Closing the AI Networking Visibility Gap

Artificial intelligence (AI) is rapidly moving from specialized research to mainstream applications reshaping entire industries. Modern AI workloads demand unprecedented scale, performance, and reliability from GPU clusters. Yet as clusters grow to thousands of GPUs and tens of thousands of links—interconnected by high-performance networks like InfiniBand or RoCE—**network fragility** has emerged as a persistent bottleneck.

To meet these demands, **AI network observability** must go beyond basic monitoring. It must address both **failures** (e.g., link flaps, NIC faults, misconfigurations) and **performance issues** (e.g., congestion, contention, path asymmetry). This requires real-time, fabric-level visibility, fine-grained latency and jitter metrics, and the ability to correlate network anomalies directly to job-level symptoms like slowdowns or restarts.

This white paper introduces **Clockwork Software-Driven Fabrics**, a new approach designed to close the AI networking visibility gap and significantly boost performance and reliability at scale.

Network Failures and Performance Issues Contribute to Poor GPU Cluster Utilization

AI training and large-scale distributed inference have fundamentally reshaped the performance demands on data center networks. State-of-the-art models such as GPT, Gemini, and LLaMA use sophisticated forms of parallelism, where each GPU computes gradients during the forward and backward pass, followed by collective operations (typically AllReduce) to synchronize weight updates across all replicas. These workloads generate highly bursty traffic, leading to flow collisions during NCCL operations (e.g., all-reduce or all-to-all operations) or during simultaneous inference bursts.

This traffic pattern drives up latency, reduces throughput, and directly impacts job completion times and token streaming performance. As a result, AI workloads are not only compute-intensive but also **tightly synchronized and communication-bound**.

Despite advances in hardware and cluster architecture, effective **Model FLOPs Utilization (MFU)** has stalled around 35–40%, largely due to communication bottlenecks and intermittent network failures.

Even brief network slowdowns can leave thousands of GPUs idle, waiting on stragglers–turning minor hiccups into substantial operational inefficiencies and slower iteration cycles.

A key challenge exacerbating these issues is the **visibility gap**: engineering and infrastructure teams lack real-time insights into network connectivity, path quality, or message-level performance across GPU clusters. Critial Metrics such as per-link throughput and per-job latency are often inaccessible or fragmented across tools. As a result, network problems often go undetected until they cause degraded job throughput or failures. **This limited visibility hinders root cause analysis, extends mean time to recovery (MTTR), and forces teams into a reactive posture—reducing overall system efficiency and GPU utilization.**

Common Runtime Network Issues in GPU Clusters

In multi-GPU clusters, networking problems can severely impact job completion time. Key runtime issues include:

- **Network Misconfigurations:** Slight misconfigurations in an InfiniBand or RoCE fabric (e.g. incorrect subnet manager settings, missing flow-control settings) can degrade the entire system's performance. For example, mis-tuned Priority Flow Control (PFC) or ECN on a RoCE network can lead to unexpected packet loss or stalls. InfiniBand's complexity means that any configuration mistake (MTU mismatches, wrong routing table, etc.) may create bottlenecks or instabilities. These misconfigs often go unnoticed until training performance drops significantly.

- **Link Flaps and Failures:** Physical link failures or flapping connections are among the most common problems in large GPU clusters. In practice, **InfiniBand error codes** like *"Link downed"* or *"Link went down"* (UFM error codes 112 and 329) are red flags often associated with training crashes. Experience from hyperscalers like Meta and Microsoft shows that a cluster of 8,000 GPUs may experience 5–20 link flaps per day and **~15% of all flaps are disruptive**, requiring manual intervention - such as cleaning or reseating optics - and can persist for tens of minutes. During this time, collective operations stall, causing timeouts and ultimately forcing the job to restart from its last checkpoint. **In a 1,000-GPU cluster, these disruptive flaps lead to 1–4 job-impacting events per week**.

Number of GPUs	Job restarts / year	Mean time to failure
1000 GPUs	100 - 250	35-87 hours
5000 GPUs	500 - 1250	7 - 18 hours
10,000 GPUs	1000 - 2500	3.5 - 9 hours
50,000 GPUs	5000 - 12,500	42 - 105 minutes

- **Congestion Hotspots and Incast:** AI workloads produce bursty, synchronized traffic (e.g., during AllReduce), often converging on shared links and causing incast congestion. Meta's experience with RoCE clusters showed that uneven traffic distribution can degrade training performance by

>30% due to congested uplinks . Even in a fat-tree InfiniBand network advertised as non-blocking, congestion can occur transiently if adaptive routing isn't perfect. These congestion hotspots manifest as longer communication times (e.g. all-reduce taking much longer than expected) and thus slower training iterations.

- **Path Asymmetry and Load Imbalance:** Large fabrics use multipath routing (e.g., ECMP in Ethernet or InfiniBand lanes) to spread traffic. However, if routing is suboptimal, you get **path asymmetry** – some routes carry heavier load or have higher latency than others. Meta's early RoCE deployments using static flow pinning saw one uplink saturated while others idled, leading to 30% slowdowns. Failures exacerbate the issue, as re-routed flows often collide on the same backup path. In InfiniBand, which traditionally uses static routing tables, similar imbalances can occur if the subnet manager's routing algorithm isn't tuned for the workload pattern (trees or rings in collective communication). These asymmetries turn some GPUs into stragglers, delaying the entire job.

- **NCCL Timeout Errors**: NCCL (NVIDIA Collective Communications Library) timeout errors are notoriously difficult to diagnose, with broad impact across the system stack and root causes that are often transient. Due to the asynchronous nature of CUDA execution, failures may not surface immediately, obscuring their origin. Transient link errors—caused by brief network congestion, hardware glitches, or environmental factors—are hard to reproduce and debug. Making matters worse, NCCL errors often lack granular logs, offering little insight into which rank or node triggered the failure, further complicating resolution.

Four Visibility Gaps in AI Networking: Challenges with Today's Monitoring Tools

Despite the prevalence of network-induced slowdowns, most GPU cluster operators have limited insight into *when and where* these network issues strike. Traditional monitoring tools tend to focus on node health (CPU, GPU errors, memory, etc.) and basic network pings, but they do not capture the nuanced performance of the network fabric under load. In practice, many network problems are discovered only *after* they have impacted a job.

The four primary observability gaps in AI networking are:

1. **Delayed Detection and Diagnosis:** Many network issues—such as link flaps, port errors, or NIC failures—are **only detected after they've already degraded performance or caused job failures**. InfiniBand fabrics often log these faults *after* a job has stalled or crashed. As SemiAnalysis points out, critical alerts like port health warnings typically arrive too late to prevent disruption. The default NCCL watchdog timeout (~30 minutes) allows jobs to hang silently, wasting compute resources before being aborted. Worse, slowdowns caused by silent degradations—such as link congestion or symbol errors—rarely trigger alerts because traffic continues to flow, albeit inefficiently. In real incidents, jobs have run for hours at up to 10% reduced speed due to a single congested link—completely unnoticed by standard monitoring.

2. **Limited and Fragmented Visibility Across the Stack:** Existing monitoring tools operate in silos and offer only partial insights. For example, NVIDIA DCGM provides detailed GPU telemetry but lacks visibility into NICs or fabric health. Meanwhile, network tools like UFM or NetQ monitor switches and links but have no awareness of which jobs are using which resources. In-band and out-of-band data are often isolated, making correlation difficult. These gaps are especially pronounced in multi-vendor or hybrid environments. InfiniBand UFM offers rich telemetry but is proprietary and difficult to integrate with open-source dashboards. Many operators forego it entirely, relying on scripts and manual checks that are error-prone and hard to scale. NetQ works well for Ethernet, but InfiniBand users are left with fragmented solutions like OpenSM. Full-system observability—spanning GPUs, NICs, switches, and fabric software—remains elusive.

3. **Lack of End-to-End Job-to-Network Correlation:** Current observability tools fail to connect network-layer anomalies with job-level symptoms like slowdowns, retries, or failures. Fabric managers can track traffic patterns but lack visibility into which jobs are affected. Conversely, GPU telemetry tools like NVIDIA DCGM offer no insight into the underlying network paths or issues.

   This disconnect forces engineers to manually correlate job stalls—such as degraded all-reduce performance or NCCL retries—with switch logs, port statistics, or congestion metrics. In many cases, the root cause is far removed from where the symptom appears. A NIC or NVLink fault may stall a job several hops downstream, with no clear error message. Without GPU-to-GPU visibility across NICs, switches, and fabric software, root cause analysis becomes slow, error-prone, and operationally expensive.

4. **Limited Congestion Awareness and Root Cause Resolution:** Current tools struggle to differentiate between congestion, hardware degradation, and topology imbalances—despite all presenting similar symptoms like slow traffic, packet loss, or retransmissions. For example, InfiniBand may report retransmissions due to credit timeouts, but these can result from either transient congestion or persistent hardware faults, making diagnosis ambiguous.

   Most tools only provide per-link aggregate counters, lacking directional or per-flow insights. This makes it difficult to detect asymmetric paths or pinpoint specific bottlenecks. Short-lived or bursty congestion often goes undetected by coarse sampling intervals (e.g., 30-second Prometheus scrapes). Without high-resolution, correlated in-band and out-of-band telemetry, distinguishing between temporary saturation and persistent failure becomes guesswork—leading to delayed mitigation and unreliable job performance.

Business Impacts of Visibility Gap

Long Response Time and Manual Interventions: Visibility gaps make mitigation harder. When the root cause of a slowdown is unclear, teams often resort to checkpointing and restarting jobs—hoping the issue was transient or that new nodes will perform better. This trial-and-error approach is inefficient. Meta's LLaMA 3 logs reported 419 unexpected job interruptions over 54 days, with 8.4% linked to network switch or cable issues. While most were resolved through automation, complex cases still required manual debugging—highlighting how difficult pinpointing failures can be, even for top AI teams. **Each manual intervention means idle GPUs and lost time**, directly impacting job completion and cluster efficiency.

Lost Productivity, Wasted Spend, and Delayed Time-to-Market: Poor network visibility directly translates into wasted spend. When GPUs sit idle due to undetected slowdowns, organizations pay for compute they're not effectively using—while still incurring power and rental costs. Industry analysts estimate that GPU underutilization from orchestration and networking bottlenecks is a multi-billion dollar problem. For example, if half a cluster's capacity waits on data, that's a 50% hit to ROI. Training delays also carry opportunity costs—slower model delivery, reduced experimentation, and longer time-to-market. Meanwhile, engineering teams lose productivity firefighting crashes and slowdowns. Many compensate by over-provisioning GPUs or adding conservative checkpoints—workarounds that further inflate cost and complexity.

In summary, existing tools leave a significant observability gap—critical network metrics and failure signals aren't surfaced in real time to the teams that need them. This leads to technical inefficiency, wasted compute, and lost time. Closing this gap could unlock major gains: higher GPU utilization, faster job completion, and greater confidence in large-scale training. That's the core motivation behind **Clockwork's software-driven fabric approach**—designed to deliver end-to-end visibility and control across the GPU networking stack.

The remainder of this document explores how **Clockwork's FleetIQ** platform addresses the visibility gap by delivering fine-grained, real-time network insights through a combination of out-of-band and in-band telemetry.

Clockwork Software-Driven Fabrics

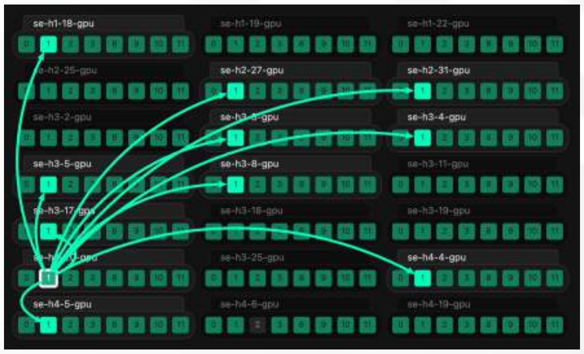
Clockwork's mission is to "Accelerate AI with fast, functional fabrics". Clockwork's Software-Driven Fabric (SDF) architecture leverages software instead of proprietary hardware to deliver resilience, determinism, and superior price-performance.

Instead of treating the network as a static, best-effort medium, Clockwork instruments it in software to achieve two key capabilities: **(a)** fleet-wide active monitoring of network health via an out-of-band probe mesh, and **(b)** in-band telemetry and optimization of actual workload traffic. These twin components work together to close the visibility gap and enhance reliability and performance. Importantly, Clockwork's solution is 100% software-based – it does not require specialized hardware or proprietary network modifications . It runs on standard Ethernet (RoCE) or InfiniBand networks, scaling from modest clusters to ultra-large supercomputers.

This capability is powered by breakthrough research in [software-based clock synchronization]() and techniques for [inferring queueing dynamics from edge data](). At its core, **Clockwork's Global Clocksync** solution aligns every node's internal clock to within nanoseconds, using a lightweight, peer-to-peer probe mesh and applying machine learning and graph optimization to achieve near-perfect synchronization. The result is tens of thousands of nodes - within a data center or across regions - operating on a unified nanoseconds-accurate timeline!

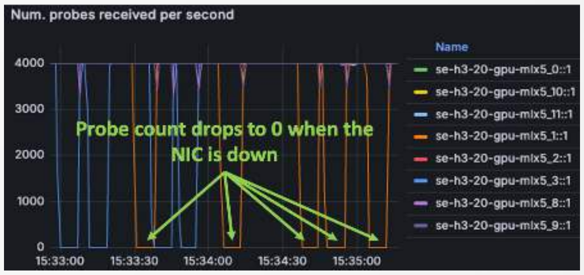
Because every host now shares a common timeline, the same probe mesh doubles as the foundation for ultra-dense telemetry fabric. Infrastructure teams can use Clockwork's dashboards and APIs to see the fabric from two angles at once: monitoring everything from fabric health during provisioning to per-job efficiency in production, raising the right alert - link, node or workload - before users notice a slowdown.

Fleet Monitoring (Out-of-Band Probe Mesh)



Clockwork deploys an **out-of-band probe mesh** through lightweight agents running on each node. These agents continuously measure **one-way delays (OWDs)** between NIC pairs, forming a real-time view of network health without interfering with workload traffic. This probe mesh offers fine-grained insights into:

- **Network Topology:** Measured OWDs are used to construct a dynamic, hierarchical view of the cluster, grouping nodes by proximity for optimized scheduling and diagnostics.



- **NIC Connectivity:** If probes stop flowing between NICs for a defined interval (e.g., 1 second), the agent flags a disconnection and raises an alert immediately.

- **Network Performance:** When probes experience increased latency due to congestion or degraded paths, those delays are recorded and, if thresholds are crossed, trigger alerts.

This continuous monitoring allows Clockwork to proactively detect link failures, flapping ports, or congestion hotspots—even on idle paths. All probe data feeds into a real-time, fleet-wide map of the network, enabling automated response.

Unlike traditional systems that rely on periodic health checks or coarse metrics, Clockwork's **probe mesh acts like a live EKG for the data center**, delivering always-on visibility and enabling the software fabric to adapt dynamically—keeping GPUs productive and training jobs resilient, even during transient network faults.

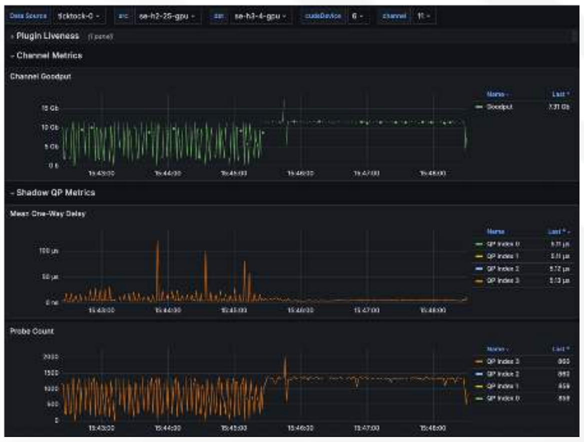
Workload Monitoring (In-Band Telemetry)

In addition to out-of-band probing, Clockwork instruments **in-band telemetry** by embedding lightweight instrumentation directly into live AI workloads. This is enabled via a **custom NCCL/RCCL plugin** (or a message-aware Libibverbs plugin), which Clockwork provides as a shared-object library. NCCL dynamically loads this plugin at runtime using standard environment variables—no changes to user code are required.

Thanks to Clockwork's sub-microsecond global clock synchronization across NICs, Clockwork can measure latency and performance directly in the data path with extreme high precision without requiring specialized hardware or vendor-specific telemetry hooks.

The plugin integrates at the **Collective Transport Layer**, using the InfiniBand verbs API to manage data queue pairs (QPs), tags data packets and captures **queue-pair–level and chunk-level metrics**, including:

- **OWDs** between GPUs
- Per-collective **throughput**
- **ECN** markings
- Real-time **flow composition** between GPU pairs



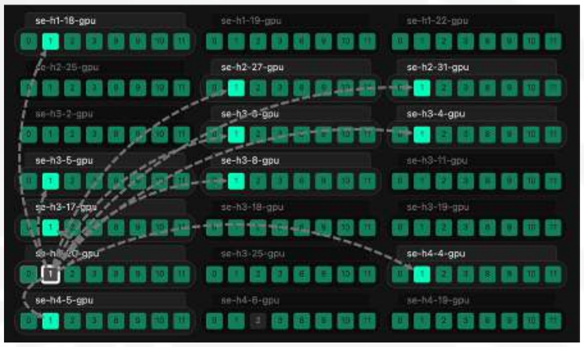
This approach gives deep, real-time visibility into the behavior of running jobs across the fabric. Infrastructure and AI engineers gain fine-grained, per-job metrics alongside fleet-wide insights, enabling rapid detection of anomalies like congestion or uneven throughput. The result: more consistent performance, faster job completion, and fewer surprises.

Example Use Cases

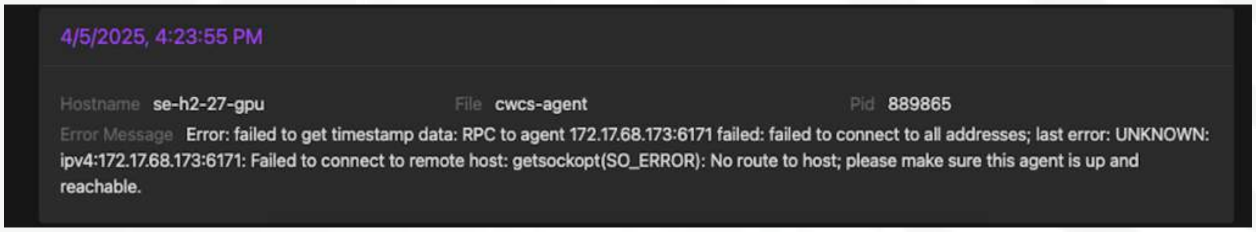
1. Real-Time Monitoring of Connectivity and Link Flaps

Clockwork Fleet Monitoring (out-of-band) continuously monitors both **Front-End** and **Back-End** network connectivity, detecting link flaps and failures in real time and raising alerts.

- **Back-End Network:** In this example, NIC 1 on host se-h3-20-gpu is repeatedly flapping between connected and disconnected states. When the NIC fails to send or receive probe traffic, it is marked **unhealthy** (shown in gray). Once probe flow resumes, the NIC status returns to **healthy** (green).



- **Front-End Network:** At approximately **4:23 PM**, node se-h2-27-gpu became unable to probe its peers on the front-end network. This loss of connectivity was immediately flagged by probes.



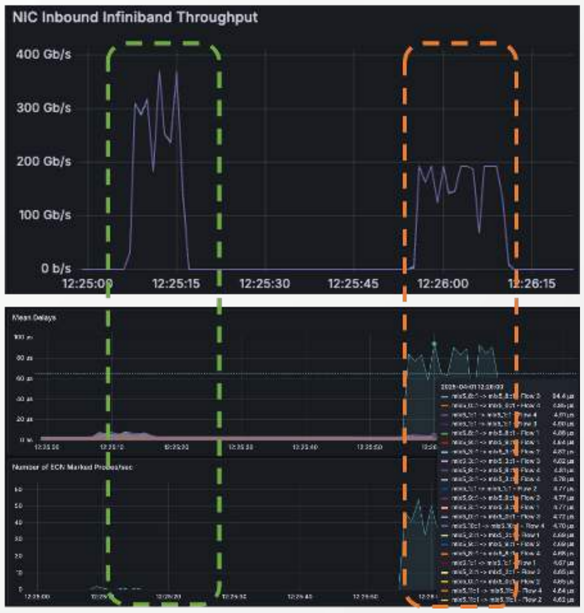
```
4/5/2025, 4:23:55 PM

Hostname  se-h2-27-gpu          File  cwcs-agent          Pid  889865
Error Message  Error: failed to get timestamp data: RPC to agent 172.17.68.173:6171 failed: failed to connect to all addresses; last error: UNKNOWN:
ipv4:172.17.68.173:6171: Failed to connect to remote host: getsockopt(SO_ERROR): No route to host; please make sure this agent is up and
reachable.
```

2. Detecting Network Bottlenecks in the Fabric

In this example, the all_reduce_perf workload was run twice. Using **Clockwork Workload monitoring** which measures both **One-Way Delays (OWDs)** and throughput at the granularity of a **Queue Pair (QP)**.

- **Green run:** Achieved ~360 Gbps
- **Orange run:** Dropped to ~190 Gbps

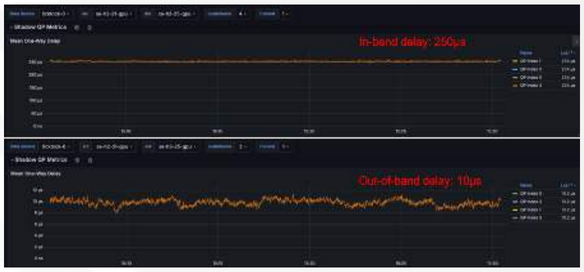
The performance drop in the second run was caused by a significant delay on **one specific flow** (94 µs), while all other flows remained low-latency (~4 µs). This imbalance reveals that traffic contention occurred on a particular spine port, resulting in uneven load distribution and degraded job throughput.



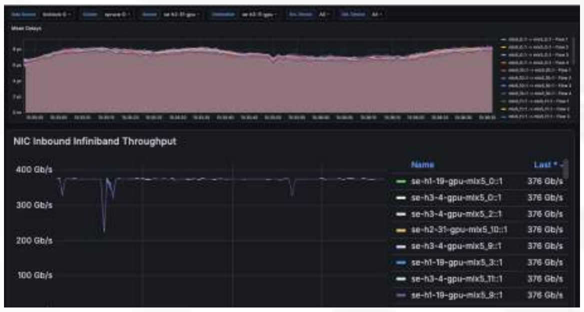
3. Detect & Resolve Workload Misconfigurations

By combining **Fleet Monitoring** (out-of-band) and **Workload Monitoring** (in-band), Clockwork can surface subtle misconfigurations that are otherwise hard to catch.

Diagnosis: A mismatch was detected between out-of-band and in-band **one-way delays** at the Queue Pair level. Further investigation revealed the workload was mistakenly using **RoCEv1** instead of **RoCEv2**.



Remedy: After reconfiguring the workload to use **RoCEv2**, throughput stabilized and consistently reached **line rate (~400 Gbps)**, all Queue Pair delays are now around 6us as shown in the accompanying screenshot.



Name	Last *
se-h1-19-gpu-mix5_0::1	376 Gb/s
se-h3-4-gpu-mix5_0::1	376 Gb/s
se-h3-4-gpu-mix5_2::1	376 Gb/s
se-h2-31-gpu-mix5_10::1	376 Gb/s
se-h3-4-gpu-mix5_9::1	376 Gb/s
se-h1-19-gpu-mix5_3::1	376 Gb/s
se-h3-4-gpu-mix5_11::1	376 Gb/s
se-h1-19-gpu-mix5_8::1	376 Gb/s

To request a demo, contact us at **hello@clockwork.io**