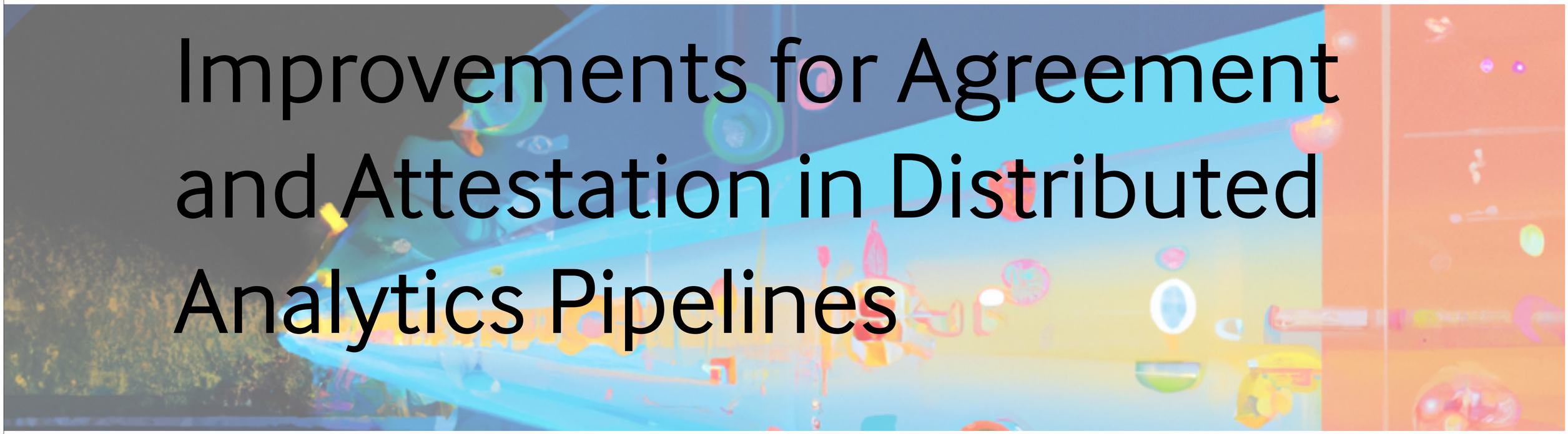
CLSAC 2023

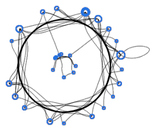


# Improvements for Agreement and Attestation in Distributed Analytics Pipelines

20231031

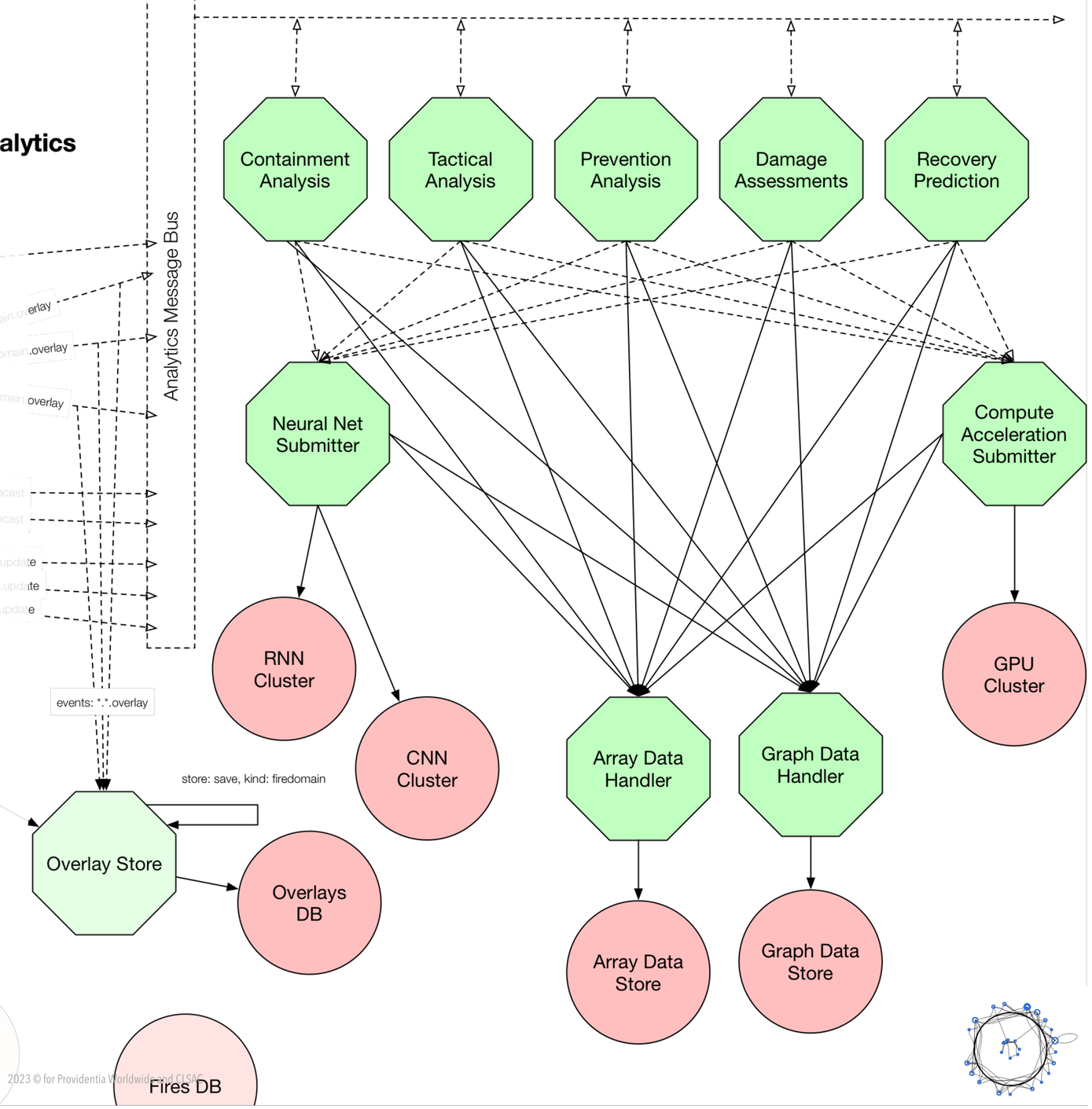
S. Ryan Quick, Providentia Worldwide

@ph43d0



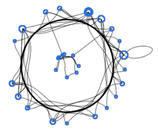
# Consensus: Critical for Analytics/Distributed Apps (like pipelines)

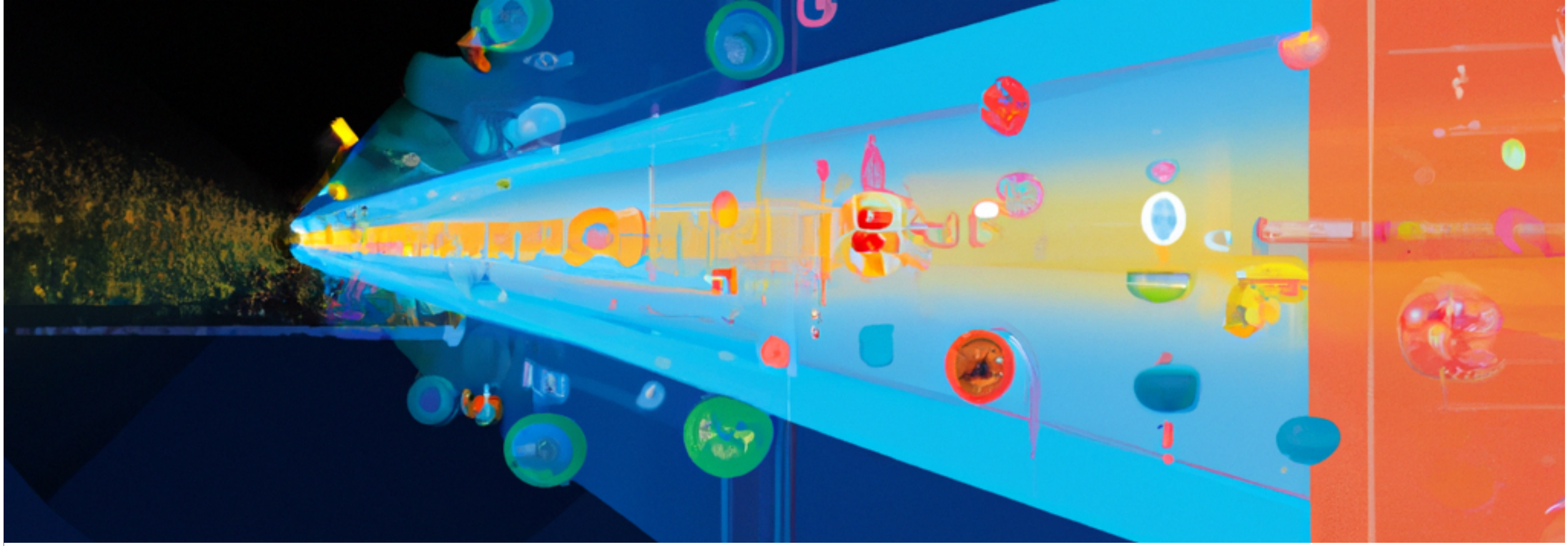
- Modern analytics pipelines rely not only on data movement, but on myriad supporting applications and services working in concert — this is true for offline and wire analytics.
- While feature development outpaces hardware innovation, generational improvements come when we co-design the application and its operating environment. (GPUs and AI, web3; Storage Advances for “BigData”; CPU for cryptography, isolation; etc)
- Agreement is not just crucial, but provides an excellent gatekeeper for safety and resiliency for the entire system — merits our attention independent of its supporting role.



# But consensus is also data dependent and needs a (lot) of explanation...

- Example scenario and problems therein
- Places where distributed applications operate, and where agreement between those components occur
- Short exposition on consensus and its characteristics (good and bad)
- Network characteristics which might influence the consensus behavior
- How did we do in our first foray?
- How could we use this in our sample scenario...

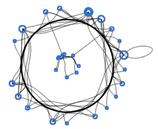




# Example Use Case/Scenario

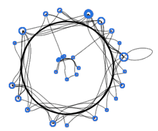
# Example Use Case: “AI Show-your-work”

- AI adoption at scale requires not just innovation decisions, but a method by which the system can show its decision-making across the wide network of the device concept and drill down to the individual device.
- This stream of events is related to but distinct from the actual work of the AI system. By making use of the combination of decisions, inputs, outputs, and aggregated results, then insight about the AI platform comes to light.
- Adding trustworthiness to insight requires bringing technologies which can provide provenance and security to the insight stream – these include cryptography, DLT/ blockchain, and the like.
- Creating a system of this sort is not simply a “logging/tracing interface” but generally a streamed complex event processing platform.



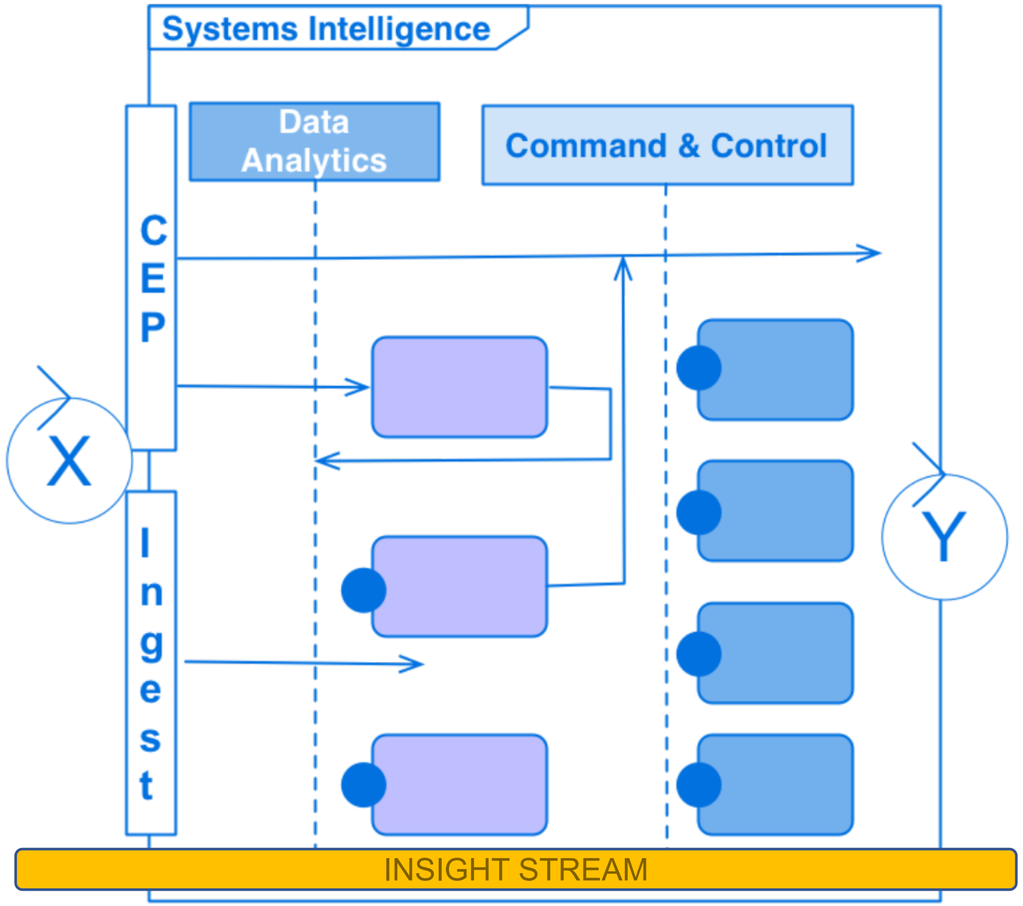
# Example Solution: Systems Insight and Intelligence

- Provide a means to capture, decide, and route high volume data reliably with channels to control, analytics/AI, and self-healing/self-actualization
- Retains data context, specificity, discretion
- Provide streaming capabilities for
  - Analysis
  - Data enrichment
  - Process, environment, and methodology telemetry/tracing
  - Internal understanding
- Scales to Millions+ events/sec
- Multi-protocol ingress/egress without impacting data sources
- Reactive and proactive real-time situation handling, command and control



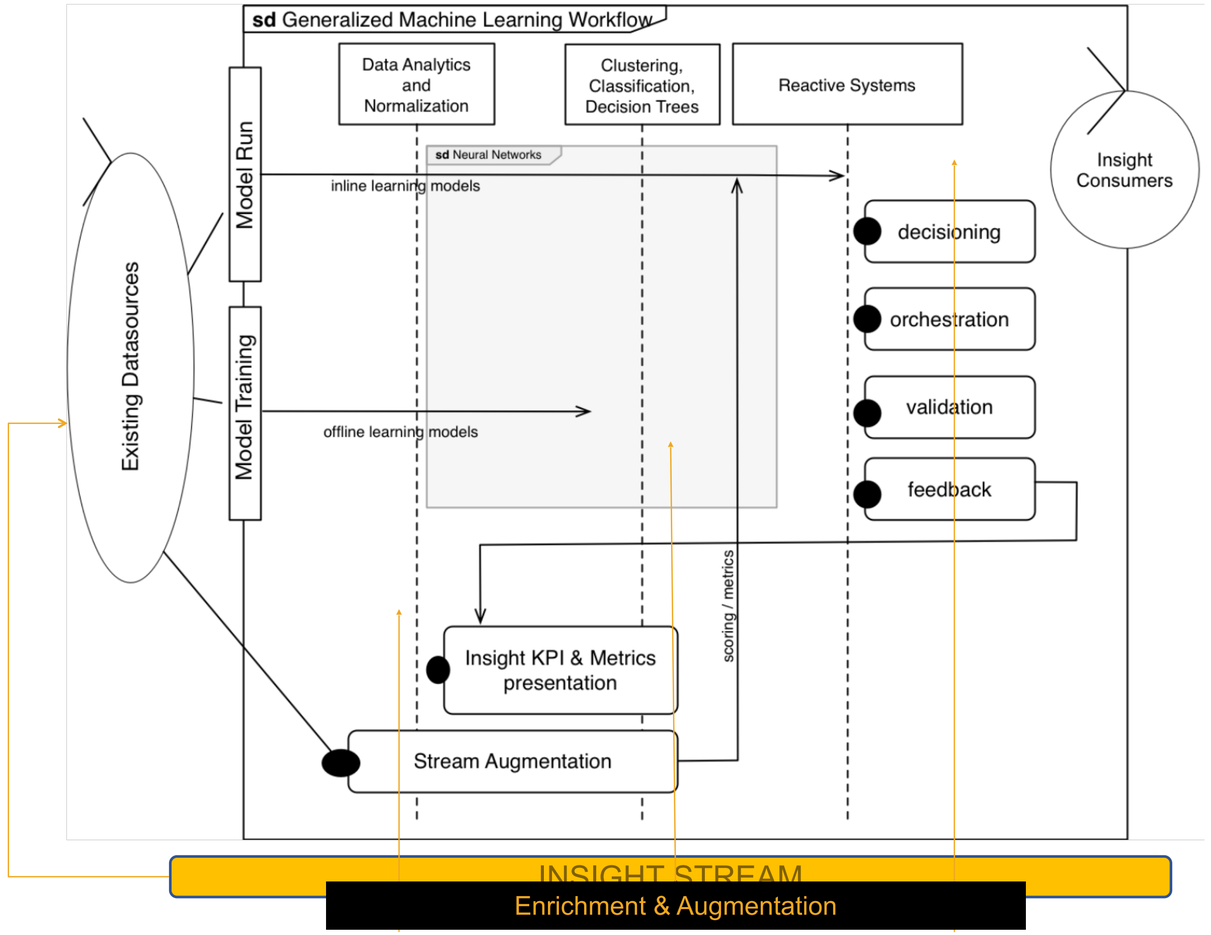
# SI: Real-time Analytics and Data Gathering

- Common ingest/publish protocols across domains
- Streaming for any combination of real-time, online, and offline analytics
- Event-driven system for command and control
- Middleware permits edge, datacenter, regional, global implementations
- Insight augments the behavior of the SI system itself
- Builds on proven enterprise messaging technologies



# SI: Generalized Learning/AI Methodology

- Leverage same platform for learning as data/ telemetry collection
- Supports inline/real-time training & learning
- Supports offline data repositories
- Event-driven platform for immediate behavior changes
- Utilize insight platform for both additional data and enriching AI data/models



# SI Methodology, not "Software"

Inter-operation through common

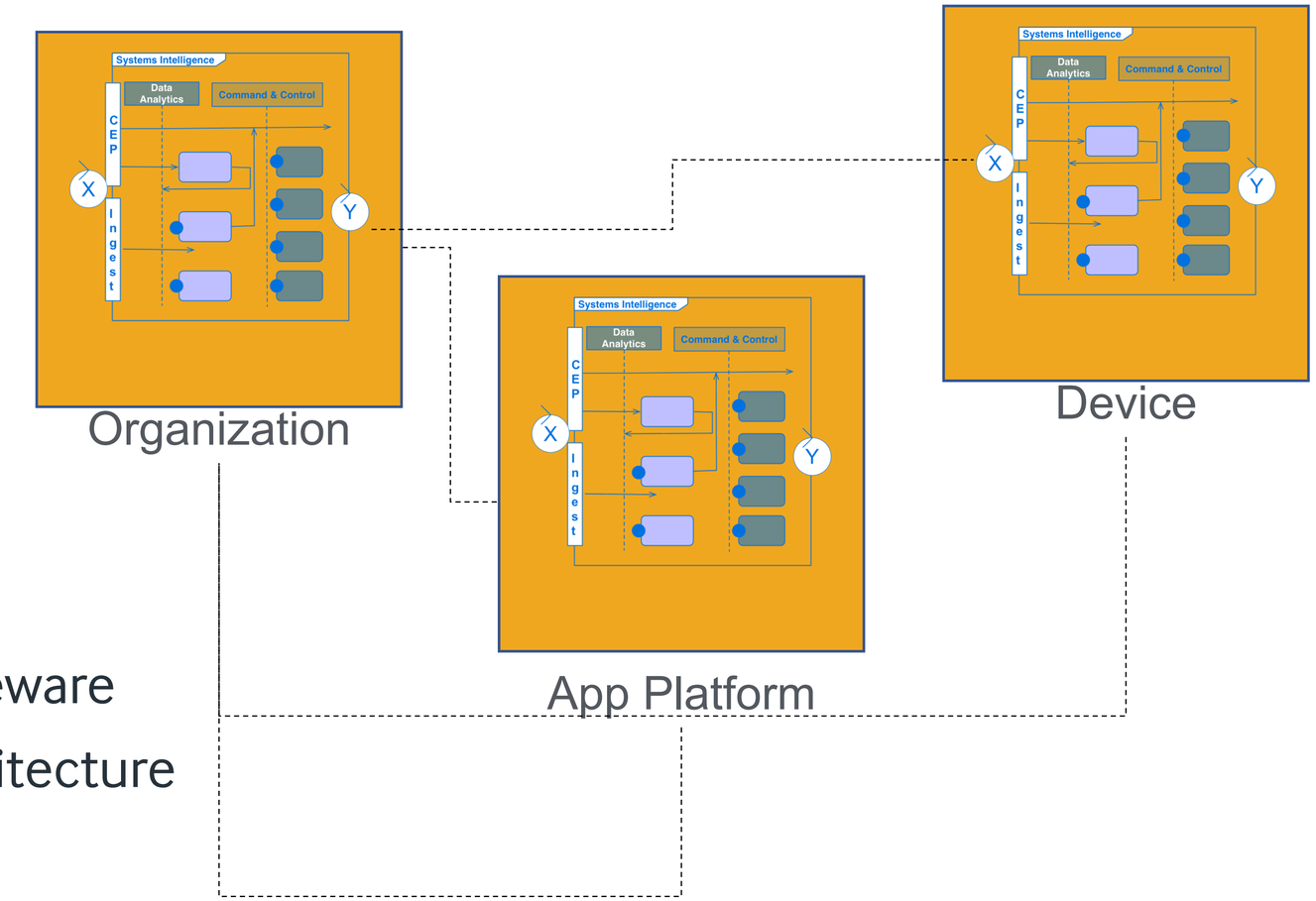
- Protocols
- Interfaces
- Functions
- Event Handler

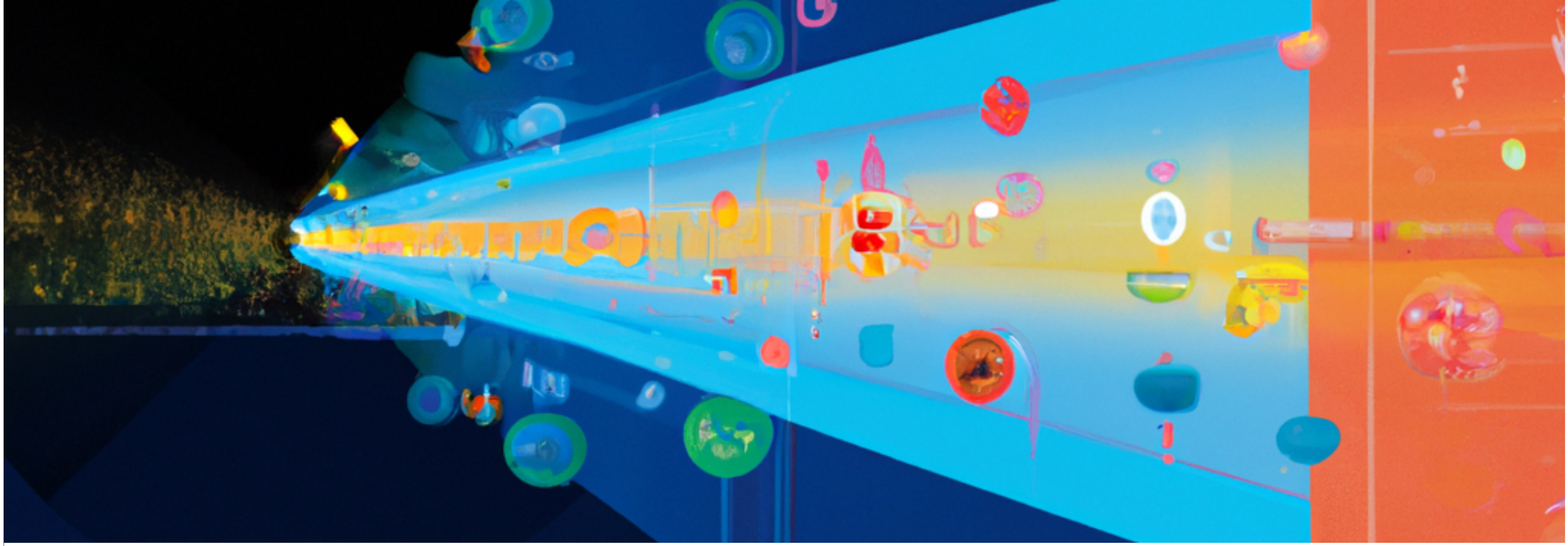
Deployed at any level(s)

- Organization
- Platform
- Edge
- Device Management
- Device Specific

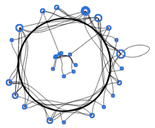
Components

- Messaging middleware
- Event-driven architecture
- Microservices
- Data handlers
- Telemetry ingress/egress





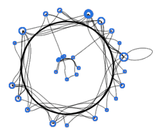
# On Consensus Algorithms In Distributed Systems



# The Good: Consensus adds resiliency, trust in faulty environments

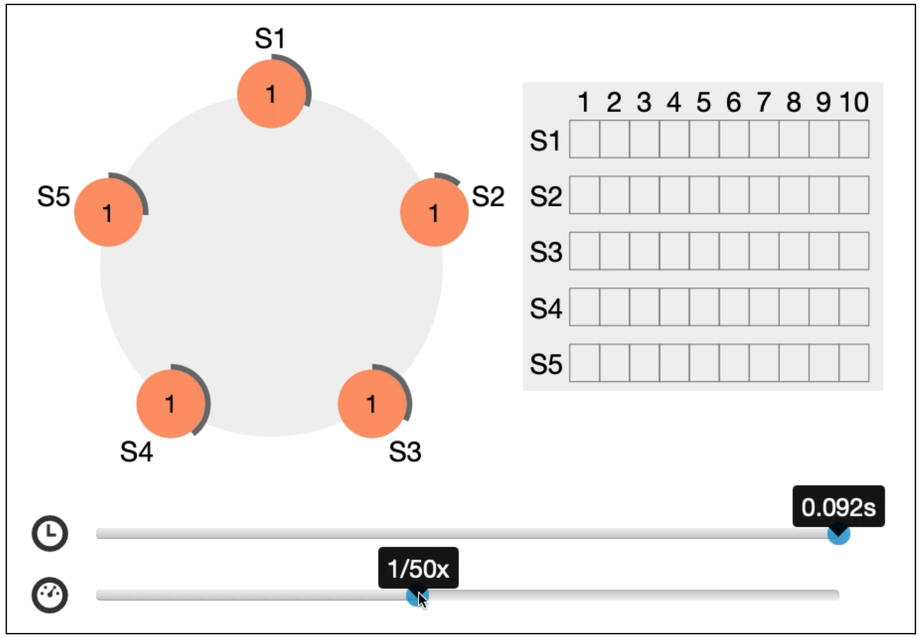
## GUARANTEES/ASSUMPTIONS

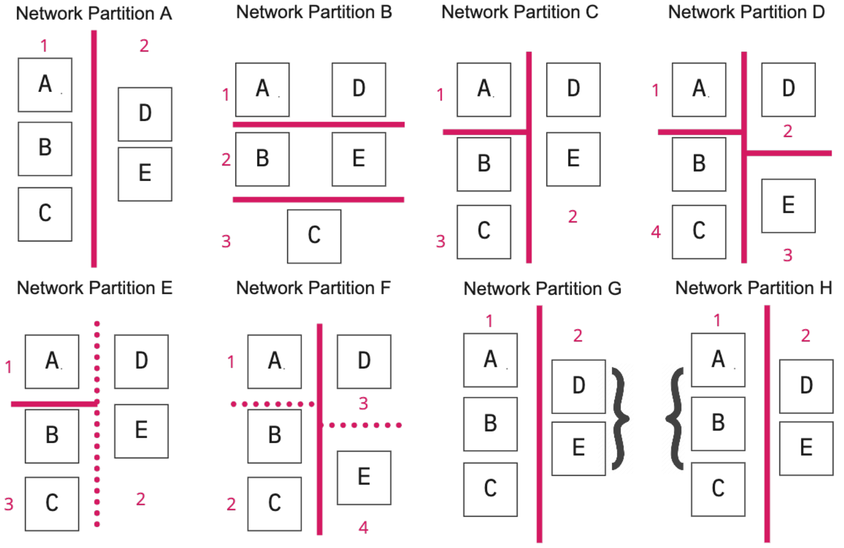
1. All participants (must) decide within finite time
2. All correct decisions must agree
3. Correct inputs must always derive the same output
4. Correct decisions imply correct proposals
5. Common message delivery latencies differ between participants
6. Heterogenous participants will derive correct decisions at different rates
7. Practical communication is a source of both crash and Byzantine faults



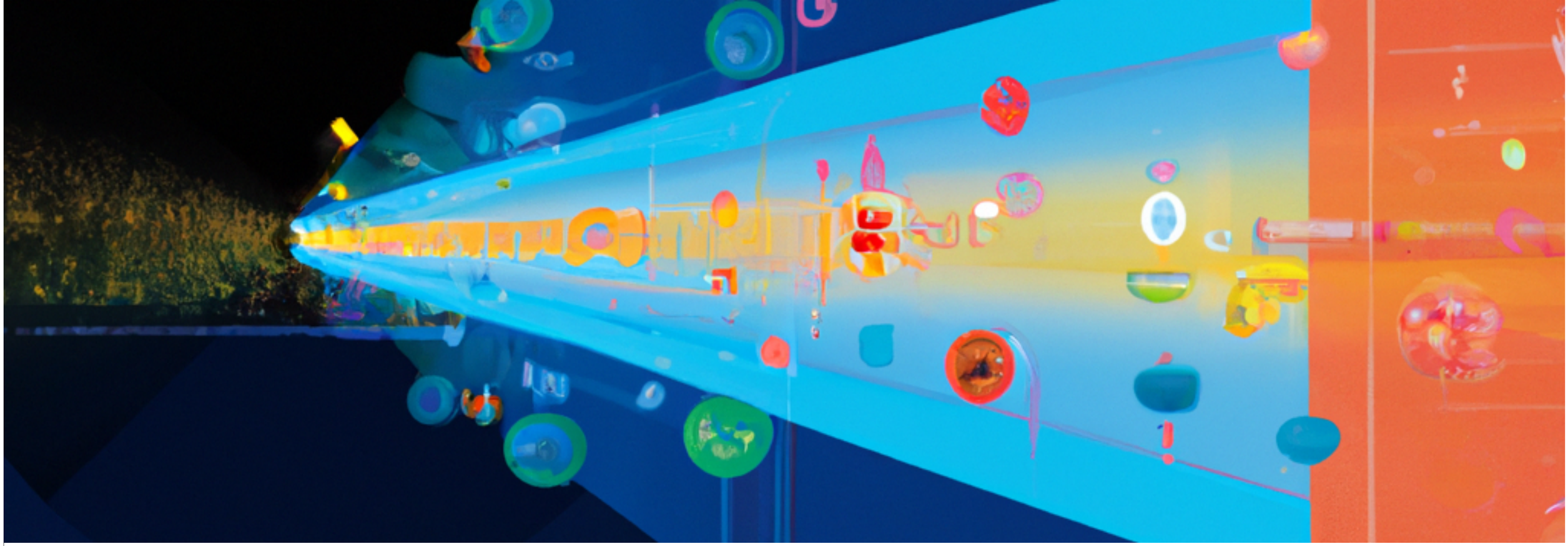
# The Bad: relies on the network for its functionality

Try out the demo at https://raft.github.io

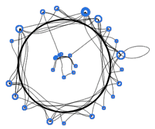



- Chatty — many messages for a single "update"
- Requires the most safety in times where it is in a known-degraded state
- Assumes asynchronous delivery — which requires even more messages and creates Byzantine fault scenarios
- Network partitions can delay data progress indefinitely

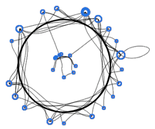


# Raft, Meet Data Vortex

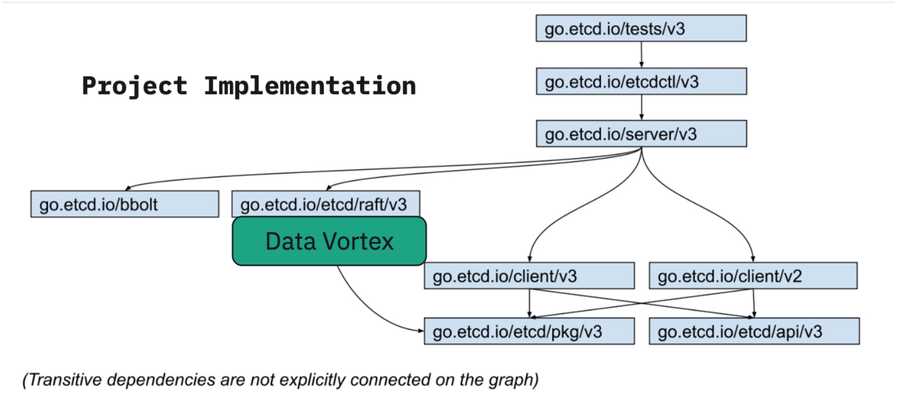


# RAFT Consensus Over Data Vortex

- 2022 White paper (Scan code 🙂)
- Performance is always “sought first”, but we looked into the functionality and needs of the algorithm, not primarily acceleration.
- Address key consensus problems to remove the need for RAFT to manage them itself:
  - Network partitioning
  - Message transmission, participant, and verification unreliability
  - Asynchronous message handling and processing; heterogenous participant behavior

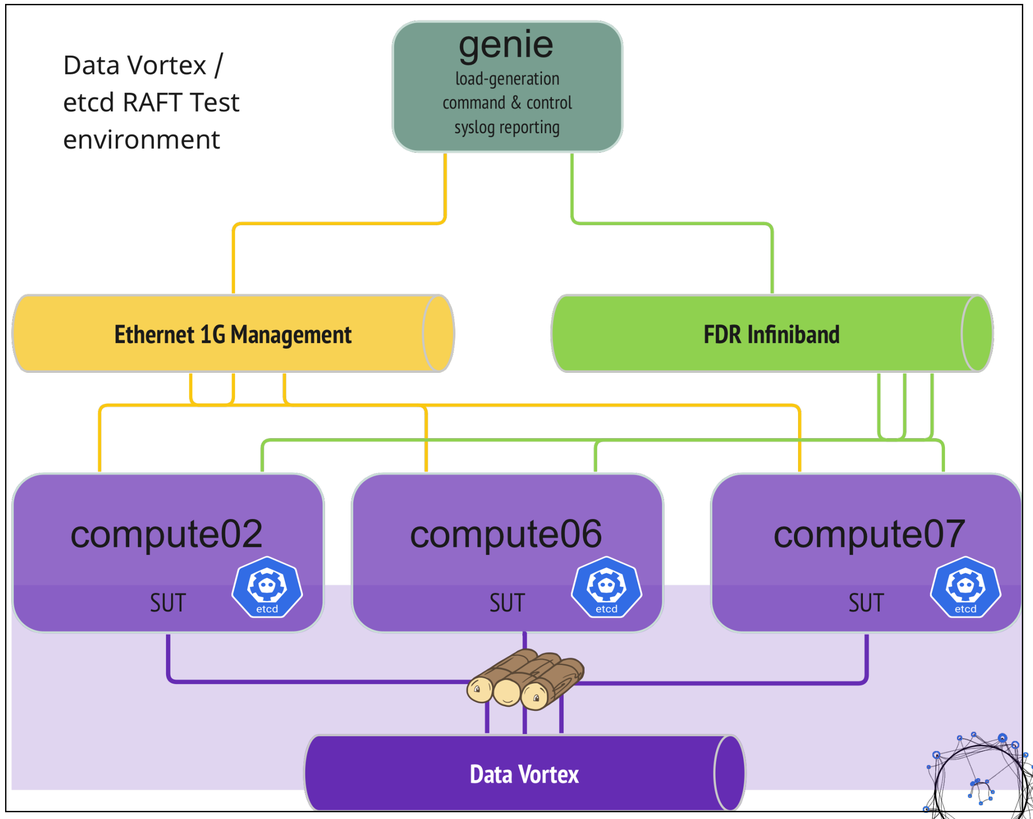



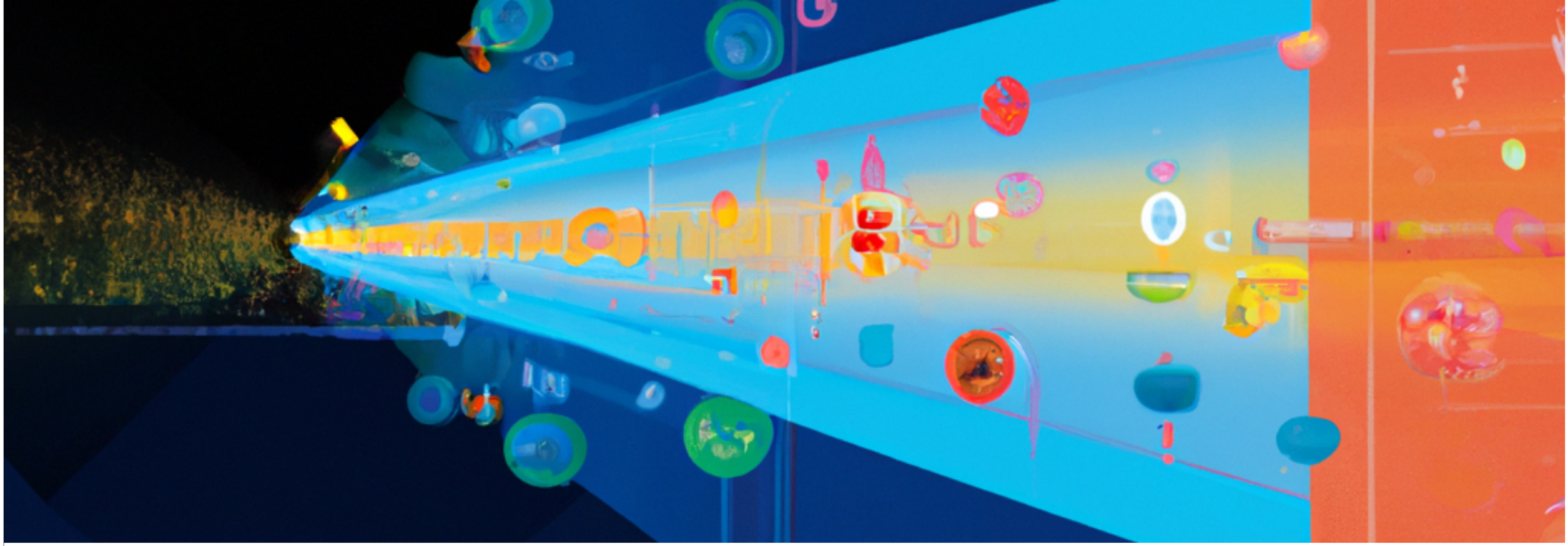
# Mapping RAFT to DV




*(Transitive dependencies are not explicitly connected on the graph)*

- Replace the network transport for RAFT with DV. Modular interface as shown here.
  - cGO interface — allowed us to use the native C DV API with etc RAFT (Go)
- DV provides: message creation (positive ack); parallel transport with synchronized delivery for all recipients; ordered messages with global barrier sync.

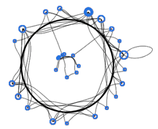




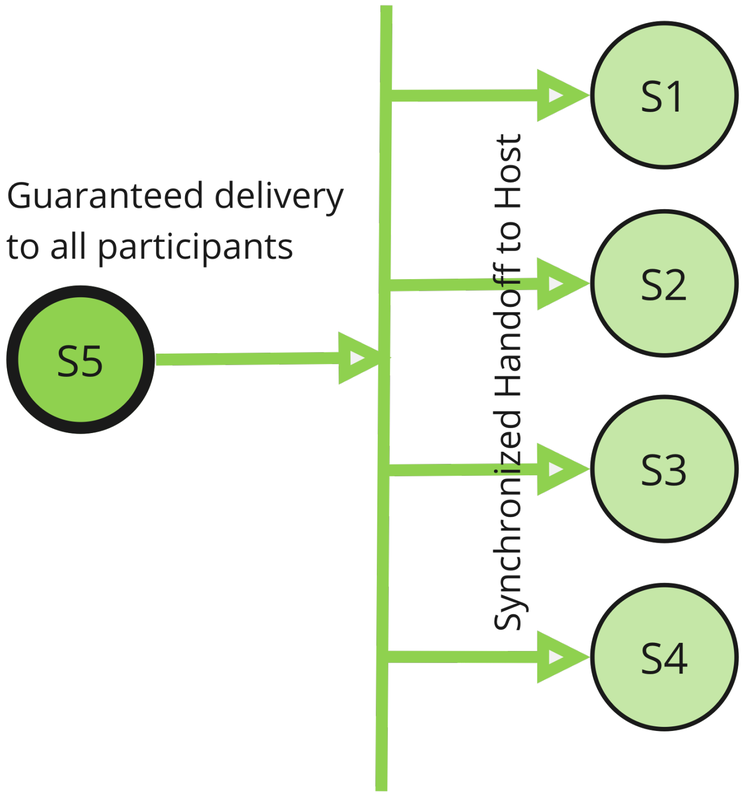
# How Did We Do? Next Steps, Other Ideas

# Optimizing the Network for Consensus

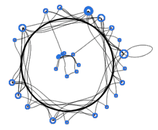
- Atomic data operations complexity reduced by over 50%
- Latency for data operations drops dramatically (with additional improvements)
- While Byzantine faults cannot all be prevented, Byzantine actors are immediately identified. Network does not allow sybil or imitation-based attack vectors.
- Elections can be limited to a maximum of 3 rounds (if failure in initiator). Currently the easiest defeat of RAFT is the “infinite elections” attack.
- Removes 4 of 7 challenges from the software algorithm



# DV Network for RAFT Solves 4/7

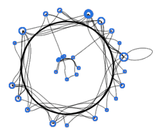



1. All participants must decide within finite time
2. All correct decisions must agree
3. Correct inputs must always derive the same output
4. Correct decisions imply correct proposals
5. Common message delivery latencies differ between participants
6. Heterogenous participants will derive correct decisions at different rates
7. Practical communication is a source of both crash and Byzantine faults



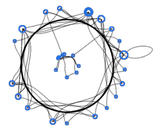
# Improving Analytics on the Wire

- For extremely complex pipelines/systems, provides unprecedented resiliency and trust without impacting performance
- Generally, tuning efforts work on optimizing network for performance within an application (GPU<>Memory; storage <> host, etc). We are looking at the entire distributed system composed of many applications.
- Example additional network topologies offering additional/alternate characteristics:
  - SyncE/IEEE1588v2/Time-Triggered Ethernet for ordered and synchronized timing/ frequency over Ethernet. These protocols allow for predictable latency, ordered messaging, and determined delivery
  - SHARP for data reduction and latency normalization over Infiniband. This creates predictable latency for large message transfers and command/control operations alike.
  - Serial RapidIO for “generally specialized” embedded and co-processor node to node communications where predictable latency, dynamic and rule-based routing, and frequency-guaranteed delivery are required.

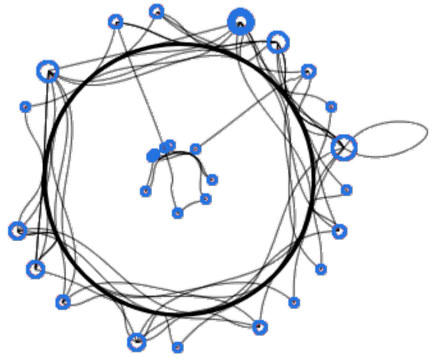


# “Generally Specialized” Design

- Distributed coordination, consensus specifically are difficult problems —— regardless of the data/use case
- For large scale problems and augmented, disparate analytics pipelines —— leverage all components to bring small improvements to safety, trust, performance, isolation, etc
- Our work with DV demonstrates the potential in looking at just one aspect of the coordination problem —— which only grows with larger and more diverse systems
- When trust, safety, resiliency are key —— do not require the most safety in degraded states —— design and engineer around these situations by using all aspects of the environment



# Providentia Worldwide



## Questions?


@phaedo

ryan.quick@providentiaworldwide.com

@ph43d0