Bandwidth Works in Practice, not in Theory

Sahas Munamala, Varun Datta, Dean Gladish, Paul Borrill, DÆDÆLUS Research

June 5, 2025

Ethernet, for the last 5 decades, has operated under the assumption that reliability is firmly within the domain of applications. This has allowed network hardware to design failure modes of packet transmission into normal operation, and sell it as bandwidth. However, there is more to the story of reliability than reconstructing exactly-once semantics with idempotent API's, nonce tokens, and retry logic. There is a simple fact: once the epistemic knowledge of an event is lost, it is unrecoverable by either sender or receiver. No amount of timeout-and-retry or fail-fast design principles can recover the exactly-once event after it is lost.

Introduction

The computer networking industry has been marking its progress by the theoretical maximum bandwidth of links. The exponential increase of bits that fit on a 3 m copper cable make a latency argument by claiming the frame will arrive earlier because it transmits faster. On point-to-point connections, this is true, based on the current definitions of bandwidth and latency. However, there is more to bandwidth than # of bits/second that makes it useful, and there's more nuance to latency that will stall distributed applications no matter how fast the hardware gets.

Round-trip interactions are a fundamental unit of computation, particularly in distributed systems, networked computing, and interactive AI workflows. However, conventional networking thinking has placed this as a requirement of Layer 3, as an optional transport layer, and not a fundamental method of networked communication. The result of this is TCP, where a Layer 3 protocol between endpoints negotiates the bidirectional transmission of variable length messages on an unreliable network.

Hidden cost of Bandwidth-First

> *If my link runs at 400 Gbps instead of 100 Gbps, then even if my network is lossy or messy, each packet gets through faster — so who cares about loss, jitter, or flow control?*

Raw, one-way Bandwidth metrics alone fail to account for the crucial aspect of round-trip reliability – the guaranteed and verifiable transfer of packets between nodes. Such guarantees require explicit handshakes by the hardware to properly transfer ownership and responsibility of each packet with the lowest possible latency. Without these mechanisms, software interfaces cannot trust intermediate nodes to handle their tokens responsibly.

In contrast, bandwidth-maximizing designs focus primarily on pushing bit streams at peak throughput. Their impressive bandwidth benchmarks are typically achieved by sending large, uninterrupted byte sequences that minimize overhead. These systems are engineered to drop packets during congestion, prioritizing throughput numbers over the integrity or reliability of tokens.

Network congestion represents more than inefficiency, it threatens the epistemic state of distributed systems. When a packet is dropped due to congestion, the information it carried vanishes completely, irretrievably erasing knowledge of the event it represented. This loss isn't just temporary. It's fundamental and irreversible. Without this information, no node can reconstruct or even verify whether the event actually occurred. Exactly-once semantics rely entirely upon preserving and transferring this epistemic state across nodes. This epistemic collapse introduces ambiguity, manifesting as inconsistency and grey failures[1]. Thus, the hidden peril of the bandwidth-first approach emerges clearly: by optimizing purely for throughput at the expense of reliable delivery, one risks catastrophic losses in epistemic certainty, fundamentally undermining the correctness and reliability of distributed computation.

[1] Grey failures represent a class of failure where events are partially known or suspected, but never fully provable

InfiniBand is Right

Shannon showed us that redundancy is necessary to overcome noise, but it doesn't have to come in the form of retransmissions. It can come from timing, structure, flow control, and error-detecting codes. InfiniBand[2], by fixing the message structure and introducing deterministic flow control, turns a noisy, uncertain channel into a nearly noiseless, deterministic pipeline, analogous to a physical circuit.

[2] InfiniBand guarantees lossless transmission even under congestion by using credit-based flow control and hardware backpressure. Instead of dropping packets, it prevents senders from overrunning buffers, ensuring reliable delivery without retries.

Fixed-size packets enable efficient buffer management, deterministic flow control, and cut-through switching— critical for maintaining lossless transmission. From a Shannon theory perspective, fixed packet sizes simplify the encoding and decoding process by reducing entropy per symbol and minimizing variance in transmission time, which stabilizes throughput near channel capacity.

InfiniBand is a layered protocol stack, but it does not rely on the TCP/IP model.

Why didn't InfiniBand win?

InfiniBand offered clear technical advantages, including guaranteed lossless delivery, extremely low latency through cut-through switching, and efficient memory transfers via RDMA. However, it failed to achieve mainstream adoption beyond high-performance computing and certain enterprise deployments. This outcome resulted from a combination of economic, architectural, and ecosystem realities. InfiniBand required specialized hardware, tight coupling, and strict credit-based flow control, making it difficult to scale and integrate in diverse environments. While these features served scientific and tightly synchronized workloads well, they were overengineered for the broader datacenter and cloud market.

Ethernet, in contrast, embraced simplicity, flexibility, and continuous evolution. Its early limitations were addressed through layered improvements such as Data Center Bridging, Priority Flow Control, and RDMA over Converged Ethernet. These additions made Ethernet "good enough" for many high-throughput, low-latency applications without abandoning compatibility with legacy systems. Ethernet also benefited from a wide base of vendors, lower costs driven by volume, and relentless IEEE standardization.

Ultimately, InfiniBand was a specialized solution in a world that prioritized openness, cost efficiency, and broad adoption. Rather than merge with InfiniBand, the Ethernet ecosystem chose to adapt and absorb its most useful features. The result was not technical defeat but strategic obsolescence.

The acquisition of Mellanox[3] by NVIDIA in 2020 was a significant move that sent ripples through Silicon Valley, though it did not cause a dramatic upheaval in the traditional sense. It was less of a shockwave and more of a strategic signal—one that made clear NVIDIA's ambitions to move beyond GPUs and into the heart of data center infrastructure.

[3] Mellanox was the steward and primary driver of InfiniBand technology. It played a central role in both the development and commercialization of the InfiniBand standard, acting as the lead implementer and key evangelist within the HPC and low-latency networking community.

AI training benefits enormously from lossless networking, but it's not because the algorithms require it for correctness. Rather, it's about efficiency, scale, and determinism in distributed computation. Reliable networking seamlessly, and near-losslessly, extends GPU memory space across multiple GPUs, allowing compute pods to share memory like a single fast memory pool.

Are Reliable Networks still Niche Today?

Applications require predictable and deterministic behavior from the networks they rely upon. Every layer – from API down to the physical transmission of bits on the wire and back – must preserve the semantics of exactly-once requests.

Today, applications are forced to tolerate all forms of network unreliability in their communication because lossless, deterministic networks are prohibitively expensive, proprietary, or specialized. However, modern applications from distributed microservices, replicated databases, robotics, and automation, are fundamentally dependent on predictable, exactly-once semantics. These applications crave reliability because it vastly simplifies their internal logic, improves consistency guarantees, and significantly reduces the burden of handling retries, duplicates, and partial failures.

With easy and open access to a fully verified, open-source standard for reliable communication, applications can trust their infrastructure implicitly, dramatically simplifying their design and enhancing operational robustness. The availability of such open, reliable standards democratizes technology previously reserved for niche, high-budget environments, enabling widespread adoption and fundamentally reshaping expectations around distributed system reliability.

New Metrics for Networks