# 1. Requirements and Use-Cases

## 1.1 Open Atomic Ethernet (OAE) Requirements

### 1.1.1 Core Technical Architecture

1. **Lossless / Near-Zero-Loss Data Plane.** Hardware flow-control or cell/TDM framing to eliminate random drops that invalidate atomic protocols.
2. **In-NIC Atomic Verbs.** One-RTT line-rate operations (e.g. CAS, multi-word commit) at $\geq$100 GbE serialization latency.
3. **Deterministic Scheduling.** Optional mesochronous or slot-based service for bounded latency (AI & HFT workloads).
4. **Reference-Free Causal Ordering.** Protocol must avoid dependence on globally synchronized clocks; use causal trees or hybrid logical clocks.
5. **Self-Stabilizing Control Plane.** Automatic convergence after arbitrary transient faults, no external reset.
6. **Composable Security & Isolation.** Native hooks for side-channel detection and optional crypto/QKD overlays without mandatory encryption.

### 1.1.2 Open Software Stack

1. **Open-Source Reference NIC RTL.** Minimal FPGA bit-file implementing OAE atomics; EFVI/Libfabric-compatible.
2. **Portable Verbs Library.** Unified C/C++/Rust API usable beneath Derecho, Raft, Aeron, Kafka, etc.
3. **Self-Healing Membership Service.** Gossip bootstrap $\rightarrow$ leader-based virtual synchrony (Derecho-style) with robust failure detection.
4. **Lightweight Management Telemetry.** Binary or gRPC schema (REST/Redfish only for out-of-band).

### 1.1.3 Research & Validation

1. Formal proofs for atomic commit without global time, gradient-clock bounds, and self-stabilizing recovery.
2. Tail-latency & failure-injection benchmark versus RoCE, DPDK, AF_XDP, Ultra-Ethernet, Tesla-TTP.
3. Hybrid quantum/classical shim for future Q-NICs.

### 1.1.4 Ecosystem & Governance

1. Public problem-statement registry (GitHub).

2. Sub-groups: Data plane; Control plane; Timing; Security; Business & Market.
3. External expert engagement (Rachee Singh, Shlomi Dolev, Roel de Vries, Chris Batten, Chris Rossbach, Tanya Zelevinsky, *etc.*)
4. Dedicated business-case work-stream quantifying TAM versus incumbent RDMA/RoCE.

### 1.1.5 Near-Term Deliverables (next 3–6 months)

| # | Deliverable | Owner | Target Date |
|---|---|---|---|
| 1 | OAE Problem Statement v1.0 (15 pp) | Draft team | 31 May 2025 |
| 2 | FPGA NIC demo (64 B atomic commit @100 GbE) | Data-plane SG | 30 Jun 2025 |
| 3 | OpenAPI verbs library (C & Rust) | Software SG | 15 Jul 2025 |
| 4 | Self-stabilizing membership prototype | Control SG | 15 Aug 2025 |
| 5 | Market & cost model white-paper | Business SG | 30 Aug 2025 |

---

*Draft for discussion—please annotate, refine, and assign owners at the next OAE Working Group call.*

## 1.2 Ethernet 2025 Requirements

### 1.2.1 Scope

- **Target deployments**:
  - Chiplet–to–chiplet interconnects *within a single package.*
  - In–rack, short–reach data-centre fabrics (one rack or a tightly-coupled pod).
- **Out of scope (initial release)**: traditional multi-rack Clos networks, WAN, metro, access, fieldbus and automotive links.

### 1.2.2 Functional Requirements

1. **Deterministic / Bounded-Latency Delivery**: provide end-to-end latency and jitter bounds tight enough for real-time AI training, HFT and chiplet coherency.
2. **Reliable Message Primitive of Bounded Size**:
   - Atomic "send → deliver" unit (frame or flit) *either arrives whole or is discarded*.
   - Optional acknowledged mode (*reliable-mode*) with automatic local retransmit; must be switch- and NIC-assisted but not mandatory for every packet.
3. **Timeout-And-Retry (TAR) Elimination in Fast Path**: protocol state machines must not rely on software timeouts for forward progress under nominal conditions.
4. **Autonomous Discovery & Configuration**: zero-touch bring-up, addressing and link training inside a rack/package; no DHCP or manual topology files.
5. **Packet-Processing Programmability**:
   - Data-plane micro-ops exposed at Layer 2½ (e.g. P4-like table, eBPF, or HDL modules) for in-line reduction, aggregation, atomics.
   - Configuration must be verifiable and sandboxed.
6. **Minimal Hardware Feature Set for Higher-Level Semantics**: supportive of virtual synchrony, one-sided RDMA, collective (AllReduce) and OLTP commit protocols, but *does not bake them in*.
7. **Classical Quantum Interworking Hooks**: reserve opcode(s) and timing model for future quantum-clock/entanglement assistance without constraining today's PHY/MAC.

### 1.2.3 Performance Requirements

1. **Latency Target**: single-hop store-and-forward <100 ns; cut-through path <50 ns (ASIC to ASIC).
2. **Throughput Target**: scale line-rate from 200 Gb/s (first silicon) to 1.6 Tb/s per lane within ten-year roadmap. Prefer parallel lanes / wavelengths over ever-higher SerDes clocks.
3. **Transactional Round-Trip**: single sender → single receiver ack <300 ns (in-rack worst-case).

### 1.2.4 Reliability, Resilience & "Atomicity"

1. **Loss / Corruption Handling**: per-hop FEC optional; end-to-end CRC > $2^{32}$ for silent error budget in AI workloads.
2. **Membership / Fail-fast Signalling**: hardware assist for fast liveness vote (virtual-synchrony–ready).
3. **Rollback Support**: hardware tag for speculative packets so software can discard/-commit without draining pipeline.
4. **Graceful Degradation**: mandatory congestion / buffer health telemetry; receivers own promise to detect and mitigate incomplete transfers.

### 1.2.5 Security

- **"Full-trust domain" assumption inside package/rack**, but hooks for Zero-Trust when links exit the trust boundary.
- Inline MACsec-lite profile ≤64 ns budget, key-roll without traffic pause.

### 1.2.6 Energy & Physical Layer

1. **Energy per bit**: <2 pJ/bit at 200 Gb/s; target sub-pJ with photonic lanes or advanced copper (e.g. TeraDSL / PAM8).
2. **Clocking**: support distributed, low-skew clock (photonic or electrical) to reduce SERDES power and jitter.
3. **Media**: copper <50 cm, twin-ax++ and/or ribbon fibre; hot-swap cages optional.

### 1.2.7 Manageability & Tooling

- Publish an open reference simulator (NS-3 or similar) and FPGA test-bed.
- Provide formal specs (TLA$^+$, DistAlg) and conformance suites for hardware vendors.
- Continuous telemetry stream: per-flow latency, congestion window, SERDES margin, temperature, link trust-state.

**Note**: The list reflects consensus *proposals* extracted from the chat; several items (e.g. mandatory rollback logic, quantum hooks) remain contentious and should be flagged for ballot.

## 1.3 Original Requirements

### 1.3.1 Charter:

### 1.3.2 Vision:

Ethernet has been the cornerstone of networking for over five decades, adapting to changing technologies, applications, and economic realities. New data center and edge opportunities—such as directly linking chiplet modules in ultra-low-cost meshes, achieving extreme low latency by executing application actions at hardware speed in the network interface, and applying these techniques to distributed updates of persistent storage or to distributed, application-level transactions-push beyond the boundaries Ethernet established in an earlier era. These needs cannot be met simply by extending Ethernet's historical capabilities.

The Ethernet 2025 Workshop and the Open Atomic Ethernet will look ahead to the next 50 years of Ethernet networking. It will focus on ways to achieve higher perfor-

mance, guaranteed service, lower latency, fewer errors, and greater scalability. The idea is to make Ethernet omnipresent and capable of doing the job for everyone.

### 1.3.3 Scope:

Align what the network does with what the distributed application needs to accomplish. Build a world where the network is not allowed to create a mess the application must discover and then clean up, but rather must either ensure both sender and receiver (user space software endpoints in the distributed application) know a message was successfully sent and received, or only the sender "instantly" knows it failed, and no other node or part of the network knows that message ever existed.

This entanglement between sender and receiver, with no library, software, or hardware permitted to disturb that entanglement, is a fundamental change from the "throw it over the wall and hope it gets there" of the last 50 years. From a different perspective, a network message becomes a transaction which exhibits ACID properties: atomicity, isolation, and durability directly, and consistency because that message exists only between endpoints and is not visible to any other observer.

- Rethink Ethernet's core primitives: frame structure, error recovery, and flow control, in the context of Shannon Information Theory.
- ntroduce API's for programmable minimal compute capability at the NIC level to enable reversibility and support application-specific protocol behavior.
- Address the trade-offs between signals, noise and energy dissipation, building upon principles from Quantum Thermodynamics.
- New Fundamental multi-phase primitives: build state machines into the link itself to enable Network assisted transactions.
- Create a protocol framework that is as simple as Ethernet's original design but scalable to today's and tomorrow's use cases, including chiplet interconnects and AI accelerators.

## 1.4 Problem

### 1.4.1 Loss of Temporal Intimacy

1. Maintaining liveness and synchronizing processes in networks is a challenge when packets can be dropped, reordered, duplicated or delayed
   - Conventional networks require protocol stacks and applications to use timeouts and retries (TAR) to maintain liveness
   - This makes exactly-once semantics impossible and precipitates retry storms which lead to unbounded tail latency and transaction failure
     - This results in silent corruption of data structures and loss of data.
   - A problem seen in every distributed database for decades
   - These failures are not understood because customers (under NDA) may not publish results that embarrass vendors
   - The core issue is the "Unreliable" nature of Ethernet. It doesn't have to be this way
   - Nvidia has clearly seen this and understood how to create a "lossless" network with Infiniband
   - The Ethernet community sticks steadfastly to imposing multiple packets on the wire before expecting an acknowledgement.
   - The argument for this (50 years ago, in The Metcalfe + Boggs paper, and Bob Metcalfe's PhD thesis, is that bandwidth will be "throttled" too much if the sender has to wait for the acknowledgment before sending another packet.
   - These assumptions are no longer true in a world of rack-scale and chiplet infrastructure, even with 100Gbit 400Gbit, 800Gbit or 1.6Tbit lanes.
   - While the original Ethernet used "alternating slots" on the cable to transmit frames, Chiplet Ethernet must contend with minimum size (64-byte) frames on the wire

### 1.4.2 Network Faults Lead to Transaction Failure

- 80% of failures have a catastrophic impact, with data loss being the most common (27
- 90% of the failures are silent, the rest produce warnings that are unclear
- 21% of the failures lead to permanent damage to the system.
- This damage persists even after the network partition heals

### 1.4.3 Market

- (Current) Ethernet Has failed to resolve this issue with PFC or other inadequate congestion control algorithms.
- The world has chosen Nvidia.
- Ethernet has become unnecessarily fragmented.
- We need to bring the World back together under ONE Ethernet.

### 1.4.4 Topologies

Physical Topologies are (relatively) static in the Chiplet and Rack-Scale Infrastructure, but Logical and Virtual Topologies built on top of them are reconfigurable by Infrastructure software or operators, and programmable by application engineers within each secure enclave.

- Dedicated Circuit switching
- Link Level Negotiation
- May have cards from different manufacturers

### 1.4.5 Physical Link Protocol:

- All packets use the Alternating Bit Protocol (each packet is acknowledged before the next packet is sent)
- Failures may not be immediately apparent, but they will stop all activity on that link and will minimize the blast radius of failures.
- Receivers may NAK at the Information level, to indicate any kind or error, or to simply refuse the transaction asking the sender to forward it somewhere else.

### 1.4.6 Logical Link Protocol:

The roots of configuration trees don't have to be fixed to physical nodes. Graph Theory allows us to build logical and physical trees on top of physical trees (the groundplane).

### 1.4.7 Virtual Link Protocol:

The roots of application trees don't have to be fixed to logical or physical nodes. New results in graph theory allows applications to program their own tree structures on top of logical trees.

## 1.5 Frame Structure

Ultra-Low Latency is not possible without eliminating unnecessary fields and features in Ethernet's standard frame protocol

- Streams without Headers whenever Point to Point
- No Preamble
- No Inter-record Gap (IRG)
- No Destination mac address
- No Source mac address
- Both sides know who is sending who is receiving and what

### 1.5.1 Error Recovery

- Conventional Ethernet is a one-way Shannon Channel
- At Minimum, it needs "forward" Error detection (FED)
- Increased "one-way" resilience (OWR) omes from "forward" Error correction (FEC)

- FEC unnecessarily increases latency
- Atomic Ethernet is a bidirectional interaction Channel
- Alternating Causality protocol (ACP) (based on ABP) returns the first slice to the sender
- "This is what I heard you say"
- Eliminates the need for FED/FEC
- An entirely new way to think about Shannon Channels,

### 1.5.2 Flow Control

From Alan:

- Separate identity from address: A node can have a unique identifier (GUID) distinct from its address, which can be a small integer.

### 1.5.3 Secure Enclave

Security is impossible without renunciation of the source/destination mode of addressing.

Physical links are individually configured, identified, and provisioned, in seconds, reconfigured in milliseconds, and healed in microseconds.

MAC Addresses (48+ 48) and IP Addresses (32+32 or 128 +128) are susceptible anonymous attacks by byzantine players

- Adjacent (1-hop) links are semi-permanent. It takes physical action or decay processes to break them.
- Neighbor to Neighbor Networks (N2N) provide local privacy and eliminate global node identities

### 1.5.4 Leveraging Formal Methods

Formal Methods are essential to modern infrastructure verification.

## 1.6 Reconfigurable Topologies

### 1.6.1 Self-Adjusting Networks

- Infrastructure Reconfiguration
- Tenant Self-Reconfiguration

### 1.6.2 Application Enablement

### 1.6.3 Transaction Hyperparameter Space Features

- Help manage the rapidly changing state of art software environment.
- Help keep the hardware optimized for best performance, for example, by avoiding power overdraws
- Gerrymander workloads to balance power dissipation and reduce hotspots
- Provide a lighthouse for engineers, so they know where they are in the hyper perimeter space and how their model is doing versus how it could be doing theoretically.

### 1.6.4 Implementation Enablement

- Three key concepts around "Shannon information", the transfer of Shannon information between nodes, and how error recovery is done when that transfer fails or is interrupted, so that certain properties are protected (conserved quantities, "I know that you know that I know", Common Knowledge).
- We understand how to design and build everything above. Several implementations have been built and proven for the reversible transactions, in Javascript, in Rust, and in FPGA implementations of bits on the wire

- We use Wolfram Mathematica and the libraries and tools to explore the space of causal structures in ways that we expect to become standard industry toolchains for FPGA application fragments (specified by Petri nets, translated to rust, and compiled to verilog or VHDL.

## 1.7 FEEDBACK SO FAR

## 1.8 Anonymous1

### 1.8.1 Distributed Systems Problems

I asked Grok what are the unsolvable distributed systems problems. Here is its answer:

"In distributed systems, there are several fundamental problems that are proven to be unsolvable under certain conditions due to inherent limitations like latency, faults, and the lack of a global clock. These challenges arise from the nature of distributed computing, where multiple nodes must coordinate without perfect knowledge of each other's state. Here are some key examples:

1. Consensus in Asynchronous Systems with Faults (FLP Impossibility) The Fischer-Lynch-Paterson (FLP) impossibility result shows that in an asynchronous distributed system, it's impossible to achieve consensus (agreement on a single value) if even one process can fail by crashing. This is because, without synchronized clocks or bounded message delays, the system cannot distinguish between a slow process and a failed one. Any algorithm that tries to guarantee agreement will either risk never terminating or fail to handle all possible failure scenarios.
2. Byzantine Agreement with Too Many Faults In the Byzantine Generals Problem, nodes must agree on a decision despite some nodes behaving maliciously (e.g., sending conflicting messages). It's provably unsolvable if more than one-third of the nodes are faulty in a synchronous system. In an asynchronous system, it's even worse—Byzantine agreement becomes impossible with any faults unless additional assumptions (like trusted components or randomization) are introduced.
3. Perfect Failure Detection Detecting whether a node has failed or is just slow is unsolvable in asynchronous systems. Without a global clock or guaranteed message delivery times, a process waiting for a response can't determine if the other node crashed or if the network is delayed. This forces systems to rely on imperfect heuristics, like timeouts, which can lead to false positives or negatives.
4. Simultaneous Agreement on Global State (Snapshot Problem) Capturing a consistent global snapshot of a distributed system's state is tricky due to the lack of synchronized clocks and the possibility of in-flight messages. Chandy-Lamport's algorithm provides a solution under specific conditions, but without those (e.g., in a fully asynchronous system with unbounded delays), it's impossible to guarantee that all nodes agree on the exact same global state at the same logical "moment."
5. Two-Phase Commit with Network Partitions The Two-Phase Commit (2PC) protocol ensures atomicity across distributed nodes, but it's unsolvable in the face of network partitions. If the coordinator fails or becomes unreachable after sending "prepare" but before "commit," some nodes might be stuck indefinitely, unable to decide whether to proceed or abort. This is a practical manifestation of the CAP theorem, which states you can't have full consistency, availability, and partition tolerance simultaneously.
6. Exact Clock Synchronization Perfectly synchronizing clocks across distributed nodes is unsolvable due to variable network delays and relativity effects (though the latter is negligible in practice). Algorithms like NTP or Lamport's logical clocks provide approximations, but absolute precision is unattainable—there's always some drift or uncertainty.
   These problems are "unsolvable" in the sense that no algorithm can guarantee a solution in all cases under the given constraints. However, practical systems often work around them by relaxing requirements (e.g., assuming partial synchrony, tolerating inconsistency, or using probabilistic methods). For instance, Paxos and Raft sidestep FLP by assuming eventual message delivery, while Bitcoin uses proof-of-work to handle Byzantine faults probabilistically."

### 1.8.2 Funny, some of them are related:

- consensus is impossible, it was proved by Bernadette Charron-Bost forever ago
- group membership service is impossible because it is a consensus: all the processes must agree on who is in each group

- failure detection is impossible because it is a group membership problem: the processes must agree on who is dead and who is alive
  There were also detailed responses from Ken Birman which we can discuss.

## 1.9 Anonymous 1

I realize it's far too late to make any changes to this document before tomorrow morning. But I am not seeing a requirements document at the requirements level of abstraction, I am seeing a proposed/example definition of a novel link layer, without enough information to infer how the rest of what link layers traditionally do would be implemented. And my still-cloudy head doesn't yet grok what I am reading in the document.

My abstract thinking keeps coming back to the choices IBM/Ancor made in "Class 1" Fibre Channel (the selector channel, which never got market traction) 35ish years ago. Yes, I know, my employer was on the "Class 3" side of that bus war, and we won, and IBM lost, and Kumar made 3/4 of a billion dollars by selling his Brocade founder shares at the peak.

But as we look at the atomicity problem we're trying to solve (immediately knowing if a packet/message was received at its destination), not just for adjacent nodes but for any pairwise combination in a 2-D mesh of only somewhat bounded size, I keep coming back to the idea of start a communication level transaction (what Fibre Channel calls a SCSI Exchange) by reserving a path all the way through the mesh between the two parties, with the intermediate nodes functioning as pass transistors (actually, retimers) in the style of what Ethernet calls Layer 1 switches; doing the series of data movements, acknowledgements, etc between the two endpoints as if they were directly connected, and then tearing down the connection when the communication level transaction (the SCSI Exchange) is finished. Obviously we'd need setup and teardown at nanoseconds speed, not microseconds or worse.

Note that the real difference between "class 1" and "class 3" is that "class 1" is dedicated/connected, while "class 3" has all of the DReDDful behaviors of switches. "Class 3" still has to populate tables at least at both ends (for my employer's ASIC 30 years ago this was entries in the SCSI Exchange State Table) on a per communication level transaction basis; it's just the forwarding tables in the switches whose entries are longer lived.

If we were to do this, there would be two "bits on the wire" packet formats, one used in reserving the path, and a second used within the path once established. The protocol outlined in this document would be used in the second. My guess is the second would need a subchannel like handful of bits to identify which of (say) 16 paths from or through or to a node to send this on, which is remapped by a tiny data structure at each hop.

Separately

I am steadfast in my belief that each node maintaining a data structure describing the next-hop for each destination, and therefore gathering data only from its immediate neighbors, is an O(N) algorithm in each node for discovering and maintaining the forwarding tables (as implemented in eBGP, for example), while the "probe" concept is an $O(N^2)$ algorithm which will collapse as it scales past N=100, just as "spanning tree" did, and just as OSPF and IS-IS do. (Note that I am not an expert in this area, and that my characterization of $O(N)$ and $O(N^2)$are gross oversimplifications.)

## 1.10 Anonymous 2

## 1.11 All Mechanisms and Algorithms will be Self-Stabilizing

Shlomi Dolev presentation [02025-APR-09 Video Slides]

Self-Stabilizing Definition: [Wikipedia].

## 1.12 Anonymous 3

Maybe I am missing the context of the conversation, but I think you need to lead with the behavior you want to see, and from what perspective. it will anchor the piece into a root outcome.

Since it is really turtles all the way down, each individual reader perspective often only sees the turtle below them and above them. I found myself bouncing between application developer, compute designer, network operator, network hardware designer, and network standards perspectives...and I know my perspective is limited.

I'm fond of the idea, but without that grounding it feels a bit like you're trying to boil the ocean.

Here are my inline notes:

Thoughts on the introduction:

- Atomic acceptance at all nodes in a network will require a shift in the hardware model. Some form of separate item index, per interface, that would allow for signaling when a fan-in problem occurs and a buffer is overrun, the payload is lost but the header remains?

- OSI abstraction was to limit information required across the layers, sure it can be rewritten but the scale of information leakage is key, more on this later.

Efficiency of SAW:

- I'm confused by your clustered ACKs proposal. That's how windowing works.

- Your snakes proposal looks a lot like RSVP for the datacenter. This is being worked on within Broadcom for Jericho3/Ramon3. I'd love to see it outside of the hardware for network, but I'm not sure it could be fast enough. I'd also love to see it in a public standard rather than proprietary firmware. They aren't round-tripping the payload, but they are per-defining a path for a session and TDMing the links to guarantee the bandwidth.

- Assuming you are immediately reversing the data flow of the snake, prior to processing, your ACK will arrive at 'almost' the same time as a traditional ACK - if you opened your traditional window size to the size of the snake. Huge windows are pretty common, as are scaling windows tracking reliability. This is why you often see a sawtooth pattern in throughput. The difference would be waiting for the NIC to process and check the checksums on the traditional ACK. It could give you a more reliable ACK, but puts the onus of checking the payload on the original sender.

- Does the receiver proceed assuming good data or does it wait for a three-way handshake? No, you'd still need some form of FEC or CRC on the snake to be able to NACK if it doesn't match. Otherwise the failure scenario is rather painful.

- It might also push the operation ordering problem higher into the application, requiring it to know the order of atomic work asked for and ACK'd.

- Interface logic hardware scaling will be key, just as it is today. You'll need to scale what the interface logic can handle to the total possible dangling snakes, which leads you to a hard timeout on transmission based on hardware limits if nothing else.

Best Effort is not enough:

- "A full-duplex link is a logical grouping of two simplex links that are independent failure domains". This has been true for more than 20 years. In fact, they're no longer 2 simplex links but some arbitrary number of simplex links, often 4 each direction. Interface logic is making them appear as one to you. After the NIC packetizes it, the

interface logic slices it up even further for sending. Again, all abstracted away and bounded. (Turtles all the way down.)

- You're talking about black and white failures, but those aren't the problem. Interface detection is pretty good. The grey failures are what kill network behavior....that and badly behaving apps. Parts degrade over time, someone stretches a fiber and cracks it just a bit, dust works its way into the interface, etc. Networks watch for light level changes and error rates, but not as well as they should. Every cycle not spent on sending traffic is latency.

- ACK/NACK, are you talking about it on a per-link basis or end-to-end? If it is per link then the input buffers just got a LOT bigger. Also, how do we account for the frame dropped from overrunning the buffer? Also, deep buffers mean latency. The first question in network hardware is 'drop or delay, which do you pick in contention?'

- "But in practice, these networks routinely violate packet causality through multi-path routing (ECMP), queuing disciplines (EQDS), and load-balancing heuristics that scatter IP fragments across the network." You're going to need to back this up. Session-based load balancing has been the norm for 20 years, specifically to solve the ordering problem. To speed things up folks are starting to look at per-pack load balancing again, but it hasn't been common since the 90s.

—

Overall, it feels like a solid half of a proposal. I'm coming at it from a, "What network hardware do I design to be able to do this?" I know that's further down the line than you are yet, but there are some very fundamental gaps in the 'what', before i start thinking about the 'how'. I think anchoring this, as I mentioned at the top, will help focus that conversation and aid sharpening the 'needs' portion of this paper.

Pushing back the other direction, let me rehash an idea Mae posed:

90+% of this could be solved with a separate E2E timing estimation as a hard time-out. Either declared based on architecture or as a side-channel application like MSFT's pingmesh, define a hard timeout that can be leveraged by the applications. "I have not received ACK within TIMEOUT, therefore resend." If all 'snakes' have a UUID, duplication isn't an issue. A rolling hash where entries expire after an hour should easily cover that very managably.

Now let's talk information leakage (necessary information sharing?). Hearkening back to the line, "All models are wrong, but some models are useful." The problem with pointing to the black box of infrastructure and citing reliability failures at the higher levels is that it is failing to discuss the scaling problems within the black box. Information flow up from the black box to the application is reasonable because it is a bounded set of data. Application knowledge flowing into the black box is an unbounded set. Network latency, cost, and behavior is directly tied to the bounding of that set., otherwise no amount of hardware will accommodate the fan-in.

End of the day, speaking across perspectives will be key to driving this. Most everyone reading will only react to it from their own perspective, and they can be quite different. When I need to demonstrate the difference, I explain it like this:"Developers are artists, they are sculpting in code and in math. Infrastructure, on the other hand, is building a bridge out of bent and cracked tinker toys. Nothing is to spec, the lengths aren't exact, the holes aren't drilled at exact angles, but they still need to be able to move the Tonka trucks over them." That usually starts a larger conversation, but it is a good statement to anchor from.

## 1.13 [Anonymous 4]

### 1.13.1 QUESTION: What's different about Atomic Ethernet?

### 1.13.2 ANSWER:

There are some significant conceptual differences between the Atomic Ethernet `Link`[1] Protocol (AE) and traditional networking protocols in today's distributed microservices, key-value stores, and databases. For example:

- The protocol is *not* an asymmetric 'source-destination' protocol. But a bipartite 'token coherence protocol' – the ability of one side, to 'know' whether or not the other side has consumed (i.e., observed) the token. It is *not* a source-destination protocol as in traditional Ethernet. Local Addressing is by port and compass point scout packets. Global Addressing is rootward or leafward on trees.
- Messages are not 'resent'. If a failure occurs for any reason during the 'transaction' then the transaction is first reversed out ('unsent') and then re-applied[2].
- This is essential; any detritus[3] from the previously attempted transactions must be completely eliminated, such that, the next time the transaction is attempted, is 'the first time'.

This is in-contrast to the conventional 'forward-in-time only' protocols, whose effect is transformation from a previous state to an irreversible future state. Genuine (logical or physical) reversibility provides simpler and more guaranteed capabilities for error recovery that are not available to forward-in-time only protocols.

The new perspective, is that instead of the source (initiator) executing a state change in a forward direction in time, both parties are responsible for the successive forward, *and* the successive reversal of the atomic operation when an error occurs; not just one of them, with perhaps an incomplete knowledge of what state changes may have occurred in the target. Logical reversibility can be viewed as *time reversal*, only if (1) it is successive, and (2) all trace of the transaction has been removed with no evidence whatsoever that it occurred[4].

A central issue in the E2E debate appears to be whether or not the network is *lossless*; and the performance penalties for recovering from those packet losses. We summarize our position on packet loss thus[5]:

> In an event-driven system we convert a packet loss to a `Link` failure. The only failure we have to deal with is `Link` failure; which is recovered by another `Link`.

I.e. We treat packet loss (or corruption), not as an exponential complexity of failure recovery mechanisms, but as an opportunity to simply 'reverse time' so that the error never occurred. This new perspective is central to our ability to achieve exactly-once semantics, fast failover, and security guarantees.

[1] We distinguish between a conventional 'link', and the AE `Link`, with capitalization and a different type face.

[2] If we allow message retry, there are classes of faults that only require one round trip (we don't allow message retry). If we allow packet loss, we would need a message retry in the 2-phase protocol. If there are no faults at all, one round trip suffices.

[3] Even with sequential programs, once we have side-effects, we have many more opportunities to confuse ourselves, and this can get much worse with concurrency.

[4] This requires that the internals of the transaction must also be hidden from any kind of observation. until both sides have reached some system level threshold of knowledge needed for the promises expected.

[5] The Bill Jackson formulation.

## 1.14 Market for a Sea of XPU's are the *closed* hyperscalars

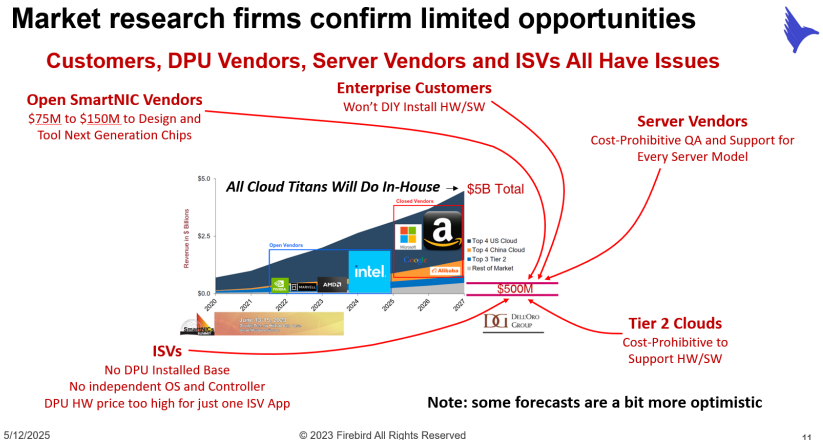


Figure 1.1: The SmartNIC Revolution died, long live the XPU Revolution. Courtesy: Harry V. Quackenboss

The FPGA NICs aren't cheap compared to a standard NIC. The only ones our use cases would work with were RISC-based Linux ones with, if not currently, at least planned, per-core (single-thread) performance close to single-thread performance of a Xeon core. Broadcom's Stingray II (shelved), Marvell's high-end Arm processor (never produced in volume), and Intel's Mt. Evans roadmap (never pursued for anybody other than hyperscalers). NVIDIA's BlueField 4 (not yet released – maybe shelved) looked good on paper.

The really high-volume prices of the SmartNICs in the BlueField-3 class were probably as low as $700 per unit, which I suspect is still less way than FPGA NICs.

For our use cases, price/performance was better on these types of SmartNICs because the data in flight nature of the workloads meant less DRAM/core, and the RISC versus X86 advantage of 70% in MIPS/Watt meant it made sense to offload those. However, if the SmartNICs couldn't keep up with he geometry shrink from 3nm -> 2nm and so on that the CPUs were on, the per-core performance and MIPS/Watt advantage evaporated.

———— Harry Quackenboss

Are these honking big, expensive smart NICs you're talking about? I think we can implement our protocol in a pretty simple FPGA. (It can be really simple if we do failure recovery with the CPU.) In fact, there was an FPGA implementation. John Lockwood may know the details.

Years ago we spoke to a NIC vendor. We could implement 99% of the protocol with small changes to their standard product. There was one corner case that required a change to their pico-code that was the showstopper for them. None of those changes would have been visible to customers not using our protocol.

———— Alan Karp

I'd say the future is highly interconnected processors that have specialized computing support, SmartNICs fall in the specialized computing category if they have engines for things like encryption and compression, or something general purpose like an FPGA.

———— Kev.

Thanks for this perspective. I had thought that smart NICs everywhere was the wave of the future.

That being said, I'm not sure we need smart NICs for the basic protocol. I believe that the happy path can be handled by simple state machine logic in the NIC. Recovery from a broken link or node is probably too complex for that, so it would have to be handled by the CPU. Recovery would certainly take longer, perhaps ms instead of microseconds.

———— Alan Karp

It strikes me that among Kevin, Alan, and Hesham, there is no common understanding of what the minimum NIC feature set is that this future protocol suite would run on (or, for that matter, possible middle boxes such as L2 switches)

Adrian is current on this stuff, and I am not, but my understanding is that SmartNICs, to summarize, are used by AWS, Google, Microsoft, and Meta (and as a first approximation, Meta is less than 1/5 the size of the others), but each vendor controls the SW stack including host drivers, and firmware on the SmartNIC, and even at least some HW features. And, at least for now, none of the public cloud subset is making programmability available to customers.

I don't see how this effort is going to impact that, but maybe Adrian has a different opinion.

Other than niches (and compared to markets big enough to support new silicon at the rate CPUs evolve, even FPGA NICs are in my taxonomy, niches). NVIDIA's adaption of their DPU family to allow tuning for RDMA in AI and HPC clusters (that NVIDIA calls "SuperNIC") isn't really an exception in my view. There isn't a significant installed base. If you scrutinize the forecasts and separate out vendors obfuscating to situation by declaring their high-end server NICs as SmartNICs, none are forecast big growth in the future (again, excluding hyperscalers).

A conclusion I draw from that is that trying to create new broad industry standards is inconsistent with depending on hardware roadmaps for NIC features that aren't forecast to be big volumes (excluding the hyperscalers that are going to be difficult to influence.

Translation: if this effort depends on custom programmable NICs (SmartNICs) isn't going to lead to broad interest.

The sooner this group can come to a common understanding about whether SmartNICs are in or out of scope, the sooner the debate can be narrowed.

I personally would love to see SmartNICs be widely adopted. As Kevin intimates, there are a bunch of things one can do.

I just think unless and until factors beyond this group's ability to influence change, it's like trying to start a campfire with damp kindling in the rain. Smoke, with effort. But not much fire.

———— Harry

## 1.15 Our Unfair Advantage in Networking

Conventional Clos Networks assume a "flat" communication model, and an Any to Any (A2A) traffic pattern. What if we could take advantage of spatial (neighborhood) and temporal (synchronization) traffic patterns and provide ultra low latency in directly connected (switchless) networks?

### 1.15.1 ML/AI Inference on The Edge

While training is for the big boys, and often needs gigantic capital and operating budgets, inference can be carried out with more modest budgets, on far smaller networks, fewer GPUs and CPUs and is highly constrained by latency requirements of customers whose attention span has been trained by expectations of search engine response times.

These latency characteristics are also naturally serviced from the edge where the customers are, rather than the big centralized datacenters owned by the superscalars.

This provides many more market insertion opportunities for DÆDÆLUS' unique insights and unfair advantage as a software infrastructure company bringing new capabilities and value to the edge.

### 1.15.2 Local not Global Traffic Patterns

Datacenter Traffic Matrices are sparse and skewed, exhibit locality and are bursty. [Griner SIGMETRICS 2020]. They are far from arbitrary or random, which is often used as an justification for a "Flat" networks model in datacenters.

Reconfigurable networks [Dynamic Trees] allow to establish direct links between two frequently communicating pairs of racks in a datacenter, e.g., using digital micromirror devices.

### 1.15.3 Snapshots need Reliable Subtransactions

Both Transactions (MVCC) and AI/ML rely on snapshots being consistent. But they can't be made consisted using timestamps alone.

### 1.15.4 Spanning Trees under application control

### 1.15.5 Failure, isolation, healing: using local information only

### 1.15.6 Topology Awareness for Applications like AI/ML

### 1.15.7 Sequencing and clocks without timestamps

Timestamps are an unsafe ordering construct.

### 1.15.8 Our unfair advantage

- Eliminate timeouts & retries, which cause cascade failures
- Switches and routers not needed
- SmartNICs or Chiplet microdatacenters
- Easy to use (compared to conventional networks)
- Links carry only functional (not transit) packets
- More potential Bandwidth at the same cost
- Ultra Low Latency is lowest possible: governed only by the length of the connections
- Connections - Ours are are shorter. (2cm vs 2m vs 20m)
- Encryption not needed between adjacent nodes (Saves computation, time, and energy)
- Developers control their own application topologies
- No bandwidth amplification, either by Gossip protocols or timeouts and retries

### 1.15.9 Opportunities we Unleash

HammingMesh is Optimized specifically for deep learning workloads and their communication patterns.

> "Deep learning training uses three dimensional communication patterns and rarely needs global bandwidth. It supports extreme local bandwidth while controlling the cost of global bandwidth."

HammingMesh uses a 4x4 Tile, connected in a valency-4 (square) mesh, and overlays a Hamming code to provide a GEV coordinate system for routing. As with many coordinate systems, including the cartesian coordinate systems used in most Torus networks, they become extremely complex when they fail and we have to route around failed links and nodes. This is not the case for DÆDÆLUS, where every node has an coordinate system with itself at the root. Because

The Strawman objection, by conventional network architects, uses "diameter" as a figure of merit (or demerit). Includes three misconceptions:

- Nodes are "fixed" in some Cartesian layout, nailed to the floor of the datacenter in some specific rack (our addressable entities are mobile)
- Hops are somehow related to latency in a fixed cartesian coordinate system, so the diameter gets worse as you scale (DÆDÆLUS exploits neighborhood locality)
- Execution entities are randomly placed throughout the datacenter — for a "flat network concept".
- Most AI/ML Traffic patterns are highly predictable. Once the pattern is established in the first few passes, the communication patterns can be exploited and the virtual trees can be optimized to match those patterns.