Temporal Coherence in Long-Horizon, Intermittently Connected Sensor Systems

Abstract

Long-horizon sensing systems operating under intermittent connectivity exhibit a characteristic failure mode: time becomes non-queryable (in the sense that basic provenance and causal queries cannot be answered reliably at the time operational decisions were made). Clock drift, buffering, and delayed uplink cause event time, record time, and ingest time to collapse into a single timestamp, destroying causal reasoning, undermining auditability, and reducing fleet-level value by forcing operators to revert to conservative or manual decision-making. This note describes the failure mode, explains why it becomes most visible in unmanned and remote systems, and proposes a minimal representation of time that preserves coherence without requiring continuous synchronisation.

1 Introduction

Modern sensing deployments increasingly rely on unmanned, remote, and power-constrained platforms operating for extended periods without continuous connectivity. Examples include seabed observatories, autonomous surface and subsurface vehicles, polar monitoring stations, and remote environmental sensor networks. In these settings, sensors must operate autonomously for long durations, often with limited opportunities for communication or maintenance.

While sensor fidelity, storage capacity, and communications bandwidth have improved significantly, system-level temporal coherence has not kept pace. In many long-horizon deployments, the first thing to degrade is not sensing or transport, but temporal coherence. Data continues to arrive and deployments appear operational, yet the ability to reason about sequence, simultaneity, and causality quietly degrades.

This degradation is rarely treated as a first-order systems problem. Instead, it is often addressed implicitly, through ad hoc timestamp correction or manual interpretation. The result is data that appears precise but is increasingly difficult to trust as deployments scale in duration, autonomy, and complexity.

This failure mode is architectural rather than algorithmic, and therefore persists across vendor changes, firmware updates, and platform refreshes. As a result, programmes may appear to modernise while retaining the same latent decision risk. It does not arise from inadequate sensing or transport, but from the

implicit assumption that time can be collapsed into a single authoritative scalar during ingestion. Specifically, systems tend to collapse event time, record time, and ingest time into a single scalar timestamp without preserving uncertainty or provenance. The consequences surface downstream, at the level of analysis, assurance, and decision-making. As such, temporal coherence must be treated as a system property that spans sensing, buffering, transport, and ingest, rather than a local implementation detail.

2 From Seabed to Screen

Consider a distributed maritime sensing deployment comprising seabed nodes, autonomous surface or sub-surface vehicles, and intermittent satellite uplink to shore-based analysis. Individual platforms operate autonomously for extended periods, resynchronising clocks only opportunistically when surfaced, serviced, or recovered. In many cases, such recovery occurs long after the operational decisions that depended on the data. Observations may traverse multiple buffering stages before ingestion, with delays spanning minutes to weeks. In such deployments, temporal coherence can degrade even when sensing and transport remain nominally operational.

A typical long-horizon sensing deployment spans multiple temporal domains:

- Sensor-local clocks operating under power and thermal constraints
- Intermediate gateways such as seafloor nodes, buoys, or vehicles
- Transport layers including acoustic links, satellite uplinks, fibre connections, or physical retrieval
- Ingest systems that assign arrival-time timestamps
- Analytical systems that assume a single global timeline

Each layer maintains its own notion of time, often with different accuracy, stability, and update frequency. In many systems, these distinct temporal domains are collapsed implicitly during ingestion, producing a single timestamp that is treated as authoritative. This collapse typically occurs by default rather than by deliberate design, as no single team owns temporal semantics end-to-end.

This collapse is convenient but misleading. The resulting timestamp is neither purely an event time nor a record time, but a hybrid whose meaning depends on undocumented assumptions about buffering, drift, and correction. Outside its immediate context, the timestamp no longer supports reliable reasoning.

3 Failure Mode

3.1 Clock Drift at the Edge

Edge sensors frequently rely on inexpensive oscillators that drift seconds to minutes over weeks or months. Continuous synchronisation (e.g., GNSS) is often

unavailable due to depth, power constraints, or environmental conditions. As a result, sensors may only resynchronise when surfaced, serviced, or physically recovered.

When drift is corrected, it is often applied as a step adjustment, retroactively shifting all prior timestamps long after operational decisions have already been taken. The uncertainty introduced by drift is rarely preserved or propagated, and downstream systems are not informed that historical timestamps have been modified.

3.2 Asymmetric Buffering and Burst Upload

Intermittent connectivity forces sensors and gateways to buffer data locally and transmit it in bursts. Data arrival time at the ingest system may lag event time by minutes, hours, or weeks. Buffering can also introduce reordering, partial uploads, or duplication.

Despite this, ingest systems often conflate arrival time with event time, either by overwriting timestamps or by implicitly privileging the time of receipt. This erases information about when observations actually occurred.

3.3 Arrival Time Masquerading as Event Time

Dashboards and alerting systems often assume that timestamps represent the moment an event occurred. When arrival time is substituted for event time, correlations appear that are artefacts of transport rather than reality. Spurious simultaneity and false causality become difficult to detect.

The system continues to function, but its outputs no longer correspond cleanly to the physical world it is meant to observe.

Operators often compensate informally by mentally re-timestamping data or discounting automation, masking the issue until scale or autonomy removes the human from the loop.

3.4 Loss of Auditability

Once timestamps are normalised or overwritten without provenance, it becomes impossible to reconstruct what the system knew at a given moment. Questions such as "What information was available when this decision was made?" or "Which observations could have influenced this alert?" cannot be answered reliably.

Each subsequent normalisation or overwrite further weakens the ability to reconstruct ordering, provenance, and system state at the time of decision. This loss of auditability is cumulative and often invisible until trust has already eroded.

Once temporal provenance has been overwritten or discarded, it cannot be reconstructed after the fact; no amount of downstream sophistication can recover it.

From an assurance perspective, this undermines the ability to defend decisions after the fact. Without preserved temporal provenance, it is not possible to demonstrate what information was available at the time a decision was taken, nor to bound which observations could have influenced it. This complicates safety cases, incident investigation, and regulatory or legal scrutiny. The resulting loss of trust is systemic rather than sensor-specific.

4 Reframing Time

Rather than treating time as a scalar, we treat it as a bounded estimate with provenance. Event time is separated explicitly from record time and ingest time, and uncertainty is represented as a first-class property of each observation.

This reframing aligns with how human operators already reason about delayed or uncertain observations. Operators naturally discount older or less certain data and reason in terms of windows rather than points. This observation is used only as an analogy for informal practice; the goal is to formalise temporal uncertainty so systems can behave consistently under intermittency. The contribution here is to render this reasoning explicit, machine-readable, and enforceable in software.

5 A Minimal Model

5.1 Event Representation

Each observation carries:

- An estimated event time
- A temporal uncertainty bound
- Clock provenance (e.g., sensor, gateway, external reference, inferred)
- Time since last synchronisation

Formally, event time is represented as an interval rather than a point:

$$t_{\text{event}} \in [t - \epsilon,\, t + \epsilon]$$

where ϵ is a conservative upper bound on accumulated temporal uncertainty due to drift and transport, which grows with time since the last trusted anchor.

Responsibility for maintaining conservative and honest uncertainty bounds must be explicit, as overly optimistic bounds reintroduce false precision into downstream reasoning.

5.2 Temporal Ordering

Event ordering is defined only when justified by uncertainty bounds. Given two events A and B with intervals $[A^-, A^+]$ and $[B^-, B^+]$, we say that A precedes B if and only if:

$$A^+ < B^-$$

If the intervals overlap, no ordering is asserted. In many deployed pipelines, interval information is discarded at ingestion, permanently eliminating safe partial ordering even when the underlying system uncertainty would have permitted it. This preserves causal ambiguity rather than forcing a false total order.

5.3 Invariants

The system enforces a small set of invariants:

- Event time is never overwritten
- Uncertainty never shrinks without a new temporal anchor
- Corrections are additive and auditable, not destructive

These invariants constrain behaviour without prescribing a specific synchronisation mechanism or transport architecture.

These invariants are compatible with incremental adoption. Systems may coexist with legacy timestamped data, and the model does not require changes to sensing hardware or transport layers. Temporal coherence can therefore be introduced selectively, at points of ingestion or analysis, without wholesale architectural replacement.

6 Relation to Prior Approaches

Related problems have been addressed in adjacent domains, including clock synchronisation protocols, event-time processing in data streams, and causal ordering in distributed systems. These approaches typically assume frequent communication, stable clocks, or the ability to overwrite timestamps as improved information becomes available. Examples include event-time semantics in stream processing, clock synchronisation in time-sensitive networking, and partial orders in distributed systems; each addresses a facet of the problem but typically assumes stable connectivity or revisable history.

Long-horizon, intermittently connected sensing systems violate these assumptions. Synchronisation opportunities are sparse, drift accumulates continuously, and historical data cannot be revised without destroying auditability. As a result, existing approaches do not preserve temporal coherence under the conditions considered here.

In operational settings, decisions must be defensible based on the information available at the time, not on retrospectively revised history.

This work reframes time not as a synchronisation problem, but as an architectural property that must persist across sensing, buffering, and delayed ingestion.

7 Operational and Assurance Implications

7.1 Queryable Time

Once time is represented explicitly, systems can answer questions that are otherwise ill-defined:

- What was known at a given time?
- Which correlations are temporally plausible?
- Which observations could have influenced a decision?
- Whether a different action would have been reasonable given the information available at the time?

Time becomes queryable rather than assumed.

7.2 Fleet-Level Value

When temporal uncertainty is bounded and explicit, data from multiple platforms can be correlated without false precision. Operationally, this manifests in tighter operating envelopes, fewer manual overrides, and increased confidence in automated recommendations. The analytical value of a fleet increases materially once coherence is preserved, because cross-platform reasoning no longer relies on brittle assumptions about simultaneity.

7.3 Why Autonomy Makes the Problem Observable

When humans are present on platforms, continuity often lives in people. Operators implicitly reconcile delays, drift, and context. As platforms become unmanned, that continuity has nowhere to live unless it is encoded in the system itself.

Autonomy does not create the temporal problem; it makes it observable.

For long-horizon sensing programmes, this reframing enables fleets to scale in duration, autonomy, and heterogeneity without a corresponding loss of trust or auditability.

These effects are most pronounced in high-latitude and polar deployments, where thermal drift, sparse synchronisation opportunities, regulatory scrutiny, and delayed recovery compound temporal incoherence.

8 Conclusion

Temporal coherence is a prerequisite for trustworthy long-horizon sensing. Systems that model time explicitly retain analytical and operational value under intermittency; systems that do not instead accumulate unbounded ambiguity in ordering, provenance, and auditability. Temporal coherence is not a property of clocks or networks, but of system architecture. By preserving temporal provenance, architectures retain the ability to explain and defend decisions long after deployment, even under prolonged intermittency.

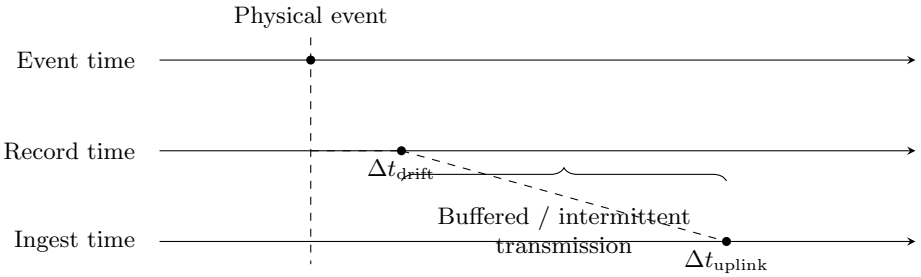

Figure 1: Temporal domains in a typical long-horizon sensing stack. A single physical event maps to divergent timestamps under clock drift and delayed uplink. Event time, record time, and ingest time must be represented explicitly to preserve causal coherence.

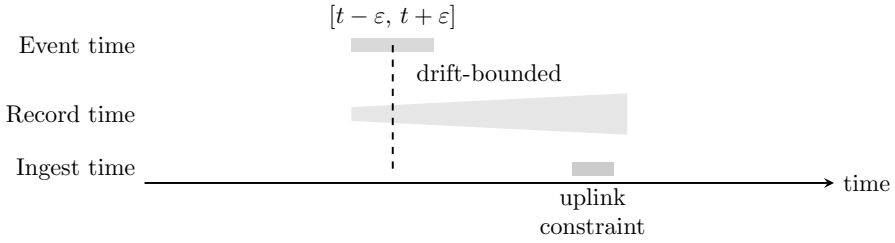

Figure 2: Temporal coherence with propagated uncertainty. Event time is represented as an interval whose uncertainty evolves across sensing, buffering, and uplink rather than collapsing into a single timestamp.

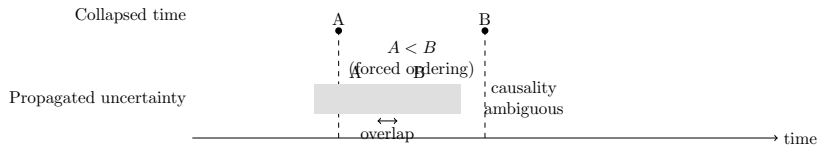

Figure 3: Temporal causality under collapsed versus propagated time. When timestamps are collapsed to points (top), events appear totally ordered even when that ordering is not justified; when uncertainty is propagated (bottom), overlapping event intervals make causal ambiguity explicit and queryable.

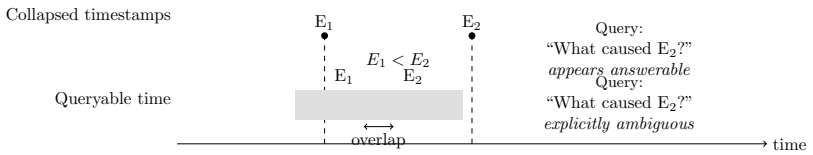

Figure 4: Timestamp collapse versus queryable time. *(conceptual illustration)* When timestamps are collapsed to points (top), systems enforce a total ordering that makes causal queries appear answerable even when that ordering is unjustified; when temporal uncertainty is propagated (bottom), overlapping event intervals expose ambiguity, making causal queries explicit rather than silently wrong.

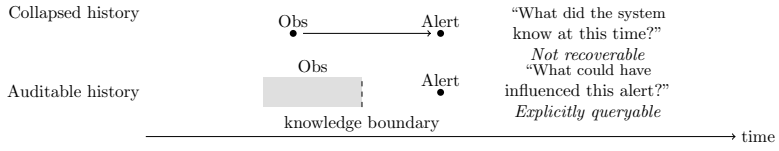

Figure 5: Auditability under collapsed versus propagated time. *(conceptual illustration)* When timestamps are rewritten or normalised (top), historical system state cannot be reconstructed; when temporal uncertainty is preserved (bottom), the system retains an explicit boundary between what was known and what was not, enabling post-hoc audit, explanation, and governance.