Nick Clark · Published April 25, 2026 · PDF

How to Keep Clocks Synchronized Across a Network Without GPS

If your fleet, sensor field, or edge mesh loses GPS or its connection to a stratum-1 time server, its clocks drift apart and every timestamped record becomes suspect. This guide describes an architecture for a shared time reference the devices compute cooperatively among themselves, with each timestamp carrying a governance credential you can audit. The approach is disclosed in U.S. Provisional Application No. 64/049,409 as the Mesh-Derived Time inventive step; it is a design you build, not a library you install.

What You Are Building

You are building a way for a set of networked devices to agree on what time it is without any of them being able to read GPS and without all of them trusting one central clock. Concretely: every device keeps its own local clock, the devices exchange timing information with their neighbors, and out of those exchanges each device computes an estimate of its offset from a shared mesh time reference. The reference is produced by the group cooperatively rather than handed down from an external source.

The people who need this are the ones whose timing breaks when a single source goes away: vehicle fleets and autonomous units operating through GPS jamming or urban canyons, industrial and sensor deployments where a link back to a network time server is unreliable, and any system that has to defend its timestamps later, in an audit or a

dispute, and cannot simply say "GPS told us so." The disclosed approach targets exactly that gap. It aims at "a cooperatively estimated temporal reference frame derived from device-to-device synchronization without dependence on GPS time or a centralized time authority."

The design adds one property beyond raw synchronization: every timestamp it emits carries the attesting agent's authority credential and its estimated uncertainty, and every step that produced it is recorded in a lineage field so the derivation can be reconstructed later.

Why the Obvious Approaches Fall Short

The standard tools are real and good at what they do. The point here is where each one structurally depends on something you may not have.

GPS and other satellite time. Satellite time services broadcast from centrally operated constellations. Acquiring the signal is a precondition for getting time, so when the signal is denied or spoofed, timing is denied with it. If your threat model includes GPS loss, GPS cannot also be your fallback.

Network Time Protocol (NTP). NTP is client-server and hierarchical: clients ultimately trace their time up to stratum-1 servers. It works extremely well when that path exists. It assumes a reachable, trusted upstream server, which is the assumption a disconnected or partitioned mesh cannot rely on.

Precision Time Protocol (PTP). PTP achieves tight synchronization on a local network, but it is organized master-slave around a grandmaster clock elected for the domain. It presumes a designated master exists and is reachable.

Blockchain timestamps. Distributed-ledger timestamps avoid a single authority, but they are pinned to block-commit events, which yields coarse, minute-scale timing rather than a continuous reference.

Trusted timestamp authorities (TSAs). A TSA issues signed timestamps you can present later, which is genuinely useful for audit. By design it centralizes issuance at one authority, so it is a single point of trust and of failure.

None of these is wrong; each simply leans on a component the others lack. The recurring structural gap is that timekeeping and time-attestation both concentrate on one source or one master. The architecture below removes that concentration: no required satellite, no master clock, no single issuing authority, and an audit trail on every timestamp.

The Architecture

The disclosed primitive treats mesh-derived time as a first-class part of the system rather than a bolt-on. Per the filing, it comprises the following cooperating mechanisms. Treat this section as the specification you implement against.

Clock-maintaining agents. Every participant maintains its own local clock and a characterization of that clock's drift properties, described by governance policy. There is no master clock; each agent is a peer that both contributes timing observations and consumes the shared estimate.

Governance-credentialed synchronization exchanges. Agents produce time-synchronization observations between each other through one of several synchronization modalities. Every such observation is governance-credentialed, meaning it carries the emitting authority's credential rather than being an anonymous packet. This is what later lets a timestamp be traced back to who vouched for it.

Temporal anchor admission. The mesh can admit governance-credentialed temporal anchor contributions through a dedicated admission interface. An anchor is an external time input the policy has decided to trust (an atomic reference, a satellite

fix while one happens to be available, a network time source). Anchors are optional inputs the mesh fuses in, not a dependency it requires.

Cooperative time-estimation engine. This is the core solver. It determines each agent's time-offset by combining the synchronization observations with whatever anchor contributions exist. The output is a per-agent offset from the shared frame; applying it to the local clock yields mesh time.

Transitive time-propagation. When an agent has no direct line to an anchor, a propagation extender derives its offset through its neighbors' references, so agents several hops from any anchor still land in the same frame.

Anchor-less temporal bootstrap. With zero anchors available, a bootstrap mechanism produces a relative-only temporal frame. The mesh still agrees internally on elapsed time and ordering; it simply is not pinned to an external absolute scale until an anchor arrives. This is the property that keeps the system working through total GPS denial.

Drift compensation and clock-model learning. A drift-compensation mechanism continuously corrects each local clock's drift using fresh synchronization exchanges, and a clock-model learning mechanism refines each agent's drift characterization over time through governance-credentialed training. Better per-agent drift models mean the estimate degrades more gracefully between exchanges.

Uncertainty propagation. A time-uncertainty propagator pushes synchronization uncertainty through the temporal graph and produces a per-agent time-uncertainty estimate. Time here is never a bare number; it is a value with an uncertainty attached, and consumers can gate their behavior on that uncertainty.

Adversarial-time rejection. A rejection mechanism screens out spoofed, injected, or otherwise inadmissible synchronization observations before they reach the estimator. Because observations are credentialed and evaluated for admissibility, a rogue node

cannot silently drag the mesh's clock.

Evidential fusion with external time. When external time is present, an evidential-fusion mechanism combines mesh-derived time with those external sources through a composite admissibility evaluator, rather than blindly overwriting mesh time with whatever a single source claims.

Time-frame federation. A federation mechanism aligns independently maintained temporal frames so two meshes that grew up separately can be reconciled with the alignment preserved in the governance chain.

Governance-credentialed timestamp attestation. On top of the estimate sits an attestation interface. When a credentialed requester asks for a timestamp, an admissibility evaluator applies policy rules, and a composer emits a timestamp observation carrying the attesting agent's authority credential, the mesh-derived time value, the estimated uncertainty, the attesting agent's identity, and a cryptographic signature. The disclosure describes attestation patterns including single-attester, multi-attester consensus signed by a policy-defined quorum, and content-bound attestation that cryptographically binds a timestamp to specific content by content-addressing.

Time-lineage recorder. Every synchronization exchange, anchor admission, estimation event, rejection, federation event, and attestation is recorded in a lineage field. The result: a timestamp's derivation chain, the synchronization chain that produced the attesting agent's time, and the authority-credential chain are all reconstructible after the fact for regulatory, legal, or forensic audit.

The primitive is also disclosed as composing with a companion mesh-derived coordinate primitive into a unified spacetime reference, where each agent carries both position and time bearings in one governance-preserving structure. If you are already building GPS-independent positioning, time is the same graph with a time axis added.

How to Approach the Build

You are implementing this yourself. A reasonable order:

1. **Nail down the credential and lineage substrate first.** Every observation in this design is credentialed and every step is logged. Decide how an agent proves its authority (the disclosure assumes a governance credential bound to the device) and where the lineage records live. Retrofitting attestation onto an anonymous sync protocol later is painful; make credentials and lineage load-bearing from day one.
2. **Give each agent a local clock and a drift model.** Start with a simple offset-plus-rate model per agent. Leave a hook for the clock-model learning step to refine drift characterization later.
3. **Implement one synchronization modality end to end.** The disclosure allows several; pick one exchange between neighbors, and make every message carry the emitting authority's credential. An illustrative observation shape, purely to fix ideas and not a wire format from the filing:

```
SyncObservation {
  from_agent, authority_credential,
  local_send_time, local_recv_time,   // as measured by each clock
  signature
}
```

4. **Build the estimation engine.** Combine incoming synchronization observations, plus any admitted anchors, into a per-agent offset from the shared frame. Add the transitive extender so agents without a direct anchor solve through neighbors, and add the anchor-less bootstrap path so the mesh converges to a relative frame with zero anchors present.

5. **Propagate uncertainty, not just offsets.** Attach an uncertainty to every offset and propagate it through the graph. Downstream consumers should be able to ask "how sure are we of this time" and refuse to act when the answer is too weak.
6. **Add admission and rejection.** Put an admissibility check in front of the estimator: verify the credential, apply policy, and drop spoofed or inadmissible observations before they influence the estimate.
7. **Add the anchor and external-fusion paths.** Let policy-trusted anchors feed in through the admission interface, and fuse external time through a composite admissibility evaluator so an external source informs rather than dictates.
8. **Build the attestation interface last.** Once the estimate is trustworthy, expose the timestamp attestation: a credentialed requester gets back a timestamp observation carrying credential, time value, uncertainty, attester identity, and signature. Add multi-attester quorum signing where you need higher assurance, and content-binding where a timestamp must be tied to a specific artifact.
9. **Exercise the failure paths deliberately.** Kill all anchors and confirm the mesh holds a coherent relative frame. Inject a lying node and confirm rejection contains it. Partition the mesh and rejoin it, then confirm federation reconciles the two frames with the alignment in the lineage.

What This Does Not Give You

This is an architecture disclosed in a patent filing, not a shipping product, an SDK, or a downloadable package. There is nothing to `npm install`. The interface sketch above is illustrative, included to make the shape concrete; it is not a wire format or reference implementation from the disclosure.

It has not been benchmarked here, and the filing does not state achievable accuracy, convergence times, node counts, or drift bounds. Those depend entirely on your clocks, your synchronization modality, your link quality, and your policy, and you must

measure them on your own build. Do not quote a precision figure on the strength of this guide, because the disclosure does not provide one.

Some boundaries are structural. With no anchors ever admitted, the mesh gives you a relative frame: internally consistent ordering and elapsed time, but not absolute wall-clock time until an anchor arrives. If your application genuinely needs traceable absolute time to a legal standard at all moments, you still need an anchor you trust; the value here is that you no longer need it continuously or from a single source. The security properties follow from your credential system and admissibility policy; weak credentials or permissive policy weaken adversarial-time rejection accordingly. And where relativistic effects matter, the disclosure notes a consistency evaluator but you own that analysis.

Disclosure Scope

The architecture described here is disclosed as the Mesh-Derived Time inventive step in U.S. Provisional Application No. 64/049,409. This guide is educational: it explains how to approach building such a system and traces its mechanisms to that filing. It is not a warranty, a specification of a released product, or an offer of software, and no working library, SDK, or package is provided or implied. You implement the architecture yourself, and any accuracy, security, or reliability characteristics are yours to establish through your own implementation and testing.

Mesh-Derived Time (/mesh-time) All 40 steps → (/inventive-steps)

Master-less consensus time. Joint spatial-temporal optimization. Multi-attester timestamps.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- Mesh-Derived Time: Master-Less Consensus and Joint Spacetime Optimization (/articles/mesh-derived-time-master-less-consensus-and-joint-spacetime-optimization)

SECONDARY TECHNICAL

- Master-Less Consensus Time (/articles/mesh-time/master-less-consensus)
- Per-Agent Learned Drift Models (/articles/mesh-time/per-agent-drift-models)
- Ranging-Piggyback Synchronization (/articles/mesh-time/ranging-piggyback-sync)
- Joint Spatial-Temporal Graph (/articles/mesh-time/joint-spacetime-graph)
- Multi-Attester Consensus Timestamping (/articles/mesh-time/multi-attester-consensus-timestamp)
- Time-Frame Federation Across Mesh Regions (/articles/mesh-time/time-frame-federation)
- Integrated Relativistic Correction (/articles/mesh-time/relativistic-correction)
- Anti-Spoofed Time Observations (/articles/mesh-time/anti-spoofed-time)
- Anchorless Time Bootstrap (/articles/mesh-time/anchorless-bootstrap)
- Audit-Grade Time Attestation (/articles/mesh-time/audit-grade-attestation)

APPLICATIONS · GENERAL

- Coordinated Time for Autonomous Fleets in GNSS-Denied Operations (/articles/mesh-time/autonomous-fleet-coordinated-time)
- Legally Defensible Settlement Timestamps Without Single-Source Clock Authority (/articles/mesh-time/financial-settlement-attested-time)
- GNSS Time Denial: Holding Critical Infrastructure Timing When GPS Is Jammed or Spoofed (/articles/mesh-time/gnss-time-denied-critical-infrastructure)
- 5G/6G Network Timing Without Master-Broadcast Dependency (/articles/mesh-time/5g-6g-network-timing)
- How to Timestamp a Blockchain Without Consensus Overhead (/articles/mesh-time/blockchain-time-without-consensus)
- Tamper-Evident Trading Timestamps Without a Single GPS Grandmaster (/articles/mesh-time/high-frequency-trading-time)
- Post-Quantum Migration for Timestamping Authorities (/articles/mesh-time/post-quantum-cryptographic-time)
- GPS-Independent Spacecraft Time Coordination for Satellite Constellations (/articles/mesh-time/spacecraft-coordinated-time)

APPLICATIONS · SPECIFIC

- IEEE 1588 PTP vs Master-Less Governed Mesh Time (/articles/mesh-time/ieee-1588-ptp)
- Microchip SyncE Lacks Master-Less Consensus Time (/articles/mesh-time/microchip-syncE)
- Microchip SyncServer Alternative: Governed Master-Less Time Consensus (/articles/mesh-time/microsemi-tsync)
- Meinberg NTP Alternative: Governed Master-Less Time Consensus (/articles/mesh-time/meinberg-ntp)
- Oscilloquartz Alternative: Governed Master-Less Time Consensus (/articles/mesh-time/oscilloquartz-timing)
- Spectracom Orolia Alternative: Governed Master-Less Time Consensus (/articles/mesh-time/spectracom-orolia)
- Trimble Thunderbolt Alternative: Master-Less Mesh Time Consensus (/articles/mesh-time/trimble-thunderbolt)
- White Rabbit (CERN) vs Mesh Time: Master-Less Timing Consensus (/articles/mesh-time/white-rabbit-cern)
- DigiCert timestamping (RFC 3161) alternative: mesh-derived, governance-credentialed time attestation for legal and forensic audit (/articles/mesh-time/digicert-timestamping)

Mesh-Derived Time overview → (/mesh-time)