satellite that preloads the necessary identities and credentials that enable the satellite's node agent (such as kubelet) to switch to the new GS without affecting the existing network functions.

The outcome is a control plane for LEO networks that limits the blast radius due to any control plane failures to only the satellites managed by the failing control nodes. In summary, this paper makes the following contributions:

- Introduces the principle of *disaggregated federation* to provide resiliency to LEO networks control plane systems (§3) and introduces a new LEO control plane architecture manifesting that principle.
- Presents the design for a new control plane system enabling the new architecture using a disaggregated coordination service and a lightweight configuration management layer on both the satellites and the GSs (§4).
- Experimental evaluation shows the resiliency of StarLoom in reducing the blast radius to less than a third of the constellation even with 10 concurrent failures, compared to the entire constellation affected with centralized designs, while mitigating any such failures in under 25 seconds (§6).

2 BACKGROUND AND MOTIVATION

LEO Networks Primer. LEO satellites orbit around the Earth at altitudes in the range 200-1600 km [25]. To maintain this lower orbit, these satellites travel at high speeds over 27,000 km/hr and hence are only accessible for 3-4 minutes from any point on Earth [20]. Recent advancements in manufacturing and launching of satellites [2, 16, 31] have led to a meteoric rise in the number of LEO satellites planned and deployed. Today, there are multiple LEO networks ranging from a few hundred to tens of thousands of satellites by different commercial players such as Starlink [10], Planet Labs [8], Kuiper [9], etc. These satellites are used for broadband internet, serving disaster affected areas, maritime and in-flight internet, cellular connectivity, IoT connectivity, disaster monitoring, etc. For most of our discussions in this paper, we use the Starlink constellation operating over 6700 satellites [11] along with more than 170 ground stations [3, 41].

Operating Model. The LEO networks consist of the satellites, ground stations (GS), and user terminals (UT). The satellites communicate with the GSs using Ka-band and with the UTs using the Ku-band wireless radio links, also called as Ground Satellite Links (GSLs) with 18-23 Gbps capacity per satellite [65]. Satellites also have inter-satellite links (ISLs) with 100-200 Gbps capacity [12]. A satellite is always connected to some GS (s) to provide connectivity as per the bent-pipe model [47] either directly using the GSLs or via the ISL hops. For instance, Starlink users in central/south Africa connect to GSs in Europe [39, 53]. Starlink divides its coverage region into hexagons of size 20 km, similar to Uber's H3 cells and makes orchestration decisions for a cell as a whole. Recent Starlnk patent approval [38] indicates the use of a global scheduling system which reevaluates connectivity decisions every 15 seconds, which has also been confirmed by network measurements [36, 39, 53, 67]. This is in contrast to the initial days of Starlink where GSLs were active as long as the satellite was within the elevation angle constraints [43].

July 2025 Outage. Starlink experienced a global 2.5-hour outage that left its services entirely inaccessible [40, 70]. Users worldwide reported severe packet loss, elevated end-to-end latency, and frequent session resets across both broadband and cellular backhaul links. These symptoms revealed that the disruption was not confined to particular regions or links but stemmed from a systemic failure affecting the entire constellation. The fact that the outage lasted longer than a satellite's orbital period strongly suggests a control plane failure, as the network was unable to direct or route traffic effectively. The recovery phase further reinforced this view, with user terminals reconnecting gradually in a staggered, region-independent manner [70].

Starlink later attributed the incident to a faulty software update in its centralized control plane [62], which manages real-time orchestration of handoffs and connections among UTs, satellites, and GSs, as well as overall traffic engineering and load balancing [37, 67]. While this architecture enables dynamic routing and seamless mobility, it also creates a single point of failure: a software bug or misconfiguration in the control plane can cascade rapidly across the entire system, producing the widespread packet loss and session churn observed during the outage.

3 DISAGGREGATED FEDERATION

Need for Disaggregated Federation. Centralized control planes, while effective in early LEO deployments, fundamentally introduce a single point of failure as seen in the July 2025 Starlink outage. Terrestrially, resiliency is achieved using federated (or decentralized) control planes, where clusters operate semi-independently and periodically reconcile global state. However, this approach is unsuitable for LEO networks where satellites continuously move and therefore change their control GS frequently. Even systems such as Twine [66] which support dynamic reconfigurations rely on manual triggers and thus do not scale for LEO networks' dynamism. Moreover, control plane configuration updates differ fundamentally from connection or application handovers studied in cellular [14, 69] or LEO cloud systems [18, 19]. Applications can mask downtime by spinning up new replicas, but a control plane update cannot tolerate ambiguity: each satellite must always be managed by exactly one GS. Therefore, prior work on hiding application unavailability isn't applicable for satellite configuration updates.

This design space reveals a fundamental tradeoff. While federation reduces the blast radius of failures, it imposes heavy coordination overhead. To keep a consistent view of satellite-GS assignments across nearly 200 geodistributed GSs, naive federation requires frequent coordination due to the rapidly changing configurations. Without the coordination, multiple GSs could attempt to manage the same satellite, issuing conflicting commands. These coordination requirements make existing designs infeasible for LEO networks.

Key Insight. We observe that the control state can be disaggregated into two components – the liveness state of the GSs which changes less frequently and the much more dynamic satellite-GS mappings. The satellite-GS mappings can in fact be deterministically computed based on the satellite position and the state of the available GSs. Therefore, instead of globally coordinating the consistent propagation of the updates, which becomes infeasible at scale, resiliency can be achieved by exploiting the determinism in this frequently changing state that allows each participating node to locally recompute its correct values. At the same time, state such as GS liveness is relative stable and can be replicated centrally

using a service similar to ZooKeeper [35]. These insights lead to the following principle:

> **Disaggregated Federation Principle:** *When frequently changing state can be deterministically computed using a shared source of truth, consistency can be preserved while requiring global coordination only for the (less frequently changing) shared source of truth.*

Conditions for DFP Applicability. The disaggregated federation principle applies under the following conditions:

- The frequently changing state can be rebuilt deterministically at each node using a shared source of truth.
- The shared source of truth can be efficiently coordinated.

Application of DFP to LEO Networks. Building on disaggregated federation, the LEO control plane provides resiliency from control plane failures in LEO networks. It does so by splitting the control state into two components: the less frequently changing state i.e., GS liveness is non-deterministic, but it updates slowly, hence it can be replicated centrally with low overhead while the frequently changing satellite-GS mappings state can be locally computed deterministically as it changes every few minutes/seconds. Ensuring that the computations are deterministic requires a consistent view of the GS liveness state, as well as the current position of the satellites. The satellite orbital parameters, which only update roughly once per day, allow nodes to predict satellite positions precisely at any instant and together with the centrally replicated GS liveness information, each GS can independently and deterministically compute consistent satellite-GS assignments. We present a deterministic algorithm for computing these satellite-GS assignments in Algorithm 1 using the set of live GSs and the satellite orbital parameters. This avoids global consensus on the frequently changing state, while ensuring correctness on all state.

Simply precomputing all possible mappings a priori is not feasible, as this requires an accurate and consistent view of the live GSs. Furthermore, with hundreds of GSs, the permutations of potential failures grow exponentially and distributing precomputed schedules slows down recovery after an outage.

In the context of LEO networks, this principle is quite powerful. By separating the less frequently changing state (e.g., GS liveness) from the one changing frequently (e.g., satellite-GS mappings) and treating each appropriately, the LEO control plane can maintain global consistency without relying on global consensus.

4 STARLOOM DESIGN

StarLoom is a LEO control plane system designed on the principle of *disaggregated federation* to minimize the blast radius of control plane failures in LEO networks. Instead of concentrating control in a single logical entity, StarLoom federates it across GSs while ensuring consistency in control operations using a lightweight global service (§4.1) and deterministic local decisions (Algorithm 1). It is designed to be integrated in existing LEO Network orchestration systems such as Kubernetes[1], allowing it to leverage the broad ecosystem of systems innovations from the Kubernetes community such as orchestration, scheduling, and workload management, while extending these mechanisms to support control plane systems for LEO networks.

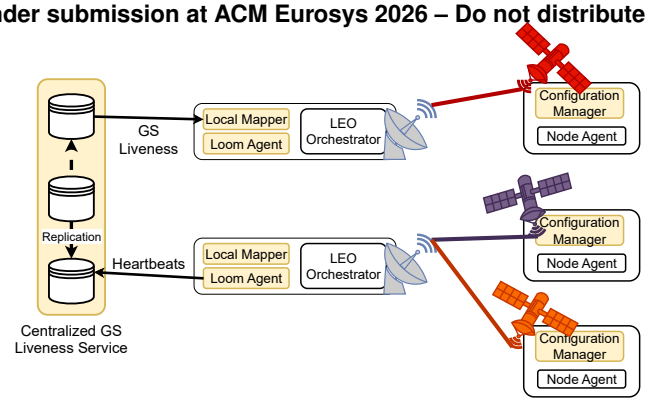


Figure 2: StarLoom Design.

Figure 2 illustrates the overall system design and its interaction with existing orchestration layers. At a high level, StarLoom consists of two components. (i) A new *Decentralized Configuration Service* that tracks and updates the satellite-GS mapping. It uses a Loom agent to centrally replicate the GS liveness using a ZooKeeper-like service, and a local deterministic mapper that uses the GS liveness to generate the satellite-GS mapping. (ii) A new *configuration management layer* enables satellites to frequently switch GSs in a seamless manner. Together, these components confine the impact of failures to the affected GSs rather than the entire constellation.

A central challenge in this design lies in the management of both satellite configuration and network function control state. Whenever a satellite changes its GS, the associated control state for onboard network functions (e.g., routing, scheduling) must also migrate to the new GS. However, doing so without affecting the network function availability requires coordination between the two GSs to ensure a seamless transition. Because different satellites from one GS may switch to multiple neighboring GSs simultaneously, these dependencies quickly cascade, necessitating every GS to share its control state with all its neighbors, and them, in turn, with all their neighbors. In practice, this would require global replication

[1]To the best of our knowledge, StarLink uses a custom orchestrator loosely based on Kubernetes [1] / Borg [72] to operate its control plane.

across hundreds of GSs with strong consistency, which is infeasible with the dynamic LEO networks.

To overcome this, StarLoom aligns configuration updates with the frequency of current state handovers used in practice. One such concrete data point is Starlink's remapping of satellites to new cells every 15 seconds. This has been empirically confirmed by multiple measurement studies [36, 39, 53, 67]. Aligning control plane updates with this interval ensures that the satellite's and its onboard network functions' control state does not need to be replicated at all. It is rebuilt deterministically at the next configuration update as part of the existing scheduling cycle. It's important to note here that while StarLoom performs the configuration updates every 15 seconds, it can be configured to be used with other update intervals as long as the configuration updates align with network function state updates.

StarLoom operates under a network model where satellites are reachable from GSs via a combination of GSLs and ISLs. If a satellite is within the line-of-sight of a GS, communication occurs directly over the GSL. For satellites not directly visible, connectivity is relayed through ISLs across neighboring satellites. In normal operation, only the GS currently assigned to manage a satellite communicates with it directly. However, in the event of a GS failure, the satellite remains accessible through an alternate GS (i.e., a DCS replica) via ISLs, ensuring continued reachability despite localized control failures. Further, StarLoom assumes that all the GSs and satellites synchronize their clocks using a time synchronization protocol like NTP [51]. Time synchronization is especially important here since the path models to predict a satellite's position require time as an input. While NTP protocol also experiences drift in the order of 100 ms, StarLoom operates at granularity of seconds, so this drift has minimal impact on its operation.

StarLoom is designed based on DFP, which ensures that only infrequently changing global state (such as GS liveness) is replicated. Frequently changing configuration state is either rebuilt or deterministically computed locally, thereby remaining consistent at each GS. By decoupling infrequent from frequent state, StarLoom avoids costly global coordination, stays within the sub-15s budget, while providing robust, failure-resilient operation. This is achieved through its two key components, described next.

4.1 Disaggregated Configuration Service

StarLoom introduces a *Disaggregated Configuration Service* (DCS) for resilient LEO control plane. At a high level, DCS maintains a consistent view of ground station availability and performs satellite-GS mapping locally. This prevents conflicting assignments, enables effective federation, and thus improves resiliency during failures.

A configuration update involves GSs recomputing which satellites they should manage. A common approach is for each GS to assign itself to the satellites that it can see. However, this leads to a lot of satellites that are not directly visible to a GS, but still need to be managed using ISLs, not having a control node. Alternately, GSs can claim responsibility to the k nearest satellites balancing the load. Both these approaches often result in conflicting mappings and hence require consensus mechanisms to resolve them. More importantly, this consensus step is one of the many steps that need to be finished within 15 seconds for all the sat-GS decisions to align configuration updates with network function state updates.

Traditional replication-based coordination protocols, such as Raft [58] and Paxos [46], ensure strong consistency but incur high latency when every configuration update must pass through a quorum. Even optimizations like Autobahn [32], TAPIR [76, 77], or EPaxos [55] still require coordination on overlapping or frequent updates, making them unable to meet the sub-15 second deadlines of LEO networks with hundreds of GSs. The large number of GS participants further amplifies these costs, as quorum sizes grow and coordination overhead scales poorly in such highly dynamic settings.

StarLoom instead introduces a *disaggregated Zookeeper -like service*. The key insight is that not all state changes in the LEO control plane are equally dynamic. Some state changes occur relatively infrequently (e.g., global GS liveness) and can be replicated centrally using existing mechanisms such as using a ZooKeeper service, while other state (e.g., satellite-GS configuration updates) needs to be updated every few seconds. Requiring consensus on these fast-changing updates would be prohibitively expensive. Therefore, StarLoom leverages deterministic orbital motion to compute mappings locally and consistently at every GS without coordination. Separating the global liveness state from the local mapping logic reduces the total latency in propagating updates. Maintaining consistent view of the GS liveness to all GSs, ensures mapping logic is executed deterministically everywhere. Mapping decisions are made locally, so failures in the liveness service affect only availability information and not every reassignment. This limits the blast radius in case of failures by enabling quicker detection and remedial actions, that can be synced with next update cycle in the worst case. Further, even in case of the failure of the liveness service, the blast radius is limited to concurrent GS failures rather than the entire constellation.

This design yields a *centralized coordinated state while disaggregating decisions*. Unlike centralized coordination systems such as ZooKeeper [35], which store all the state centrally, this approach avoids placing expensive coordination on the critical path. By centralizing only the less frequently changing GS availability while letting the mapper compute deterministic local decisions, the hybrid structure retains consistency without consensus on the critical path.

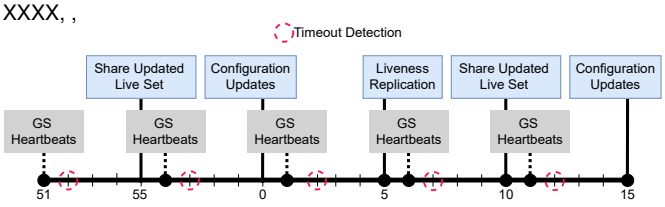


Figure 3: Illustrating the timeline of events happening driven by the GS liveness service.

GS Liveness Service. The liveness service is responsible for maintaining a globally consistent view of which GSs are alive. This service replicates its state globally, i.e., the state is geodistributed to ensure independent failure modes for all the replicas. This is similar to ZooKeeper's role in providing a reliable membership service, however, this is strictly limited to GS liveness.

As mentioned earlier, StarLoom performs the configuration updates in a 15 second cycle to align it with Starlink's network function state updates. Therefore, the rest of the values in this section are based on this 15 second interval and may need to be adjusted if the update interval changes. Each GS sends periodic heartbeats every five seconds to the nearest replica of the GS liveness service. These heartbeats are aggregated locally by the replica with failure detected when a GS misses two consecutive heartbeats. We illustrate the sequence of the events in Figure 3. GSs sends updates at times 1, 6, 11, and so on while the configuration updates are at times 15, 30, and so on. The liveness service replica waits for one second before timing out on the heartbeats. Further, in every 15-second cycle, the liveness service initiates replication with its other replicas at the five second mark and further shares the updated live GS set to all the GSs and satellites at the ten second mark. This ensures that there is sufficient time to perform the global replication of the liveness state and to share this updated state to all the GSs and satellites.

All the heartbeat messages as well as the messages sharing the liveness state are epoch'd based on current time. The liveness service responds to the heartbeat messages with acknowledgements to let the GSs and satellites know its liveness status. Further, it also has a retry mechanism if the liveness update messages to any GS or satellite are lost to ensure reliable delivery of these messages. Note that the messages used for both replication and sharing of liveness state are usually very small since only the delta is shared.

Since control plane failures are inherently rare, only failure detection state (and subsequent recovery) is replicated with all other GSs assumed to be alive. This ensures that updates to the GS liveness state are infrequent, ensuring enough time for sharing the updated state with the GSs and the satellites as well as computation of the sat-GS mapping.

Algorithm 1 StarLoom deterministic algorithm for computing the Satellite-to-GS mapping ensuring that no GS has more than an upper threshold T_{max} or fewer than a lower threshold T_{min} satellites, while prioritizing nearby GSs.

Require: GS_alive, S, TLEs for S, thresholds T_min, T_max
Ensure: Mapping $M : S \rightarrow GS$

```
Compute distance(s, g) for all s ∈ S, g ∈ GS_alive
for each satellite s do
    g_1 ← arg min_g distance(s, g)
    M[s] ← g_1
end for
Initialize load counters for all g
Identify overloaded and underloaded GS sets
Compute total deficit D = Σ_g max(0, load[g] − T_max) + Σ_g max(0, T_min − load[g])
for each overloaded g do
    for each excess satellite s of g do
        Build candidate set C from backup list with load < T_max
        Prioritize underloaded GSs, else choose nearest
        for each g' in C do
            If moving s to g' reduces D: reassign and update loads
        end for
    end for
end for
return M
```

Local Deterministic Mapping. Once the GSs and satellites receive the updated set of live GSs, they independently compute the correct satellite-GS mapping. The inputs are the current time, alive GS set, and satellite configurations for predicting their positions, all of which are globally known to all the satellites and GSs. Because the algorithm is deterministic, all participants converge to the same assignment without any need for explicit coordination. It is important to note that while the algorithm is deterministic, it still depends on the GS liveness set which is fetched from the centralized GS liveness service. And since the mapping runs locally, it operates at sub-second timescales.

Algorithm 1 shows the mapping algorithm used by StarLoom to assign satellites to GSs. The mapper enforces both an upper bound T_{max} and a lower bound T_{min} on the number of satellites per GS. The upper bound prevents overload on any single GS, while the lower bound ensures that each GS has at least some satellites assigned, avoiding cases where some GSs are left idle. This minimum threshold is critical for maintaining balanced utilization and ensuring that GS infrastructure is not stranded without traffic. It also provides a useful knob when deciding where to deploy new GSs. To maintain determinism

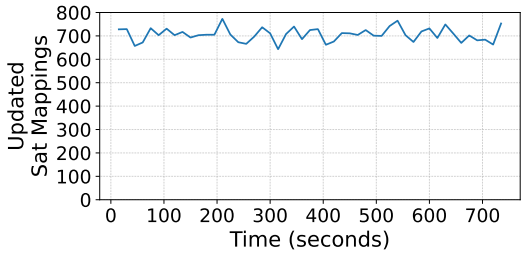
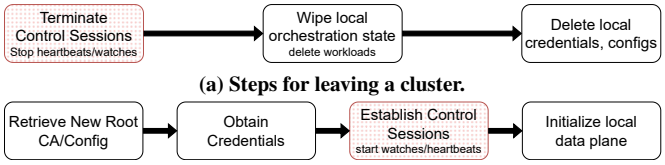


Figure 4: At every 15-second cycle, there are nearly 700 satellites that change their control node.



(a) Steps for leaving a cluster.

(b) Steps for joining a new cluster.

Figure 5: Showing the steps required for a node to leave an existing cluster and then join a new one. The highlighted steps are the only ones that StarLoom needs to perform since the credentials are locally available.

and fast convergence, the algorithm performs only a single cleanup pass. Mapping updates are accepted only if they strictly reduce a global deficit metric, and each satellite can be moved at most once per epoch. These constraints eliminate oscillations where mappings might otherwise bounce back and forth. Importantly, computing the nearest GS is not based on Euclidean distance but on actual routing paths through GSLs and ISLs [20, 44]. The framework can also incorporate additional policy constraints such as data sovereignty.

In summary, employing DCS reduces the burden of satellite reassignment by *limiting global coordination and using satellite prediction to compute consistent, deterministic mappings*. Next, we look at the system level mechanisms required to support these updates on both the satellite (compute node) and GSs (control nodes).

4.2 Configuration Management Layer

StarLoom introduces a configuration management layer atop DCS to enable satellites to switch their controlling GS every fifteen seconds as per LEO control plane design without incurring any satellite or application downtime. This layer provides the *missing systems mechanisms* necessary for continuous, dynamic reconfiguration. This is a sharp departure from state of the art terrestrial control plane systems where such configuration updates are rare and disruptive.

Terrestrial systems such as Kubernetes federation and Twine operate statically-defined disjointed, federated clusters that help limit the blast radius. While Twine in principle supports the idea of moving logical resources such as CPU/memory, it is a simple metadata change rather than the physical control node change required by LEO networks. In contrast, the LEO control plane demands configuration updates every 15 seconds, with nearly 700 satellites simultaneously switching their GSs (fig:mappingUpdates). Therefore, this necessitates that these configuration updates incur minimal overhead on the satellite as well as its onboarded network functions.

The limitations of existing mechanisms stem from assumptions in terrestrial systems. Node agents such as *kubelet* are bound to a single, immutable control node at join time. Updating this control node association requires the node to first leave its current cluster, since it can only connect to one control node at a time. Leaving the cluster entails several steps – terminating control plane sessions (watches, heartbeats), wiping local workloads and orchestration state, and deleting credentials and configs with the control node. Joining a new cluster is equally expensive. The node agent first retrieves the new cluster's root CA and configs, provision node credentials with the new root CA, starts control-plane sessions (watches/-heartbeats), obtains a Node Lease and creates a Node object (CSR submit/approve if required), and initializes CNI/kube-proxy state for the new cluster. In Kubernetes, node identity is tied to certificates that typically remain valid for up to a year [45], reinforcing the assumption of infrequent change. This combination of leave and join operations makes existing mechanisms infeasible for the rapid configuration updates demanded by the LEO control plane.

To address this, StarLoom runs a configuration manager on each satellite. The configuration manager is responsible for preloading and managing credentials as well as dynamically swapping them on the critical path of GS changes. Rather than regenerating identities at every reconfiguration, the satellite reuses prevalidated certificates and simply redirects its node agent (kubelet) to the new GS. The mapping algorithm from DCS helps compute the next GS deterministically and the configuration manager updates the identity of the control node at the node agent.

An important implication of this approach is its effect on the security perimeter. Because credentials are pre-generated for all satellite-GS pairs and stored onboard all satellites, the system shifts from per-update trust establishment to a model of pre-distributed trust. This reduces the overhead of certificate issuance on the critical path but also requires robust mechanisms for secure storage, rotation, and revocation of these credentials to ensure that a compromise of one satellite does not weaken the integrity of the broader constellation. Some approaches that can potentially help reduce the impact include provisioning certificates for only the next few potential GSs or only generating and storing certificates on each

satellite for a subset of GSs and use this static knowledge while making the mapping decisions. However, this requires efforts in ensuring that enough certificates are provisioned and are available ahead of time. We leave the exploration of this tradeoff for future work.

These new mechanisms make it feasible for configuration updates to occur transparently and deterministically every 15 seconds while avoiding any satellite downtime as well as expensive identity regeneration.

4.3 StarLoom Implementation

The current prototype implementation of StarLoom is built on top of Kubernetes. It uses the configuration parameters of all the satellites as two-line elements (TLEs) to predict the position of a satellite in the True Equator Mean Equinox frame. StarLoom is implemented as a Python package. The entire Python package consists of about 1200 lines of code. We will open-source StarLoom at the time of publication.

5 EVALUATION METHODOLOGY

The primary goal of our evaluation is to establish the effectiveness of StarLoom in reducing the blast radius during control plane failures. Our evaluation is structured around three key questions:

- How does StarLoom help reduce the blast radius during control plane failures (§6.1)?
- How long does StarLoom take to mitigate a control plane failure (§6.2)?
- How effective are the design choices made by StarLoom in enabling these benefits (§6.3)?
- What is the operational impact of using StarLoom in practice (§6.4)?

Key Metrics and Baselines. We focus on two core metrics. *Blast radius* defined as the number of satellites affected during a control plane failure and *recovery time* as the time taken for the system to recover from a failure and restore full control plane functionality. We compare the LEO control plane implemented by StarLoom against a *centralized control plane* design, which represents the existing approach in current LEO constellations such as Starlink.

Experimental Methodology. We combine controlled fault injection and trace-driven simulation, where the simulation is driven by real satellite orbital traces generated via the SGP4 model. We construct a model of the currently deployed Starlink constellation consisting of 6,400 satellites and 198 ground stations, with satellite trajectories computed using the standard SGP4 orbital model, as has been used in prior work [19, 27, 44]. We inject representative control plane faults (both ground station failure and coordination service crash) and measure their impact under both centralized and StarLoom deployments. We repeat these experiments across

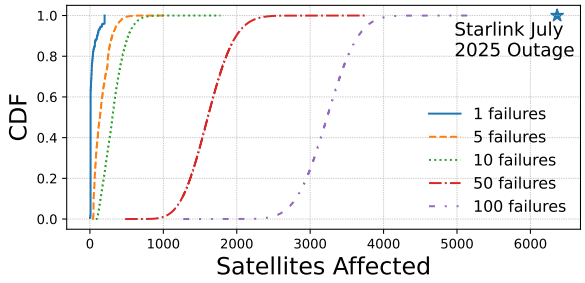
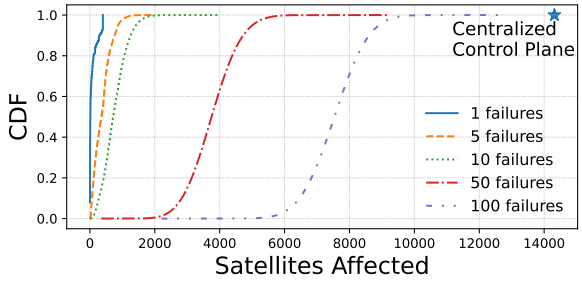
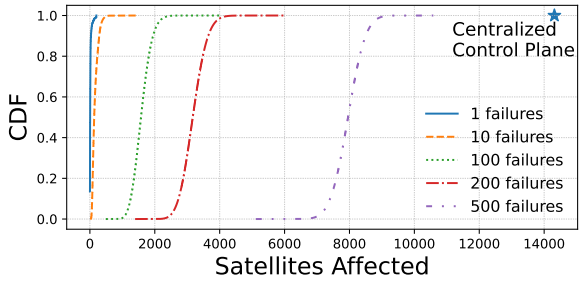


Figure 6: CDF of the number of satellites affected by different number of control plane failures with StarLoom. We compare the blast radius with the Starlink July 2025 outage that had a global outage.



(a) 14000 satellites, 198 GSs.



(b) 14000 satellites, 1000 GSs

Figure 7: CDF of the number of satellites affected by varying number of control plane failures with StarLoom in a future state with an increased number of satellites and GSs.

different constellation scales to evaluate sensitivity to constellation size and density.

6 EVALUATION

6.1 Reduction of Blast Radius Using StarLoom

We first evaluate how StarLoom reduces the blast radius of control plane failures. We generate these plots using a trace-driven simulation of the deployed Starlink constellation. From

these traces, we compute the satellite-GS mappings using StarLoom design (from Algorithm 1) with each GS managing at most 200 satellites and minimum 5 satellites. We then inject concurrent failures at randomly selected GSs to measure how many satellites lose their active controller. To generate the CDFs, we compute the sum of affected satellites in all permutations of the number of failures. Note that this number represents the maximum number of satellites concurrently affected by the failure. A smaller blast radius indicates stronger fault isolation and reduced systemic disruption.

Figure 6 shows the CDF of the number of satellites affected under different numbers of concurrent GS control plane failures. We compare these results with the Starlink July 2025 outage, which resulted in a global outage due to the failure of a centralized control plane. The contrast highlights the benefit of StarLoom: instead of a single failure impacting the entire constellation, failures are spatially and temporally localized, affecting only a small subset of satellites at any given time. We observe that even with multiple concurrent control plane failures, the proportion of satellites affected remains a small fraction of the constellation. For example, even with 10 concurrent failures, fewer than 2000 satellites experience disruption, compared to all the 6400 satellites impacted during the real-world Starlink outage. This demonstrates that StarLoom reduces the systemic blast radius by more than a factor of three even under worse conditions.

We further extend our analysis to a future state with larger constellations and expanded GS deployments, as shown in Figure 7. Even with 14,000 satellites, the distribution of affected satellites maintains the same trend. The blast radius grows gradually with more failures, rather than exhibiting catastrophic, system-wide collapse. Note that, since the number of satellites are nearly doubled with GSs staying the same (Figure 7a), StarLoom mapping caps each GS to 400 satellites compared to the earlier 200 satellites. Notably, scaling the number of GSs enhances resilience by distributing control responsibilities more evenly, thereby containing the impact of any localized failure (Figure 7b).

Impact of GS Liveness Service. An important aspect is the potential failure of the GS liveness service itself. Because this service is disaggregated from the local deterministic mapping, its failure only exposes the system to the concurrent GS failures already in progress. Essentially, the blast radius is still bounded by the number of GSs that simultaneously fail. Without such disaggregation, a liveness service failure would cascade globally, causing an outage across the entire constellation.

In summary, these results demonstrate that the design of StarLoom significantly limits the blast radius of failures, providing robust fault isolation and ensuring that localized failures do not escalate into global outages.

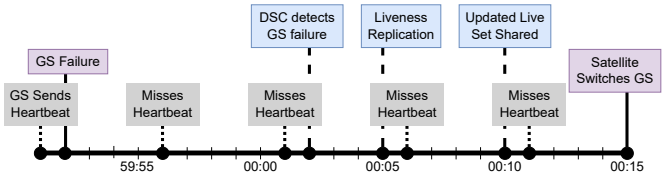


Figure 8: Showing how StarLoom helps mitigate a control plane failure within 25 seconds.

6.2 Effectiveness of StarLoom in Mitigating Control Plane Failures

We evaluate how quickly StarLoom recovers from a ground station (GS) failure. Our emulation (similar to Celestial [59] and Krios [19]) uses a CloudLab cluster of m510 nodes equipped with eight-core Intel Xeon 2.0 GHz processors and 64 GB of memory [4]. Three nodes host replicas of the Disaggregated Configuration Service (DCS), placed in Ashburn, Virginia (US), Frankfurt, Germany (Europe), and Singapore (Asia) to achieve geodistribution. We emulate three GS control nodes located in Slope County, Rollete, and Cass County in North Dakota, each running the Kubernetes control plane stack. A separate node acts as a satellite, provisioned in advance with certificates from all three GSs.

To model realistic network conditions, we assign wide-area latencies between GSs and DCS replicas as twice the c-latency (the time light takes to traverse a geodesic fiber link), consistent with empirical Internet measurements [17, 22]. Satellite-GS latencies are continuously updated using the SGP4 orbital path model, and GSL or GSL-ISL delays are generated in line with prior work [19, 59].

Figure 8 shows the timeline of events when the Slope County GS fails. At 59:51, the GS sends its last heartbeat to its DCS replica in Ashburn, but experiences a failure soon after. By 59:56 the node has stopped sending heartbeats, and within the next 6 seconds the Ashburn replica detects the failure after two consecutive missed heartbeats. At 00:05, the Ashburn replica replicates the updated liveness to Frankfurt and Singapore. At 00:10, the DCS service disseminates the updated live set to all GSs and satellites, and by 00:15 the satellite reassigns itself to a neighboring GS. The end-to-end recovery from GS failure to satellite reconnection completes in 25 seconds.

This is in sharp contrast to the July 2025 Starlink outage, where a faulty update to a centralized control plane left the system globally unreachable for 2.5 hours. While Starlink's recovery mechanisms remain proprietary, the scale of disruption is consistent with centralized architectures. A single failure must propagate changes globally before service can resume. In contrast, StarLoom relies on disaggregated federation and deterministic local mapping, ensuring that GS failures remain locally scoped and are masked by rapid reconfiguration.

Approach	# Consensus Decisions
Classical Consensus (e.g., Paxos)	6400
Optimized Consensus (e.g., EPaxos)	1205
StarLoom Deterministic Mapper	0

Table 1: Comparison of consensus decisions required for satellite-GS mapping updates during every 15-second cycle across different approaches.

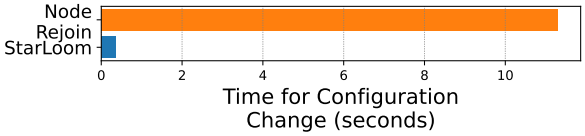


Figure 9: Comparing the time taken for configuration change for a single satellite by StarLoom with the traditional node leave/join latency in Kubernetes. Importantly, StarLoom implements this change without impacting onboard applications.

It is also important to note that not all satellites mapped to the Slope County GS experience the full 25 second disruption. Satellites already scheduled to switch GSs at 18:00:00 would have transitioned seamlessly to another GS and seen just 8 seconds of disruption. Similarly, satellites expected to be managed by Slope County starting at 18:00:15 would have already received the updated liveness information and chosen a different GS. Thus, the 25-second delay represents the upper bound, corresponding to the worst-case scenario of a satellite actively managed by the failed GS at the time of failure.

Another non-intuitive aspect of the recovery process is that the *25 second recovery time is constant per failure event*, and does not grow linearly with the number of failures. For example, if several GSs fail simultaneously, all affected satellites recover within the same 25 second window. However, if these failures are staggered, the recovery time depends on their with the 15-second cycle. Even then, each affected satellite reconnects to another GS within 25 seconds of its own failure event. In the worst case of chained failures, where the fallback GS fails as satellites attempt to join, and the next fallback fails too, the recovery time scales linearly. Yet even under such an extreme scenario with ten consecutive chained failures, recovery completes within 250 seconds. This bounded behavior is crucial for limiting overall service disruption during large-scale outages.

In summary, this experiment shows that StarLoom restores satellite connectivity to a healthy GS within 25 seconds of a control-plane failure. More importantly, combined with the blast radius observed in §6.1, StarLoom results in much smaller blast radius while recovering nearly 300 times faster compared to existing centralized solutions.

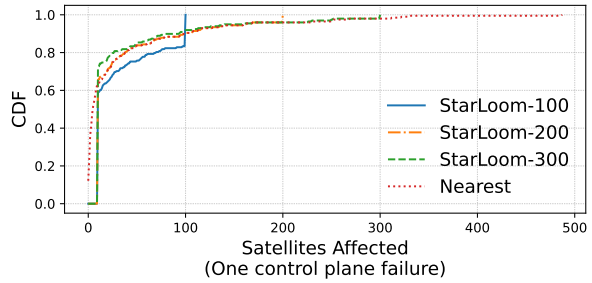
6.3 Effectiveness of StarLoom Design

Effectiveness of StarLoom in Reducing Configuration Updates Requiring Consensus. In classical consensus protocols, every satellite-GS mapping update requires coordination, resulting in 6400 consensus decisions per 15-second cycle. Optimized protocols such as EPaxos reduce this overhead by requiring consensus only for conflicting updates, thereby bringing the number down to 1205 satellites that are visible to multiple GSs. We assume that these conflicts would not exist for satellite managed using ISLs as the ordering them by path distance would always give a single GS. StarLoom goes further by using a deterministic mapper that computes this mapping locally and consistently across GSs and thus eliminating the need for consensus for making configuration updates. Note that all the three scenarios still require additional mechanisms to share the liveness state on which the satellite-GS mapping depends which is not included in this benchmark. We show the results in Table 1.

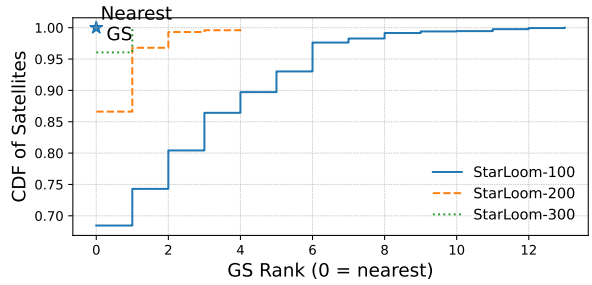
Effectiveness of StarLoom Configuration Change Mechanisms. We evaluate the latency of reconfiguring a satellite node to switch from one control node to another. We compare StarLoom mechanism with the conventional approach where a worker node (in this case, a satellite) explicitly leaves one cluster and rejoins another involving a full node teardown and bootstrap, including certificate verification and state reconciliation, while StarLoom exploits the fact that the satellite already holds valid credentials trusted by both the control nodes. We perform both sets of measurements thrice and present the averages in Figure 9. The leave/join approach incurs $\sim$11 seconds of downtime, whereas the StarLoom reduces this to under half a second. More importantly, the leave/join approach requires terminating existing applications on the satellite while StarLoom's minimal approach does this without affecting the onboard applications.

Effectiveness of StarLoom Mapping. Figure 10 illustrates the coupled tradeoff that StarLoom makes in bounding failure impact while preserving proximity. The blast-radius CDF (Figure 10a) shows that a purely nearest-based assignment creates highly skewed load, with some GSs managing hundreds of satellites. Failures at those GSs translate into very large outage radii. By introducing a load threshold, StarLoom redistributes satellites more evenly, sharply cutting off the heavy tail of failures. With thresholds of 200 or 300, over 90% of single failures impact fewer than 100 satellites.

The cost of this load balancing is visible in the rank-CDF (Figure 10b). When thresholds are tighter (e.g., 100), many satellites can no longer be served by their nearest GS and are reassigned to their 2nd or 3rd ranked choice. As this threshold increases (200-300), the assignment approaches the nearest-only case, reducing the fraction of satellites managed by non-nearest GSs, but at the expense of allowing somewhat larger



(a) Effect of a single failure with different variations of mappings.



(b) Choice of GS used in the mapping

Figure 10: Comparing the different sat-GS mappings. If only the nearest satellite is chosen, the effect of the failure

blast radii. Taken together, the two plots highlight the design tradeoff. Reducing the blast radius requires capping per-GS load, but this necessarily shifts some satellites away from their closest GS. StarLoom explicitly exposes and manages this tradeoff, choosing a threshold of 200 satellites per GS, achieving bounded failures while still keeping most satellites close to their top-ranked ground stations.

In summary, these results show that StarLoom substantially improves control-plane effectiveness by eliminating consensus from frequent mapping updates, enabling sub-second reconfiguration updates, and bounding the blast radius of failures through load-aware satellite-GS assignments.

6.4 StarLoom Overheads

Overheads on Satellite. While StarLoom enables seamless configuration updates for satellites update their GS, it does incur modest overheads on the satellite itself. The deterministic mapper, which runs locally to avoid coordination delays, takes on average 0.79 seconds per epoch to compute a new mapping. During this period, this process requires 1.69 cores on average while the maximum memory used is about 146 MB. This is well within the 15-second scheduling interval but still represents additional compute cost on a typically resource-constrained node. However, modern satellites have much larger compute resources available with each Starlink v2 satellite having upto 250 computer nodes [15].

In addition, storing the credentials and configuration artifacts required for switching across multiple GSs adds storage overhead. A satellite maintains a handful of kubelet client and serving certificates, along with their rotated copies, which in aggregate amount to about 16 KB per GS. For all the 198 GSs, this comes out to be 3 MB of storage overhead. Note that this overhead is much lower than the available storage on satellites that ranges in the order of hundreds of GBs or a few TBs [29, 61].

Overheads on Ground Stations. StarLoom introduces additional overheads on GSs too stemming from their expanded role in the control plane. Each GS hosts the entire control plane stack which incurs significant CPU (about 0.16 cores on average) and memory consumption (486 MBs on average). The GSs also send in heartbeat messages every five seconds and like satellites compute the satellite-GS mapping every 15 seconds. While these functions collectively increase the load on every GS, GSs are well-equipped to add some extra resources unlike satellites which have inelastic compute.

In summary, StarLoom provides resiliency to LEO networks while incurring minimal overhead on the satellites and GSs.

7 DISCUSSION

Need for Separate Service for Managing Liveness State. Instead of a centralized liveness service as part of DCS, one might consider a simple peer-to-peer mesh where GSs directly exchange heartbeats with one another. While this design may work with a small number of GSs, it does not scale to the increasing number of GSs (already in hundreds and still growing) that modern LEO networks require. The number of messages in such a mesh grows quadratically with the number of GSs, leading to excessive network chatter and wasted network capacity. By decoupling liveness into a lightweight global service, StarLoom ensures that the dissemination of availability updates remains efficient, scalable, and fault-tolerant, while avoiding the bottlenecks and fragility of full-mesh exchanges.

Other Control States Shared using DCS. Similarly, there is other control state that changes less frequently and can also be managed centrally without incurring high coordination costs. A good example is the load of all the workloads on a GS. This is particularly important if a GS is nearing the compute capacity on all the satellites it is managing and thus requires more number of satellites. While this will require some form of a resource negotiator [71], some simpler versions can be achieved by sharing such aggregate load information through the DCS to enable more informed mapping decisions without requiring to share this state on the critical path.

8 RELATED WORK

LEO Satellite Infrastructure. In addition to serving as the next generation communication infrastructure, recent work has explored using LEO satellite constellations as a compute infrastructure. This includes their deployment as space micro-datacenters [21], compute cloud [19], as well as proposals for a constellation-as-a-service [48] and a decentralized constellation [57]. Other efforts are targeted at specific application use cases, such as Content Delivery Networks (CDNs) [78], observational services and visual analytics [24, 26, 27], machine learning [68] and data compression [30]. Collectively, these systems highlight the need for resilient operations in control plane that StarLoom is designed to provide.

Orchestration Systems. Early cluster managers such as Borg [72] relied on centralized scheduling at scale. Twine [66] unified Facebook's fleet via global optimization across clusters, and Godel [75] presented production-scale unified resource management at ByteDance. Omega [64] introduced optimistic multi-scheduler concurrency over shared state, while Kubernetes generalized declarative control with reconciliation and extensibility [23]. All of these systems, however, target relatively stable datacenter environments with low-latency, high-bandwidth connectivity and thus, leveraged centralization as means to achieve resilience. For LEO settings, Krios [19] provides centralized scheduling mechanisms tailored to satellite motion, while Komet [60] coordinates frequent live service migrations but relying on underlying logical centralization for resilience. StarLoom presents an alternative, the first to best of our knowledge, demonstrating a resilient control plane using disaggregated federation as a principle.

Consensus Protocols. Replication-based coordination such as Paxos [46] and Raft [58] enforce strong consistency via quorum agreement per update, which we show is untenable for frequently changing satellite-GS mappings across hundreds of GSs. Even other approaches to reduce the cost of consensus fail for LEO networks. Autobahn [32] reduces coordination costs through speculative execution and splits replication into two logically distinct layers, but the consensus-based validation layer still introduces latency that cannot meet the sub-15 second deadlines of LEO configuration updates, particularly when hundreds of GSs simultaneously recompute assignments. TAPIR [76, 77] avoids consensus on some non-overlapping operations through weaker consistency semantics. However, in LEO networks, most updates are overlapping as multiple GSs target the same satellite and thus these updates force coordination almost every time, eliminating this advantage. EPaxos [55] exploits dependency tracking to avoid some quorum steps, but given that multiple GSs might end up claiming a given satellite, EPaxos requires consensus to avoid them causing delays. Consequently, while effective in datacenters, these protocols remain mismatched for LEO networks' rapidly changing updates. In that sense, StarLoom approach is unique design point in that space.

9 CONCLUSION

This paper presents StarLoom, a system designed to provide a resilient control plane for LEO satellite constellations. By leveraging the principle of disaggregated federation, StarLoom effectively minimizes the blast radius of control plane failures. The system introduces a disaggregated coordination service that separates global liveness tracking from local configuration management, enabling resilience without the need for untenable consensus across all ground stations. Additionally, StarLoom incorporates a configuration management layer that spans both ground stations and satellites, facilitating seamless configuration changes as satellites move. Our evaluation shows that StarLoom significantly limits the impact of failures, while recovering nearly 300 times faster than existing centralized solutions. The outcome is a system with robust fault isolation, where localized failures do not escalate into global outages. On a broader scope, StarLoom demonstrates the need for a fundamental rethink of the principles of system design when applied to the unique challenges of the LEO and space computing infrastructure.

REFERENCES

[1] Kubernetes. https://kubernetes.io/. (Accessed on September 25th, 2025).
[2] Launch Costs to Low Earth Orbit, 1980-2100. https://www.futuretimeline.net/data-trends/6.htm, 2018. Accessed on September 25th, 2025.
[3] FCC Selected Application for Space Exploration Holdings, LLC. SES-LIC2019090601171, 2020. Accessed on September 25th, 2025.
[4] CloudLab Hardware. https://docs.cloudlab.us/hardware.html, 2025. Accessed on September 25th, 2025.
[5] Lacuna Space. https://lacuna.space/, 2025. (Accessed on September 25th, 2025).
[6] Lumen Orbit. https://www.starcloud.com/, 2025. (Accessed on September 25th, 2025).
[7] OneWeb. https://www.oneweb.world, 2025. (Accessed on September 25th, 2025).
[8] Planet Labs. https://www.planet.com, 2025. Accessed on September 25th, 2025.
[9] Project Kuiper. https://www.aboutamazon.com/what-we-do/devices-services/project-kuiper, 2025. (Accessed on September 25th, 2025).
[10] Starlink. https://www.starlink.com, 2025. (Accessed on September 25th, 2025).
[11] Starlink. https://celestrak.org/NORAD/elements/table.php?GROUP=starlink&FORMAT=tle, 2025. Accessed on September 25th, 2025.
[12] Starlink - Technology. https://www.starlink.com/technology, 2025. Accessed on September 25th, 2025.
[13] Telesat. https://www.telesat.com, 2025. (Accessed on September 25th, 2025).
[14] Saddam Alraih, Rosdiadee Nordin, Asma Abu-Samah, Ibraheem Shayea, and Nor Fadzilah Abdullah. A survey on handover optimization in beyond 5G mobile networks: Challenges and solutions. *IEEE Access*, 11:59317–59345, 2023.

[15] Akash Badshah, Natalie Morris, and Matthew Monson. Over-The-Vacuum Update–Starlink's Approach for Reliably Upgrading Software on Thousands of Satellites. In *Proceedings of the AIAA/USU Conference on Small Satellites*, 2023.

[16] Leandra Bernstein. Affordability Is Transforming the Satellite Industry: How Each Segment Is Keeping Up with the Pace of Change. https://www.kratosdefense.com/constellations/articles/affordability-is-transforming-the-satellite-industry, 2022. Accessed on September 25th, 2025.

[17] Debopam Bhattacherjee, Waqar Aqeel, Sangeetha Abdu Jyothi, Ilker Nadi Bozkurt, William Sentosa, Muhammad Tirmazi, Anthony Aguirre, Balakrishnan Chandrasekaran, P Brighten Godfrey, Gregory Laughlin, et al. {cISP}: A {Speed-of-Light} internet service provider. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1115–1133, 2022.

[18] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. In-orbit Computing: An Outlandish Thought Experiment? In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, pages 197–204, 2020.

[19] Vaibhav Bhosale, Ada Gavrilovska, and Ketan Bhardwaj. Krios: Scheduling Abstractions and Mechanisms for Enabling a LEO Compute Cloud. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pages 322–340, 2024.

[20] Vaibhav Bhosale, Ahmed Saeed, Ketan Bhardwaj, and Ada Gavrilovska. A Characterization of Route Variability in LEO Satellite Networks. In *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings*, pages 313–342. Springer, 2023.

[21] Nathaniel Bleier, Muhammad Husnain Mubarik, Gary R Swenson, and Rakesh Kumar. Space Microdatacenters. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 900–915, 2023.

[22] Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. Why is the internet so slow?! In *Passive and Active Measurement: 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings 18*, pages 173–187. Springer, 2017.

[23] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[24] Zhuo Cheng, Bradley Denby, Kyle McCleary, and Brandon Lucia. Eagleeye: Nanosatellite constellation design for high-coverage, high-resolution sensing. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 117–132, 2024.

[25] Christopher S. Allen and Martina Giraudo and Claudio Moratto and Nobuyasu Yamaguchi. Chapter 4 - Spaceflight environment. In Tommaso Sgobba and Barbara Kanki and Jean-François Clervoy and Gro Mjeldheim Sandal, editor, *Space Safety and Human Performance*, pages 87–138. 2018.

[26] Bradley Denby, Krishna Chintalapudi, Ranveer Chandra, Brandon Lucia, and Shadi Noghabi. Kodan: Addressing the Computational Bottleneck in Space. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 392–403, 2023.

[27] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 939–954, 2020.

[28] Kubernetes Documentation. Kubernetes Federation. https://github.com/kubernetes-retired/kubefed, 2025. (Accessed on September 25th, 2025).

[29] Doug Mohney. Terabytes From Space: Satellite Imaging is Filling Data Centers. https://www.datacenterfrontier.com/internet-of-things/article/11429032/terabytes-from-space-satellite-imaging-is-filling-data-centers, 2020. (Accessed on September 25th, 2025).

[30] Kuntai Du, Yihua Cheng, Peder Olsen, Shadi Noghabi, and Junchen Jiang. Earth+: On-board satellite imagery compression leveraging historical earth observations. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 361–376, 2025.

[31] Warren Frick and Carlos Niederstrasser. Small Launch Vehicles-A 2018 State of the Industry Survey. 2018.

[32] Neil Giridharan, Florian Suri-Payer, Ittai Abraham, Lorenzo Alvisi, and Natacha Crooks. Autobahn: Seamless high speed BFT. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 1–23, 2024.

[33] Harry Menear. KDDI taps SpaceX's Starlink for rural cellular backhaul. https://mobile-magazine.com/wireless-networks/kddi-taps-spacexs-starlink-rural-cellular-backhaul, 2021. (Accessed on September 25th, 2025).

[34] Takashi Hiramatsu, Shusaku Yamaura, Tomomi Otani, Naoko Sato, Katsumi Morita, Toru Koyama, and Kikuko Miyata. UNIFORM-1 Ground System: Mission Planning, Scheduling, Control, and Data Processing. In *Proceedings of the AIAA/USU Conference on Small Satellites, Mission Lessons, SSC12-XII-1*, 2014.

[35] Patrick Hunt, Mahadev Konar, Flavio P Junqueira, and Benjamin Reed. ZooKeeper: Wait-free coordination for internet-scale systems. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[36] Geoff Huston. Starlink protocol performance. https://datatracker.ietf.org/meeting/118/materials/slides-118-iepg-sessa-starlink-protocol-performance-00, 2023. Accessed on September 25th, 2025.

[37] Geoff Huston. Navigating starlink's fcc paper trail, June 2024. (Accessed on September 25th, 2025).

[38] Jayasuryan V Iyer, Khasim Shaheed Shaik Mahammad, Yashodhan Dandekar, Ramakrishna Akella, Chen Chen, Phillip E Barber, and Peter J Worters. System and method of providing a medium access control scheduler, October 24 2023. US Patent 11,800,552.

[39] Liz Izhikevich, Manda Tran, Katherine Izhikevich, Gautam Akiwate, and Zakir Durumeric. Democratizing LEO Satellite Network Measurement. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(1):1–26, 2024.

[40] Al Jazeera. Elon musk 'sorry' after starlink satellite internet suffers global outage. https://www.aljazeera.com/news/2025/7/25/elon-musk-sorry-after-starlink-satellite-internet-suffers-global, 2025. (Accessed on September 25th, 2025).

[41] Jessica Watkins. Starlink Ground Station: Overview And Locations. https://starlinkspot.com/starlink-ground-station-locations, 2024. Accessed on September 25th, 2025.

[42] John Liu. Elon Musk makes India inroads with two major deals. https://www.cnn.com/2025/03/12/tech/india-reliance-jio-bharti-airtel-starlink-deals-intl-hnk/index.html, 2025. (Accessed on September 25th, 2025).

[43] Mohamed M Kassem, Aravindh Raman, Diego Perino, and Nishanth Sastry. A browser-side view of starlink connectivity. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 151–158, 2022.

[44] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. Exploring the "Internet from Space" with Hypatia. In *Proceedings of the ACM Internet Measurement Conference*, pages 214–229, 2020.

[45] Kubernetes Documentation. Certificate Management with kubeadm. https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubead

m-certs/, 2025. (Accessed on September 25th, 2025).

[46] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, 2001.

[47] Wiley J Larson and James Richard Wertz. Space Mission Analysis and Design. Technical report, Torrance, CA (United States); Microcosm, Inc., 1992.

[48] Demi Lei and Ahmed Saeed. Do we need a million satellites in orbit? constellation-as-a-service with modular satellites: Challenges and opportunities. In *Proceedings of the 2nd International Workshop on LEO Networking and Communication*, pages 61–66, 2024.

[49] Yuanjie Li, Hewu Li, Wei Liu, Lixin Liu, Yimei Chen, Jianping Wu, Qian Wu, Jun Liu, and Zeqi Lai. A case for stateless mobile core network functions in space. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 298–313, 2022.

[50] Michael Nicolls. X Post. https://x.com/michaelnicollsx/status/1948509258024452488, 2025. (Accessed on September 25th, 2025).

[51] David L Mills. Network time protocol (NTP). Technical report, 1985.

[52] Mitchell Clark, Richard Lawler. T-Mobile and SpaceX Starlink say your 5G phone will connect to satellites next year. https://www.theverge.com/2022/8/25/23320722/spacex-starlink-t-mobile-satellite-internet-mobile-messaging, 2022. (Accessed on September 25th, 2025).

[53] Nitinder Mohan, Andrew E Ferguson, Hendrik Cech, Rohan Bose, Prakita Rayyan Renatin, Mahesh K Marina, and Jörg Ott. A Multifaceted Look at Starlink Performance. In *Proceedings of the ACM on Web Conference 2024*, pages 2723–2734, 2024.

[54] Monica Alleven. Verizon taps Amazon's Kuiper for backhaul, rural connectivity. https://www.fiercewireless.com/wireless/verizon-taps-amazon-s-kuiper-for-backhaul-rural-connectivity, 2021. (Accessed on September 25th, 2025).

[55] Iulian Moraru, David G Andersen, and Michael Kaminsky. There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 358–372, 2013.

[56] Sebastian Moss. Thales Alenia Space wins EU feasibility study for 'Ascend' space data centers. https://www.datacenterdynamics.com/en/news/thales-alenia-space-wins-eu-feasibility-study-for-ascend-space-data-centers/, 2022. (Accessed on September 25th, 2025).

[57] Seoyul Oh and Deepak Vasisht. A Call for Decentralized Satellite Networks. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 25–33, 2024.

[58] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, 2014.

[59] Tobias Pfandzelter and David Bermbach. Celestial: Virtual software System Testbeds for the LEO Edge. In *Proceedings of the 23rd conference on 23rd ACM/IFIP International Middleware Conference*, pages 69–81, 2022.

[60] Tobias Pfandzelter and David Bermbach. Komet: A Serverless Platform for Low-Earth Orbit Edge Services. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pages 866–882, 2024.

[61] Planet Labs. Planet Imagery Product Specifications. https://assets.planet.com/docs/Planet_Combined_Imagery_Product_Specs_letter_screen.pdf, 2023. (Accessed on September 25th, 2025).

[62] Reuters. Spacex probes for cause of starlink's global satellite network outage. https://www.reuters.com/technology/spacex-probes-cause-starlinks-global-satellite-network-outage-2025-07-25, 2025. (Accessed on September 25th, 2025).

[63] Klaus Schilling. Mission analysis for low-earthobservation missions with spacecraft formations. *NATO RTO Lecture Series–Small Satellite Formations for Distributed Surveillance: System Design and Optimal Control Considerations. RTO-EN-SCI-231. NATO RTO Lecture Series, Würzburg, Germany*, 2011.

[64] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. Omega: flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 351–364, 2013.

[65] Starlink. APPLICATION FOR APPROVAL FOR ORBITAL DEPLOYMENT AND OPERATING AUTHORITY FOR THE SPACEX NGSO SATELLITE SYSTEM. https://cdn.arstechnica.net/wp-content/uploads/2016/11/spacex-Legal-Narrative.pdf, 2016. Accessed on September 25th, 2025.

[66] Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, et al. Twine: A Unified Programming Model for Shared Memory and Message Passing. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 787–803, 2020.

[67] Hammas Bin Tanveer, Mike Puchol, Rachee Singh, Antonio Bianchi, and Rishab Nithyanand. Making sense of constellations: Methodologies for understanding starlink's scheduling algorithms. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, pages 37–43, 2023.

[68] Bill Tao, Om Chabra, Ishani Janveja, Indranil Gupta, and Deepak Vasisht. Known knowns and unknowns: Near-realtime earth observation via query bifurcation in serval. In *25th USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 809–824, 2024.

[69] Muhammad Tayyab, Xavier Gelabert, and Riku Jäntti. A survey on handover management: From LTE to NR. *IEEE Access*, 7:118907–118930, 2019.

[70] Cisco ThousandEyes. Starlink Global Outage Analysis. https://www.thousandeyes.com/blog/starlink-outage-analysis-july-24-2025, 2025. (Accessed on September 25th, 2025).

[71] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, pages 1–16, 2013.

[72] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the tenth european conference on computer systems*, pages 1–17, 2015.

[73] Vivek Wadhwa and Alex Salkever. How Elon Musk's Starlink Got Battle-Tested in Ukraine. https://foreignpolicy.com/2022/05/04/starlink-ukraine-elon-musk-satellite-internet-broadband-drones, 2022. (Accessed on September 25th, 2025).

[74] Andrew Wooden. Lynk Launches 'First Commercial-Ready Cell-Tower-In-Space'. https://www.telecoms.com/satellite/lynk-launches-first-commercial-ready-cell-tower-in-space-, 2022. (Accessed on September 25th, 2025).

[75] Wu Xiang, Yakun Li, Yuquan Ren, Fan Jiang, Chaohui Xin, Varun Gupta, Chao Xiang, Xinyi Song, Meng Liu, Bing Li, et al. Gödel: Unified large-scale resource management and scheduling at ByteDance. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, pages 308–323, 2023.

[76] Irene Zhang, Naveen Kr. Sharma, Adriana Szekeres, Arvind Krishnamurthy, and Dan R. K. Ports. Building consistent transactions with inconsistent replication. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 263–278, New York, NY, USA, 2015. Association for Computing Machinery.

[77] Irene Zhang, Naveen Kr Sharma, Adriana Szekeres, Arvind Krishnamurthy, and Dan RK Ports. Building consistent transactions with inconsistent replication. *ACM Transactions on Computer Systems (TOCS)*, 35(4):1–37, 2018.

[78] William X Zheng, Aryan Taneja, Maleeha Masood, Anirudh Sabnis, Ramesh K Sitaraman, and Deepak Vasisht. StarCDN: Moving Content Delivery Networks to Space. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 948–962, 2025.