0.1 Interactions, not Bandwidth

Traditional networks treat bandwidth as a fungible resource; Like a pipe to be filled as much and as fast as possible. Success is measured in utilization, and failure in dropped packets. In contrast, Æthernet redefines communication as a series of causal interactions between peers, where each exchange has semantic weight and is governed by `LINK` state machines.

In this model, throughput is an emergent property of sustained, reversible token exchanges — not burst transmission. Initiators flow frames toward responders without waiting, and responders flow responses back in kind. The rate of progress is governed not by the raw link speed, but by the rate of acknowledged interaction. This change in framing brings several practical consequences:

- **Stable Congestion Behavior**: Systems built on causal flow control naturally avoid head-of-line blocking and buffer overflows, especially under sustained high load.
- **Implicit Clock Recovery**: Each interaction provides timing and synchronization cues, enabling robust clock alignment without separate timing channels.
- **Minimized Latency Variance**: Because there are no speculative transmissions, queuing and jitter are dramatically reduced, even under full load.
- **Atomic Forward Progress**: A transfer either completes causally or is fully rolled back, preserving global consistency without the need for speculative multi-path packet spraying.

0.1.1 Hidden cost of Bandwidth-First

Raw, one-way Bandwidth metrics alone fail to account for the crucial aspect of round-trip reliability–the guaranteed and verifiable transfer of knowledge 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 knowledge 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.

Consider a lossy link with infinite bandwidth, as illustrated in Figure **??**. In this hypothetical, the bottleneck is not raw capacity but the

Figure 1: A infinite bandwidth pipe with packet loss can still limit throughput of TCP flows.

$$\mathrm{BW} = \frac{\mathrm{MSS}}{\mathrm{RTT}} \cdot \frac{C}{\sqrt{p}}$$

MSS: max segment size
RTT: round-trip time
C: constant (1.22)
p: packet loss probability

twin constraints of latency and packet loss. Since TCP relies on acknowledgments to regulate its sending rate, the time it takes to complete a round trip –RTT – becomes a limiting factor on throughput. As shown by the Mathis equation**?**, throughput degrades proportionally to the inverse of RTT and the square root of the loss probability. In such regimes, increasing bandwidth alone does not improve performance; if anything, the absence of reliable round-trip feedback renders the network incapable of sustaining high-throughput flows. Even in a system of perfect raw transmission capacity, epistemic uncertainty introduced by loss and latency can strangle performance. Round trips are essential to any communication that requires certainty, ordering, or acknowledgment.

In practical networks where congestion is a real and dynamic force, TCP flows must adapt their behavior to avoid collapse. This adaptation is governed by Additive Increase/Multiplicative Decrease (AIMD) – a deceptively simple algorithm that allows each sender to probe the available capacity of the network while reacting swiftly to congestion signals. Each flow increases its sending window linearly over time, but upon detecting loss (interpreted as a sign of congestion), it slashes the rate multiplicatively. This feedback loop produces a sawtooth pattern in throughput, enabling multiple flows to converge to a fair, stable sharing of the underlying path. Crucially, AIMD is fully decentralized and stateless beyond the endpoints. Yet, this elegant self-regulation only functions when loss reflects congestion, and when RTT remains a trustworthy signal of delay. In networks where loss is stochastic or induced by buffer mismanagement, AIMD underperforms or misbehaves **?** – but in its ideal regime, it is a marvel of distributed equilibrium: each sender, optimizing selfishly, contributes to global stability.

Packet loss and network congestion represent more than inefficiency, they threaten the epistemic state of distributed systems. When a packet is dropped due to congestion, the information it carried vanishes completely, erasing knowledge of the event it represented. This loss isn't just temporary. It's fundamental and irreversible. Without this information, no node or application can know if the packet is coming late, or not at all. Exactly-once semantics rely entirely upon preserving and transferring this epistemic state across nodes. Failures in handling epistemic state leads to 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

0.1.2 Are Acknowledgments Expensive? Not Anymore

Traditional wisdom paints acknowledgments as throughput killers: in stop-and-wait protocols, the sender halts until it receives confirmation before transmitting again. But in modern Ethernet–particularly at 100 Gbps over short links–this notion is obsolete. In fact, acknowledgments can be issued and received *in-flight*, with virtually no cost to throughput.

Let's examine why.

- **Frame size (including overhead)**:

$$64\text{B} + 8\text{B}\,(\text{preamble}) + 12\text{B}\,(\text{IPG}) = 84\,\text{bytes} = 672\,\text{bits}$$

- **Transmit time at 100 Gbps**:

$$T_{\text{tx}} = \frac{672\text{ bits}}{100 \times 10^9} = 6.72\text{ ns}$$

- **Propagation speed**: 2×10^8 m/s (5 ns per meter)

$$\text{Length on wire} = T_{\text{tx}} \cdot v = 6.72\,\text{ns} \times 2 \times 10^8\text{ m/s} = 1.344\text{ m}$$

That means the entire frame is over a meter long on the wire–longer than many physical links. For links under this length, the first bits of the frame can arrive at the receiver before the last bits have even left the sender.

This framing leads to a surprising result: on short links, round-trip time (RTT) is often shorter than transmission time. The table below compares these values and shows utilization ($U = \frac{T_{\text{tx}}}{T_{\text{tx}}+\text{RTT}}$):

Cable Length	RTT (ns)	T_{tx} (ns)	Utilization (U)
10 m	102.4	6.72	6.16%
1 m	12.4	6.72	35.14%
10 cm	3.4	6.72	66.40%
1 cm	2.5	6.72	72.88%

The shorter the link, the more transmission dominates RTT, and the higher the achievable utilization—even with a per-packet acknowledgment model.

0.1.3 Æthernet: Circulating Snakes

In this model, each packet and its response form a closed-loop interaction: a reversible token with a forward and return path. Like a snake on a track, the packet's body spans the link, and the acknowledgment follows behind, curling the path into a circuit of semantic closure. The only idle space is the inter-packet gap between snakes.

Figure 2: In short links, a 64-byte packet spans the wire, with its acknowledgment returning before transmission completes—forming a circulating snake.

This has profound implications:

- **Acknowledgments are in-flight, not blocking**.
- **Causality is enforced at hardware speed**.
- **Utilization is limited only by the spacing between interactions—not protocol overhead**.

There's no stop-and-wait. No idle windows. No speculative retries. Every token sent is causally resolved or rolled back. This transforms acknowledgments from a performance tax into a foundational mechanism for maintaining epistemic certainty.

This leads to a key insight: acknowledgments don't require idle time. They can be:

- **Embedded or piggybacked** in return frames,
- **Pipelined** using snakes as carriers,
- **Issued in parallel** with frame reception.

Modern Ethernet resembles a conveyor belt more than a stoplight. Multiple frames are in motion, bidirectionally, all the time. With proper credit-based flow control, there's no need to wait for a response before initiating the next action.

Acknowledgments–once considered expensive–are now cheap, even free, on modern high-speed links. As frames stretch out physically and RTT shrinks, acknowledgment can arrive before the sender finishes transmission. Add to this the amortization effects of jumbo frames and the elegance of pipelined protocols, and the old "stop-and-wait" bottleneck dissolves.