White Paper

Clock Manager: Extending Synchronization Visibility into Industrial Automation

Synchronization Health Monitoring for Modern Systems

Authors
Yoong Siang Song
Jun Ann (Peter) Lai
Wei Sheng Goh
David Zage
Christopher S. Hall
Shi Jie (Donavan) Liow
Weifeng Voon
Intel Corporation

Executive Summary

Downtime, safety risks, and quality escapes in automation and energy systems often stem from invisible synchronization failures. When clocks drift or a grandmaster changes without detection, manufacturing workflows break down, power grids destabilize, and critical operations falter, costing millions and eroding trust.

Clock Manager is a novel, permissively licensed software solution designed to monitor synchronization across critical environments. It provides real-time telemetry and proactive alerts for synchronization status changes, enabling rapid intervention and minimizing downtime. The solution consists of two core components: a client-runtime library that provides APIs for applications to subscribe to synchronization updates and a backend service that interfaces with synchronization daemons such as the Linux PTP (ptp4l) or Chrony (chronyd) to deliver real-time status.

This paper provides an overview of the Clock Manager's value, architecture, technical advantages, and roadmap for scaling synchronization management across diverse environments.

Introduction

Time synchronization has become a strategic enabler of resilience, safety, and profitability for modern industrial automation and critical infrastructure. Technologies such as Time-Sensitive Networking (TSN) and IEEE 1588 Precision Time Protocol (PTP) enable deterministic, low-latency communication across controllers, motor drives, sensors, and gateways.

However, achieving synchronization is only the starting point. Sustaining continuous visibility into synchronization health and proactively mitigating degradation is equally critical for operational excellence. Even robust system deployments remain vulnerable without real-time monitoring and intelligent fault detection. In mission-critical applications where microseconds matter, blind trust in synchronization status is not an option.

Problem Statements

Industrial automation depends on synchronized timing to keep workflows predictable, maximize throughput, and reduce the risk of costly malfunctions that could lead to equipment damage, financial loss, or personal injury. Similarly, in the energy sector, precise timekeeping is essential for grid stability and optimized power generation. For example, wind farms depend on accurate synchronization to coordinate turbine operations and maximize output.

Real-world networks are inherently dynamic. Network load fluctuations, grandmaster changes/failures, hardware aging, interference, topology reconfiguration, and

firmware updates can introduce offset, drift, jitter, and path-delay variation. If left undetected, these issues can manifest as:

- Motion discoordination
- Packet loss
- Missed control windows
- Protective system trips

The operational consequences of not understanding synchronization health are high: failures don't just impact individual devices, they cascade through entire systems, creating ripple effects that can shut down production lines, compromise product quality, or trigger safety systems.

Continuous synchronization monitoring addresses four critical business needs:

- Operational continuity – Detect drift early to prevent downtime.
- Safety & compliance – Trigger alarms or safe-state transitions.
- Quality & performance – Maintain throughput and precision.
- Cost control – Reduce fault triage time and avoid cascading failures.

Beyond operations, direct integration with synchronization daemons introduces licensing risk. Many widely deployed solutions, including Linux PTP Project, operate under copyleft licenses that can mandate disclosure of proprietary industrial control code, an unacceptable risk for competitive manufacturing applications. Organizations need synchronization visibility without compromising their intellectual property or violating compliance requirements.

Solution Overview

Clock Manager bridges this critical gap by providing real-time synchronization visibility without licensing risk and with minimal integration overhead.

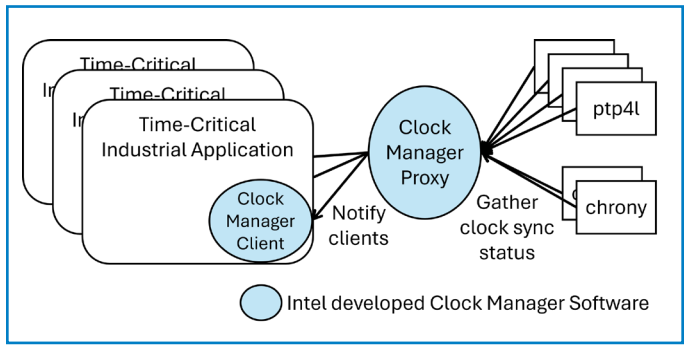

Figure 1. Clock Manager Functional Overview

As shown in Figure 1, Clock Manager acts as a secure, lightweight intermediary between customer applications and synchronization daemons. It communicates with restrictively licensed daemons on one side and applications on the other—while Clock Manager itself is released under a permissive BSD 3 Clause license. This decoupled design eliminates licensing concerns and simplifies adoption, without sacrificing functionality.

Key Benefits:

- **Reduced licensing risk** – Protect proprietary code, no exposure to restrictive licenses.
- **Real-time monitoring** – Timely notification of synchronization status changes.
- **Simplified integration** – One interface for all time sync needs (Clock Manager proxy).
- **Multi-domain support** – Seamlessly handles complex network topologies.
- **Cross-platform compatibility** – Works across major Linux distributions.
- **Production-ready** – Includes comprehensive documentation and reference examples.
- **Language flexibility** – Supports nine programming languages for easy adoption.

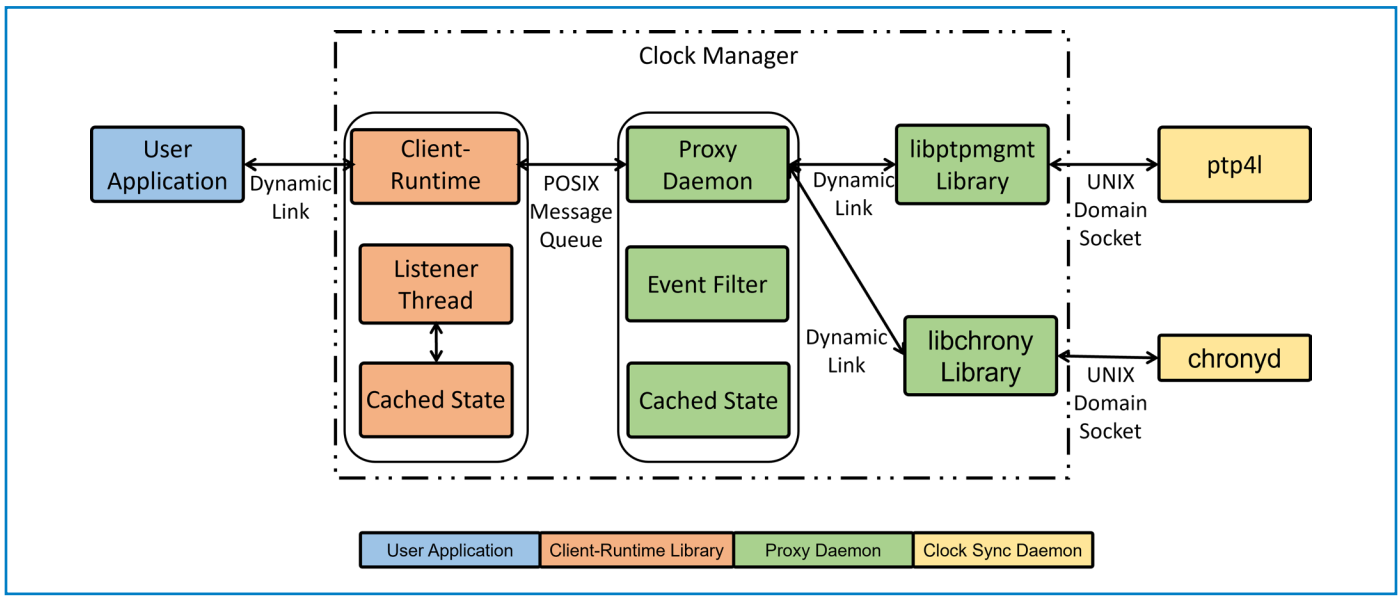

Figure 2. Clock Manager System Architecture

Architecture and Functionality

As shown in Figure 2, the Clock Manager is built using a two-tier architecture designed for flexibility and responsiveness:

- Proxy Daemon – Interfaces with multiple time sync daemons (ptp4l, Chrony) to aggregate and monitor synchronization data.
- Client Runtime Library – Exposes a clean API for applications to subscribe to events and receive updates without direct interaction with synchronization daemons.

The proxy daemon subscribes to ptp4l via libptpmgmt and listens for IEEE 1588 management traffic. For Chrony, it uses libchrony to query synchronization status. When changes occur, the proxy filters relevant data such as:

- Clock offset
- Grandmaster UUID
- AS-capable status
- Synchronization interval

These updates are published to applications via message queues.

Applications can subscribe to specific events, ensuring they receive only the notifications they need. This design supports fine-grained monitoring and scalable integration across diverse industrial environments.

Technical Advantages

- **Real-Time Notifications** – Synchronization state changes propagate quickly, ensuring timely notification.
- **Stability Under Stress** – Efficient design delivers reliable performance during high volume conditions.
- **Scalable Multi-Domain Awareness** – Supports parallel monitoring of independent PTP domains with correct isolation and event handling.

Future Roadmap

Currently, the Clock Manager supports a single PTP-distributed time source that is sufficient for legacy applications, but emerging industrial automation applications require multiple time sources. Manufacturing facilities are partitioned into work cells that require a private source of time to synchronize communication and computation within the work cell. Some devices may not be TSN-capable or cannot accept external time sources at all requiring a source of time that may not be traceable to UTC. A non-traceable clock would not be appropriate to use for logging or other administrative functions because it may not be comparable with clocks in other work cells.

The IEEE 60802 standard addresses this challenge by defining a dual-clock framework with two independent time sources per work cell. The "working clock" is private to each work cell and may not be UTC-traceable, while "global time" is always UTC-traceable for cross-facility logging and debugging. Both clocks are distributed via PTP with separate GM instances, and the Clock Manager monitors both by tagging synchronization status reports, enabling applications to make decisions like shutting down work cells based on timing reliability. Linux kernel 6.16 now supports this multi-clock approach through auxiliary clocks accessible via standard POSIX APIs.

The IEEE 60802 standard also incorporates hot-standby redundancy defined in IEEE 802.1ASdm, providing near instantaneous failover compared to the default best timeTransmitter clock algorithm (BTCA) mechanism that can take tens of milliseconds. Hot-standby redundancy uses two GMs designated primary and secondary time sources that represent the same timescale: the working clock or global time. The endpoint devices switch to the secondary time source if the primary is unavailable. The Clock Manager enables redundancy by reporting status from the primary and secondary time sources allowing seamless failover. Use of hot-standby redundancy does not require changes to operating system APIs or changes to the application business logic.

Conclusion

Clock Manager is more than a monitoring tool—it's a strategic enabler for industrial reliability, safety, and scalability. By providing real-time synchronization visibility without licensing risks, it empowers businesses to maintain up time, protect assets, and optimize performance.

Ready to make synchronization management effortless? Explore Clock Manager today at https://github.com/erezgeva/libptpmgmt/blob/master/clkmgr/README.md.

intel

0226/DZ/PDF 368170-001US