Please ask questions about anything you don't understand

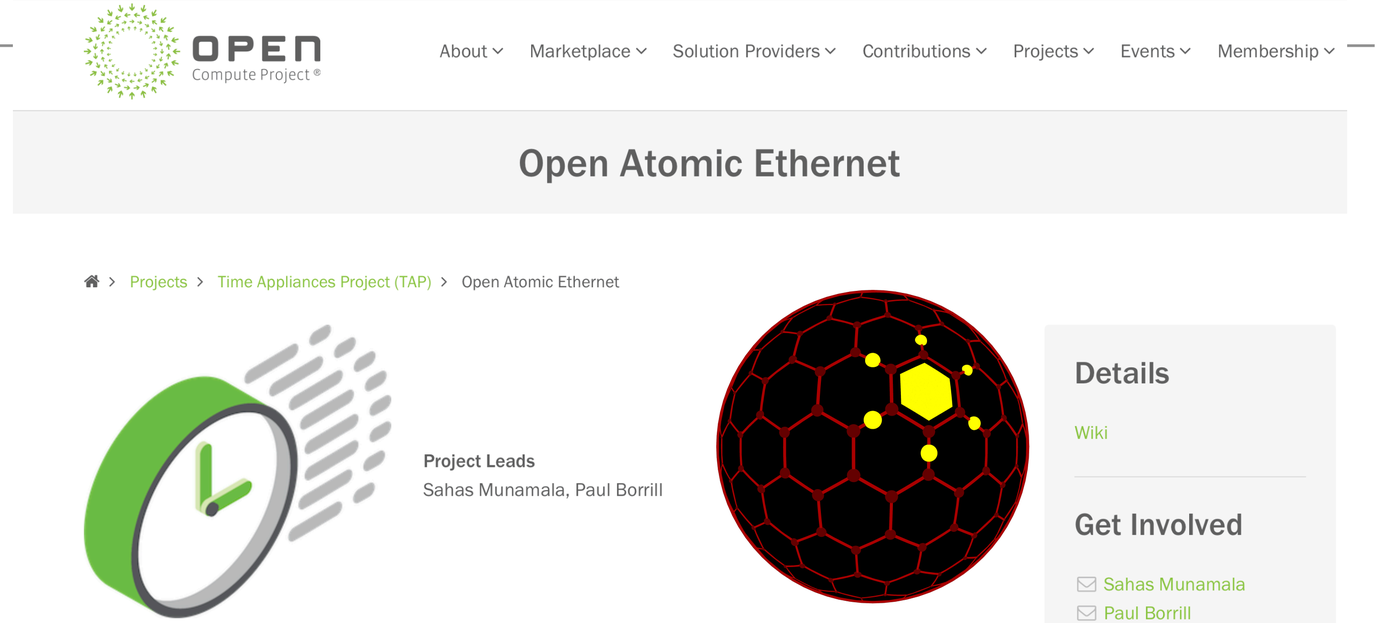
Introduction to DÆDÆLUS

Welcome Mathematica Programmer

V1.0 (02025-Feb-10)

Open Atomic Ethernet and WOLFRAM MATHEMATICA

We would like to promote the use of Mathematica for all the specifications for the new OAE Standards Project.
Starting with:
1. Metcalfe+Boggs Original 1976 Paper.
2. Briggs 1978 Paper that defines Exactly once semantics and the alternating bit protocol.
We now have over 40 members of this new initiative, with more coming in every day.
[click on image to open page and navigate the project] We meet *every* Wed at 9am PST.



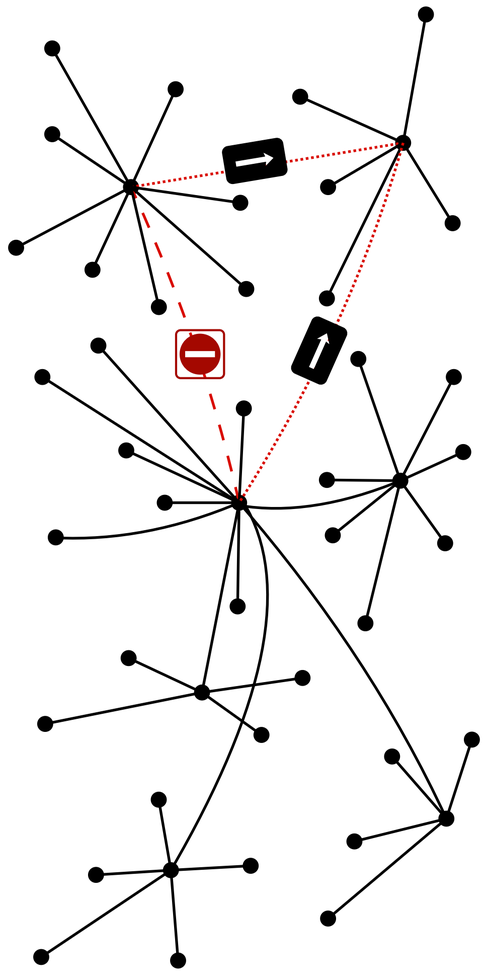
Problem

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 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

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



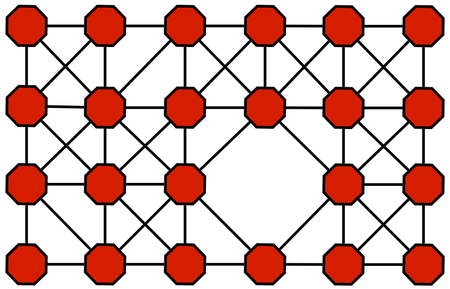
cassandra hazelcast redis ceph MESOS elastic mongoDB Apache Zookeeper kafka AEROSPIKE hadoop apache Ignite APACHE HBASE RethinkDB APACHE GEODE VOLTDB APACHE ActiveMQ riak MooseFS Infinispan chronos TERRACOTTA

An Analysis of Network-Partitioning Failures in Cloud Systems:
https://www.usenix.org/conference/osdi18/presentation/alquraan

Tail Latency

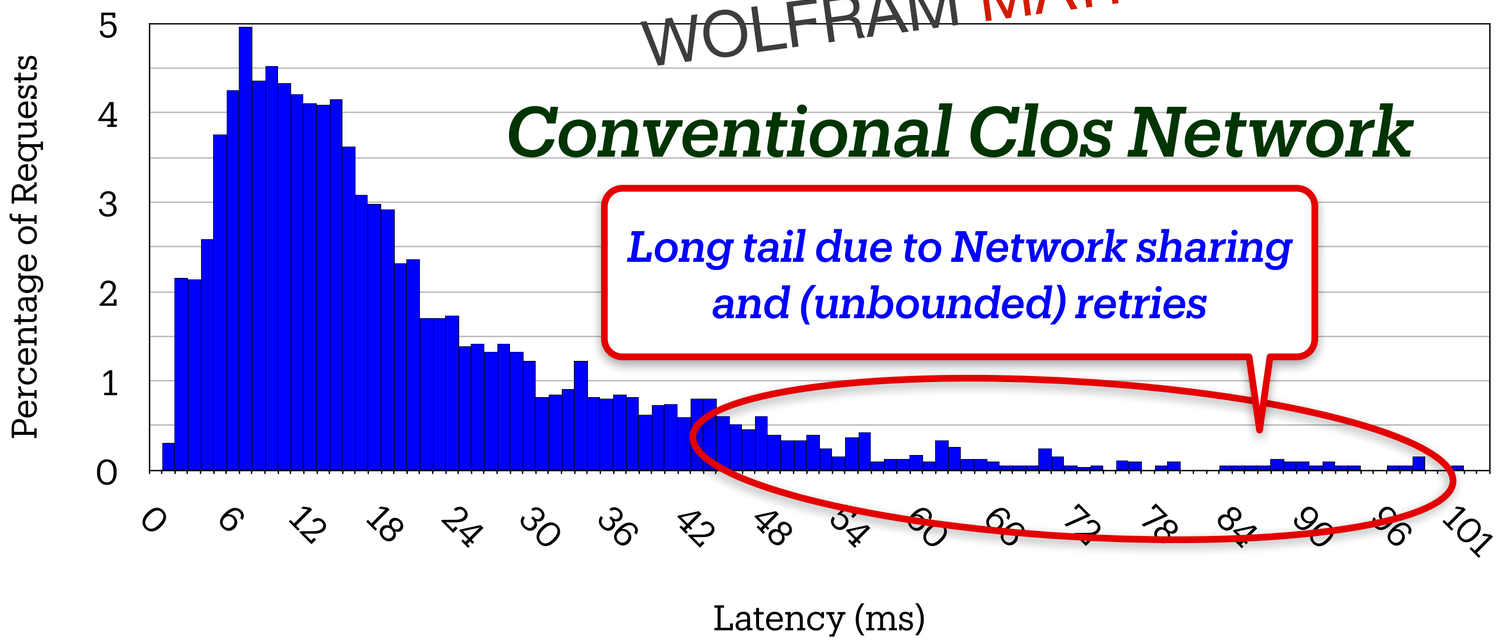
Daedaelus reduces latency in ways conventional networks cannot:

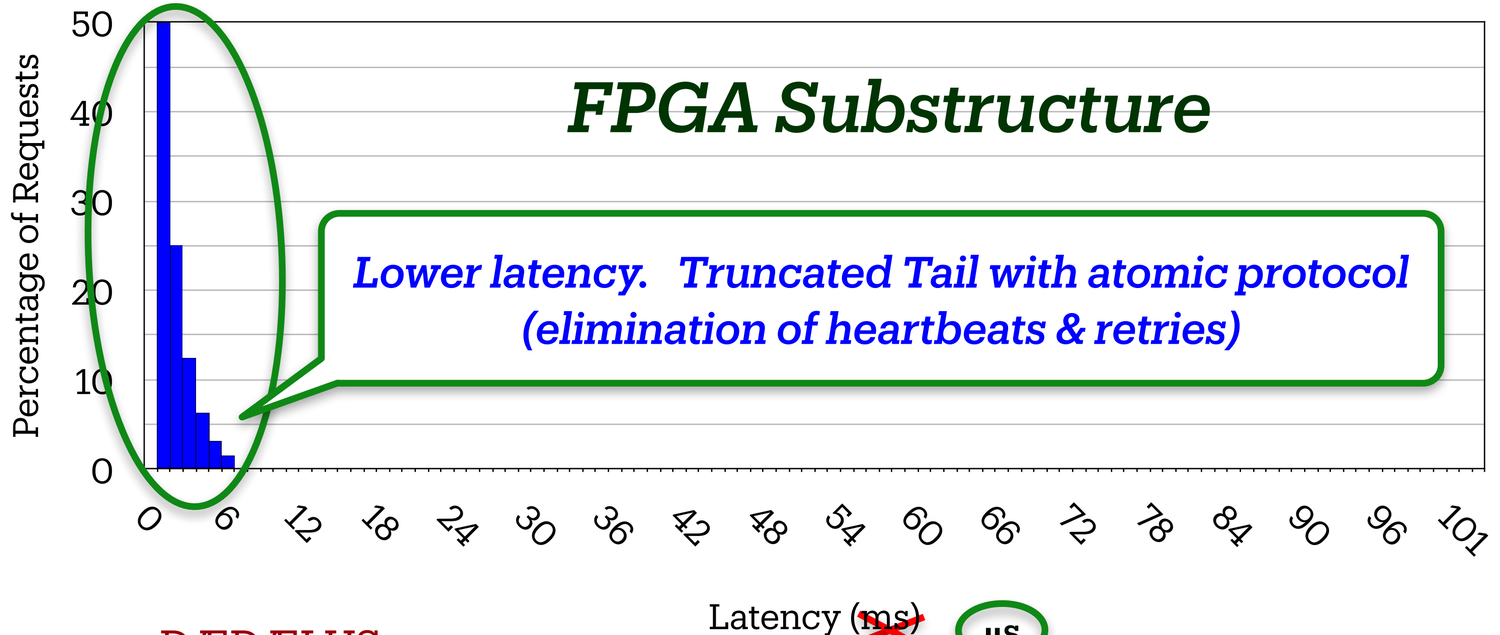
- Direct connections
- Multicast consensus, in parallel over 8 ports instead of serial over 1
- Truncated Tail Latency — protocol knows it failed or succeeded (without heartbeats or timeouts)



Daedaelus

WOLFRAM MATHEMATICA





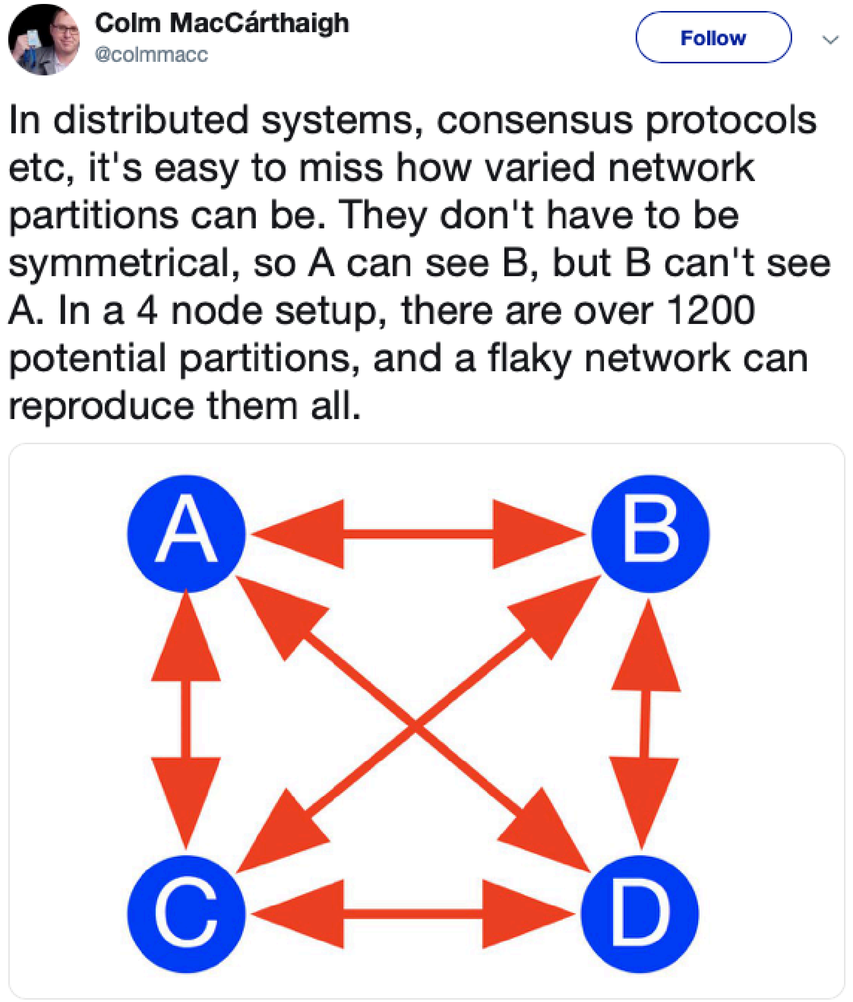
DÆ

DÆDÆLUS

Network Partitions

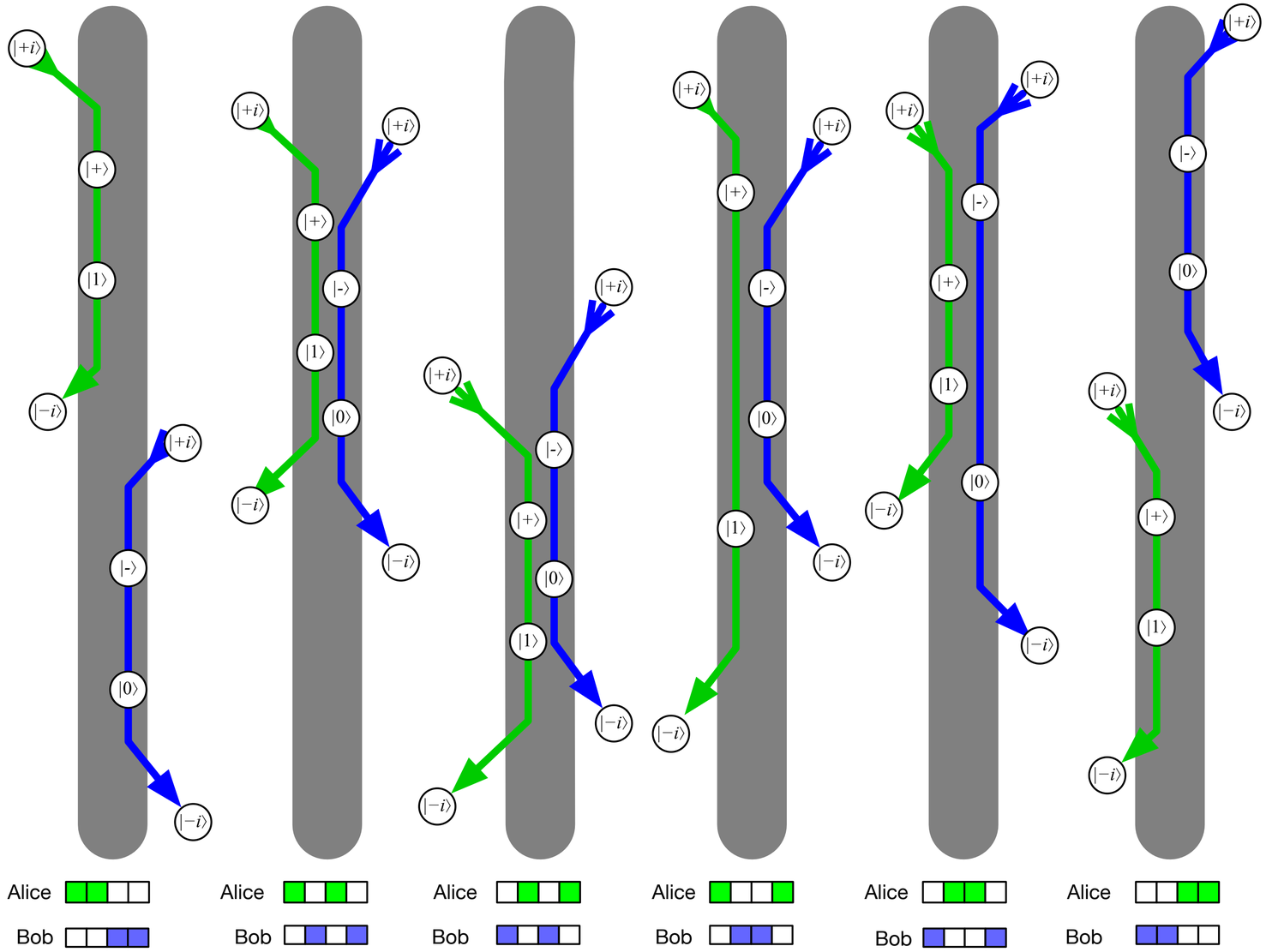
WOLFRAM MATHEMATICA

- Link failures are invisible (hidden) in a Clos
- They are 100% visible to us in a 4- Node mesh
- For 4 nodes, there are $(n(n-1)/2) = 6)$ links, 4 configurations on each link (00,01,10,11); gives $6^4 - 1 = 1295$ possible failure modes
  - And that's only the clean (binary) failures
  - *Flakey connections are much worse*
- DÆ protocol is designed to be exquisitely sensitive to packet loss and corruption
- We monitor, detect, diagnose link failures, and recover reversibly and automatically



DÆ

Serialization in the FPGA SerDes

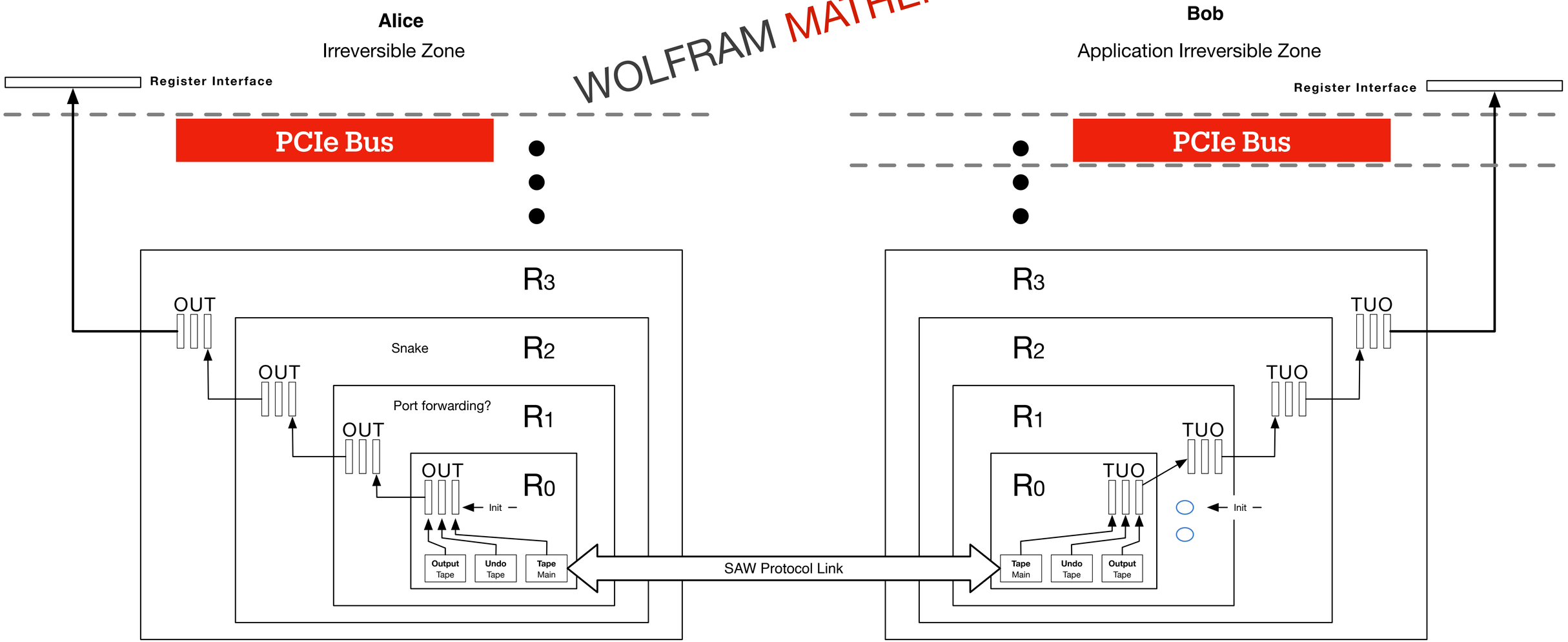


Petri-Spekkens Token Orderings in the FPGA SerDes and Disjoint Pairs

WOLFRAM MATHEMATICA

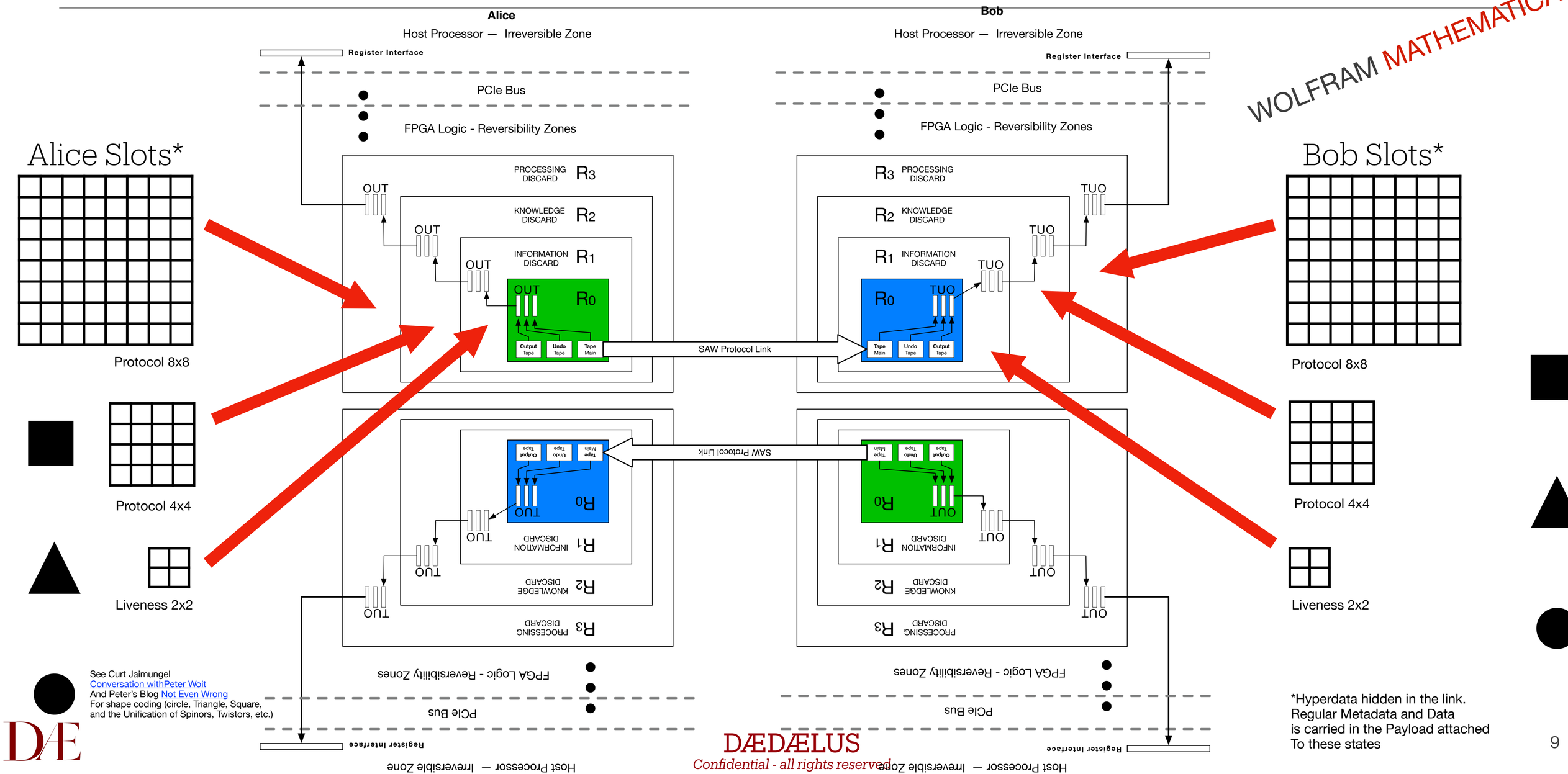
We proactively ***reorder events*** to match application constraints, eliminate race conditions, and prevent write skews before the database/application sees them as an error

Logical Reversibility of Computation



[Logical Reversibility of Computation (Charlie Bennett)]

Shannon - Alice to Bob - Two Way Mutual Information Slots



See Curt Jaimungel
Conversation withPeter Woit
And Peter's Blog Not Even Wrong
For shape coding (circle, Triangle, Square, and the Unification of Spinors, Twistors, etc.)

*Hyperdata hidden in the link.
Regular Metadata and Data
is carried in the Payload attached
To these states

Read Me First

WOLFRAM MATHEMATICA

- Don’t let anything go by you don’t understand at this stage. Ask questions!

Rethinking Datacenter Fabrix

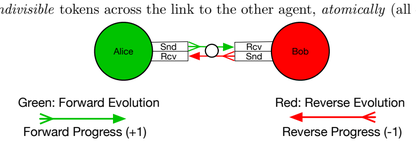
Many problems encountered in datacenters today arise from our inability to distinguish between a node that is merely slow from a node that has failed or become unreachable due to a network failure.

We take the two most recognizable elements in datacenters today: *servers* and *switches*, and refactor them into simpler, more foundational elements (fully *independent* failure domains): `cells` and `links`. A `cell` is a single type of node element (autonomous unit of compute, storage and packet processing). A `link` is an individual, bidirectional, computation object (an autonomous communication entity between *two* `cells`)[1].

A consequence of the former is that unifying node elements makes things simpler because we have only one type of node to manage instead of two. The consequence of the latter is profoundly more interesting: we raise the notion of a `link` to *first order* – a first-class citizen in the infrastructure – a bipartite[2] element of information with two *complementary* halves – *persistable* through failure and recovery events. i.e., a communication object that doesn’t rule out that some fault-detection and computation is involved.

An example `link` utility is *The I Know That You Know That I Know* (`TIKTYKTIK`) property; which enables us to address some of the most difficult and pernicious problems in *distributed systems* today.

Another example `link` utility is *Atomic Information Transfer* (`AIT`). Unlike *replicated* state machines[3] used throughout distributed applications today, `links` are *single* state machines: the two halves of which maintain *temporal intimacy* through hidden packet exchanges. When a local agent or actor is ready, the `AIT` protocol transfers *indivisible* tokens across the link to the other agent, *atomically* (all or nothing)[4].



These `TIKTYKTIK` and `AIT` properties are *composable*. Trees of `links` provide a resilient *conserved quantities* mechanism to reliably distribute tokens among *agents* on an application graph. Intermediate cells *promise*[5] to never lose `AIT` tokens. This defends against lost tokens because if any part of the chain (or tree) breaks, alternate paths are available to seamlessly recover the conserved quanity and continue operation.

By strengthening the system model, `links` and `AIT` provide a general foundation to solve many distributed systems problems[6], such as failure-detection, consensus and distributed transactions.

Failure Modes

One might imagine we could achieve the properties of `links` over existing switched networks. If each host (or its NIC) maintains its half of the *shared state*, then shouldn’t the switched network be able to act as a proxy for a single *logical* link? When a switched network fails, and reroutes, can’t the two sides (NICs) just stitch the two halves of the *shared state* back together again?[7]

This simple hazard analysis[8] misses a fundamental issue: *networks don’t maintain state on behalf of applications*. Switches drop packets (and state) whenever *they* feel like it, so there are many more ways for *logical links* to get confused over switched networks and compromise the integrity of the *shared state*.

Key issue: Switched networks may drop packets anywhere along the path; eradicating state *and* events needed to maintain *promises* and *liveness* respectively. When a `link` fails, both sides are preserved. If there is a failure in the *token transfer* it can always be detected, and retransmissions occur only on a *real* failure (such as disconnection–where alternative routes are explicitly coordinated with applications), thus enforcing that tokens have no duplicate or out of order deliveries on the `link`[9,10].

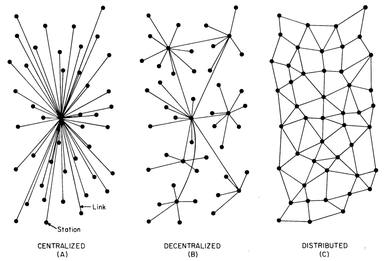
Summary-Links.pdf

Rethinking Datacenter Management

Owners and operators of the network determine the relationships among distributed applications today. Minimum spanning trees, on which all routing is done, are built, and torn down, by *switches*; based on protocols standardized long ago when we first learned how our computers could communicate.

Today’s datacenter architects build their infrastructures using two kinds of boxes: *switches*[1] and *servers*[2]. They connect them using individual cables, which they bundle together to make them convenient to route within and around physical structures. This forms a *centralized* or *decentralized*[3] topology, where the switches become hubs and servers become leaves.

Paul Baran’s classification provides insight: `CENTRALIZED (A)` shows 46 univalent nodes connected to a special high radix or *valency*[4] hub. If the central hub dies, all nodes are cut off. `DECENTRALIZED (B)` shows 47 nodes and links, 7 nodes with a valency of 5-7 serve as switches, and 40 as univalent (leaf) nodes. If one of the switches fails, the network fractures into isolated partitions; and only nodes within the partitions can continue to communicate locally. `DISTRIBUTED (C)` shows 47 identical, multivalent (valency $\sim$ 5) nodes, and 98 links. The network has better resilience: failed nodes are routed around, and many links must fail before *any* node is finally isolated.



Datacenter Topologies

`CENTRALIZED` topologies are avoided because they represent bottlenecks and have a single point of failure. `DECENTRALIZED` topologies[5] are most prevalent, which is surprising given how *non-optimal* they are from a perspective of distributed microservices[6], container life-cycles[7], migration, elasticity and resilience[8,9]. Switches that perpetuate this model, are embarrassingly complex, unreliable, arcane, and parochial. This results in very high operational costs, poor security/high vulnerability, and nothing close to five nines reliability[10]. `DISTRIBUTED` topologies are rarely used (so far) in datacenters, except for a few HPC applications[11].

However, within the same *or lower* capital cost, `DISTRIBUTED` topologies provide: greater resilience, lower latencies, higher available bandwidth and far more flexibility; by connecting `cells`[12] directly with neighbor to neighbor (N2N) connections rather than through a switched or aggregated network[13]. By not perpetuating the management complexity of switched networks, and introducing new, simpler, control/forwarding planes through `cells`, we can also dramatically lower operational costs.

Perhaps the time has come to recognize the genius of Paul Baran’s insights, and ask why `DISTRIBUTED` topologies are not deployed in datacenters, where their resilience and security can be readily exploited?

Datacenter Programmability

Two types of teams co-evolved to manage modern datacenters: one to design and manage the networks, and one to program and manage the servers. This worked when datacenters had a single owner or tenant, their applications and physical infrastructure evolved slowly, and different business units could work within their own silo’s. This is no longer a viable architecture in today’s highly dynamic multi-tenant datacenters.

Distributed applications can no longer afford to be held back by the slow pace of networking innovation.

Rethinking-Datacenter-Management.pdf

FAQ: GVM

Graph Virtual Machine (GVM): Naming, Topology, Equations

DAEDAELUS (DAE) has developed a low-cost, high-performance Transaction Fabrix^{TM} for datacenters using N2N (Neighbor-to-Neighbor) links (repurposing Ethernet frames), with selectable (reliable, tree cast) delivery based on a new way to rendezvous on *trees* (without fixed IP endpoint names). The lowest level graph is a set of black trees,where each black tree in the set is named by the cell on which they are rooted; together they form our *groundplane*. The architecture is layered upwards through logical stacked trees that provide hardware-enforced *confinement*, and virtual stacked trees that provide a foundation on which developers can automatically provision microservice graphs with *an equation*.

Introduction

The DAEDAELUS (DAE) Transaction Fabrix (TF) uses standard Ethernet NICs. However, East-West interactions use tree-based naming for address groups of closely related microservices in *sets*, rather than the source/destination addressing mode that practitioners of conventional networks are accustomed to[1].

This simple change, implemented *below* L2, is the first step to enabling *Structured Topology Management* of microservice sets that need to evolve dynamically in response to perturbations (failures, disasters, attacks).

With 8 ports per cell, a 1-hop cluzter comprises 9 cells. The center cell is the root of its own tree, and the default transaction manager for consensus operations. Neighbor cells become proxies for resource (compute/memory/storage) elasticity, dynamic load balancing and failover. A preselected neighbor is provided for voluntary (or involuntary) failover when the center cell dies or needs to be taken out of service.

Trees extend to any number of hops, out to the edge of the Transaction Fabrix (the boundary to the conventional network). The primary means of addressing is *radial* (by port) - it knows which ‘direction’ a target entity exists along, but may not know ‘how far’ along a branch, or which sub-branch it exists. This is an essential mechanism – to allow the migration of an entity without having to change addresses from the point of view of the source requestor. This mechanism also enables failover and elastic growing and shrinking of a microservice set in response to traffic demand.

Trees are subsetted to enable hardware confinement between zones. These are used typically to create tenants and subtenants within a datacenter. All these API’s are directly available to the distributed systems developer through an API. Within the Transaction Fabrix, there is no need for conventional datacenter segmentation using firewalls, iptables, or ACL’s in the routers.

The central mechanism driving this simple yet powerful approach to datacenter address management is the Graph Virtual Machine (GVM): A protected virtual machine environment running in the FPGA Substructure (SmartNICs). All packets going into a cell go through this vantage point enables: routing to be done differently, provisioning to be done differently, and security to be done differently.

Fundamental organizing unit: Graphs, not Lists

The entire datacenter is a graph (groundplane). All (non-trivial) applications in datacenters are sub-graphs. We create sub-graphs by recursively stacking trees above the groundplane. These sub-graphs are the regions or tenants that respond to commands from an orchestrator to do something. Within each subtenant, developers can orchestrate their own microservice sets without being required to work within the constraints of conventional networking tools, such as command line interfaces and ACL’s in switches/routers, or the cumbersome sets of rules used in todays internal segregation firewalls.

It is inevitable that the above description will create more questions than it answers.
The rest of this document is a deeper dive for the technically curious.
This FAQ is one of many in the DAEDAELUS repository.

[1]This FAQ assumes that the reader has read the first two ‘read-me-first’ documents: Exec-Summary-Links (Rethinking Datacenter Fabrix) and Exec-Summary-TRAPH.pdf (Rethinking Datacenter Management)

FAQ-GVM.pdf

Taming Indefinite Causal Order on Ethernet

- Our Protocol in a Nutshell*
- Alternating Causality enables Alice and Bob to “catch and hold” the direction of the Edge (The causal arrow)
- Once captured, it is maintained by Intanglement (circulating packets)
- Can only be reversed by a “higher” Ancilla

WOLFRAM MATHEMATICA

We Implement a Classical version of the Quantum Switch in FPGAs

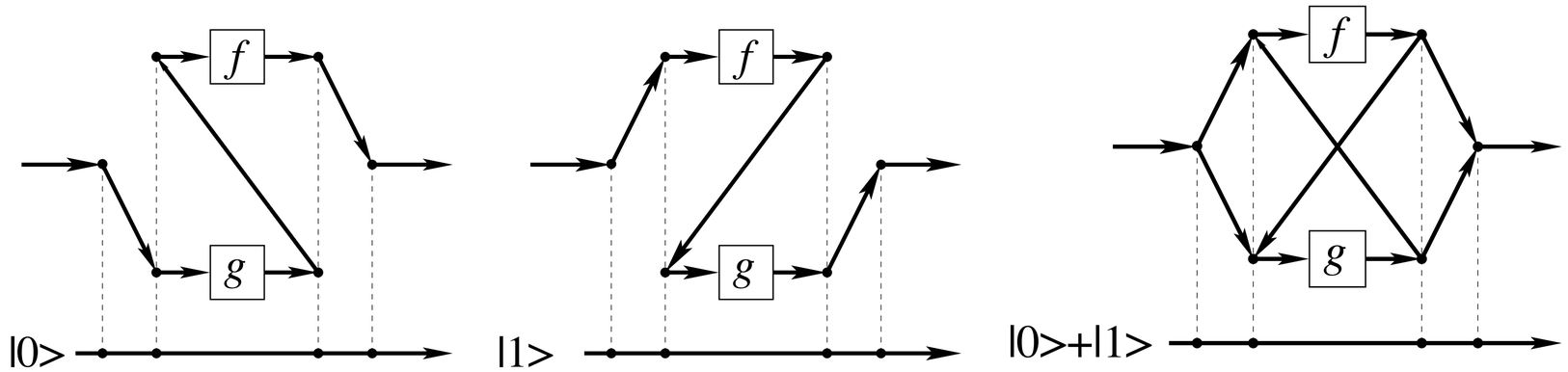


FIG. 2: Quantum machine with classical control over movable wires.

We control the “direction of evolution” in the boxes f and g above, using our “reversible” protocol on Ethernet

See Packet format discussion in Separate Document

* Chiribella et al. Quantum computations without definite causal structure

* Rubino et al. Experimental Verification of an Indefinite Causal Order

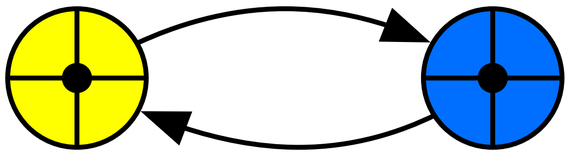
Multiway Link Protocol

WOLFRAM MATHEMATICA

Trapped (circulating) “information” token on link, exchanged with strata above, or strata below for token ***conservation*** in and between layers

- Interfaces for logical time (Lingua Franca) reversal ancilla for reactors
- Causal invariance leads to precise definition of confined Shannon information recovery (based on knowledge from Cluster Liveness), in the face of *all hazards* presented via fault injection, via externally accessible ancilla from the *next* level up. i.e., the ONT Petri place and oracle

Superposition of Arrow Head & Tail.
Indicates Alternating causality is occurring under the hood, but you can’t see It because reads are destructive.



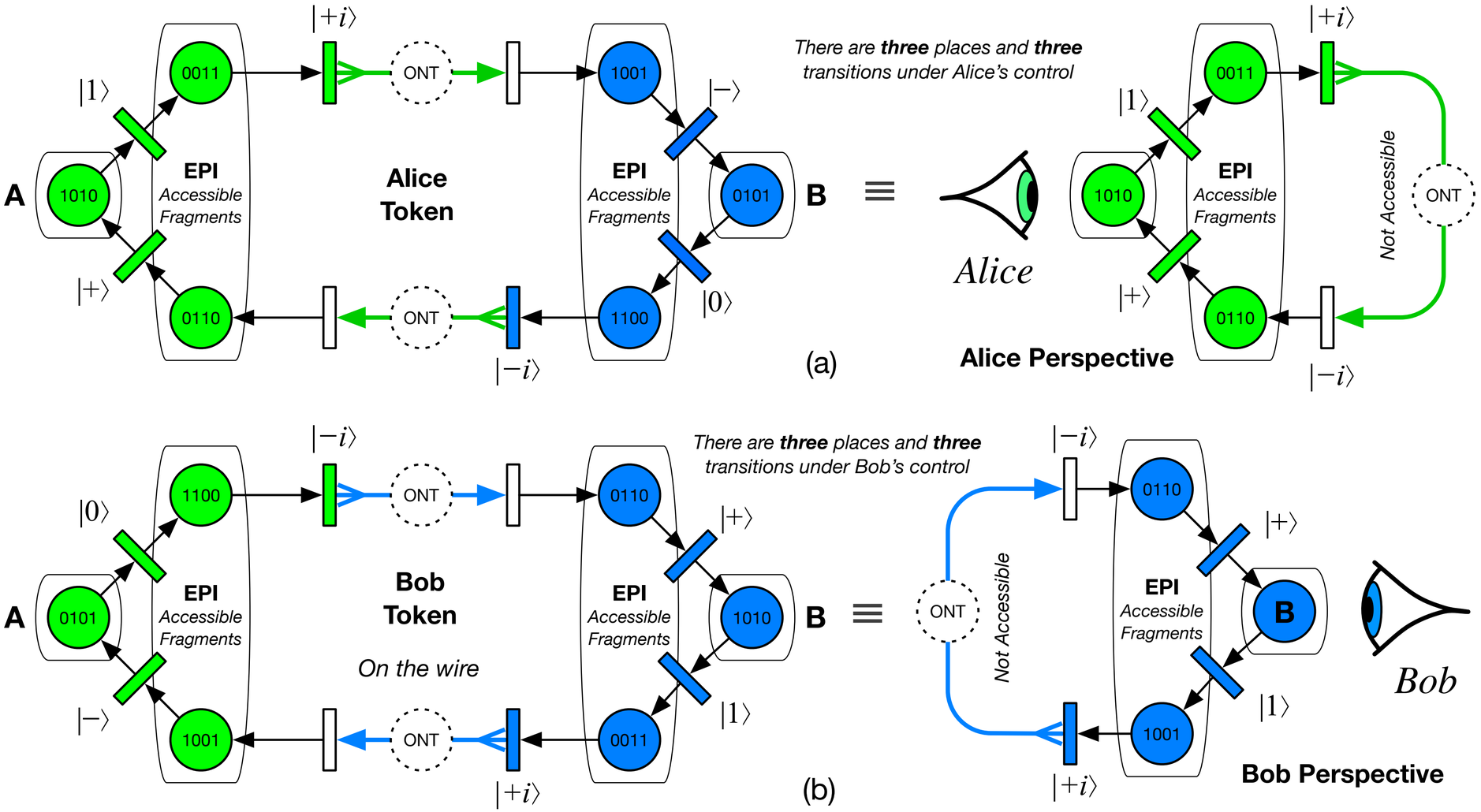
We separate the physics of entanglement From the mathematics of entanglement.

Dual SAW Petri Spekkens

WOLFRAM MATHEMATICA

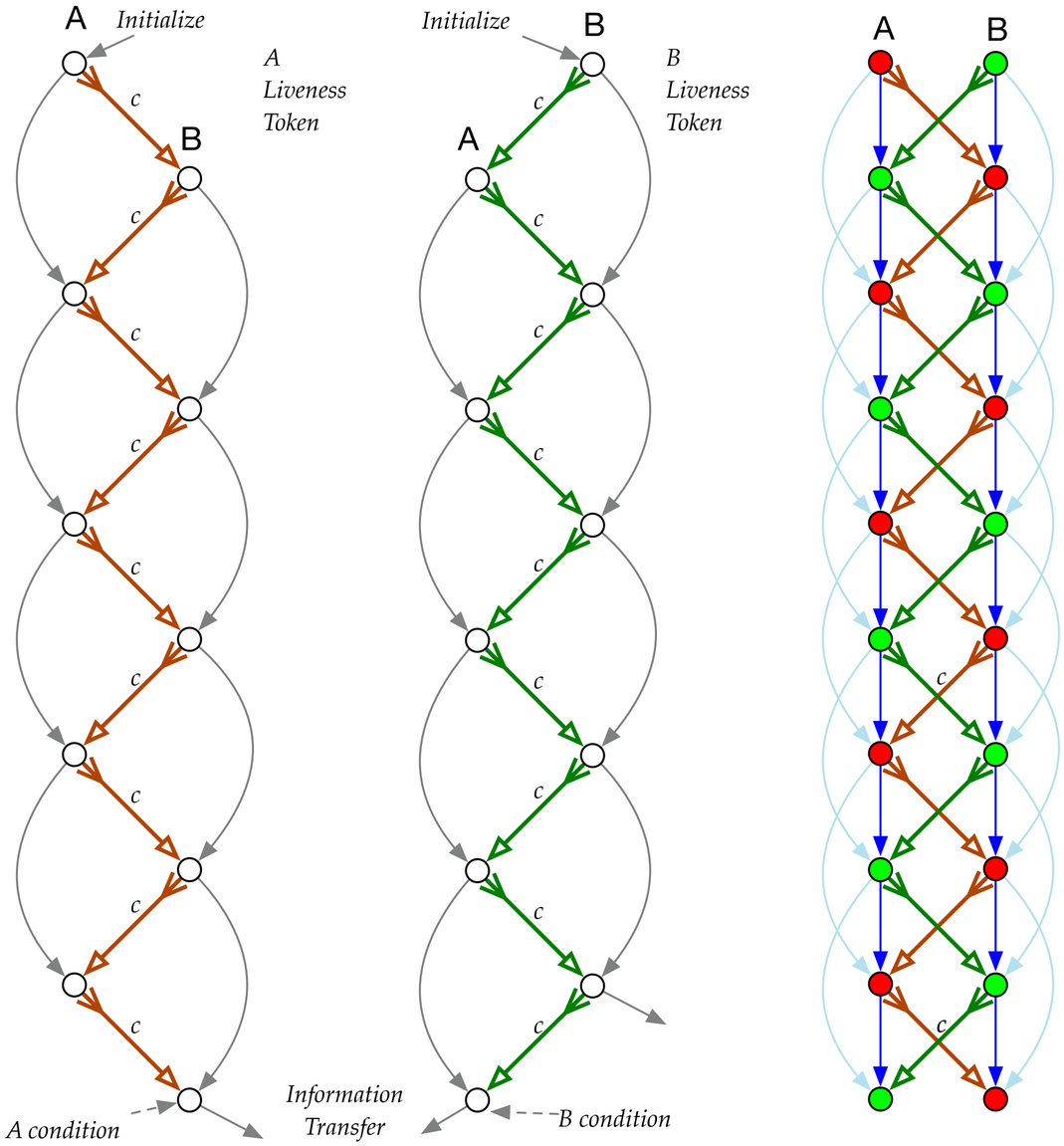
https://community.wolfram.com/groups/-/m/t/2575423

Petri-Net Model



Ethernet Link Liveness

WOLFRAM MATHEMATICA



Alternating Causality protocol

Imagine an Ethernet cable connecting two SmartNICs

Liveness provides symmetric question & answer dialog:

Are you there? **Yes I am!** (First slice of frame)

Each frame may be modified *only by the receiver* when it has the frame in its possession. There is only one serializability focus per frame, but there are two frames: one owned by Alice, one owned by Bob.

- In Alice's frame we say that Alice's token is owned, but Bob's token is borrowed.
- In Bob's frame we say that Bob's token is owned, but Alice's token is borrowed.

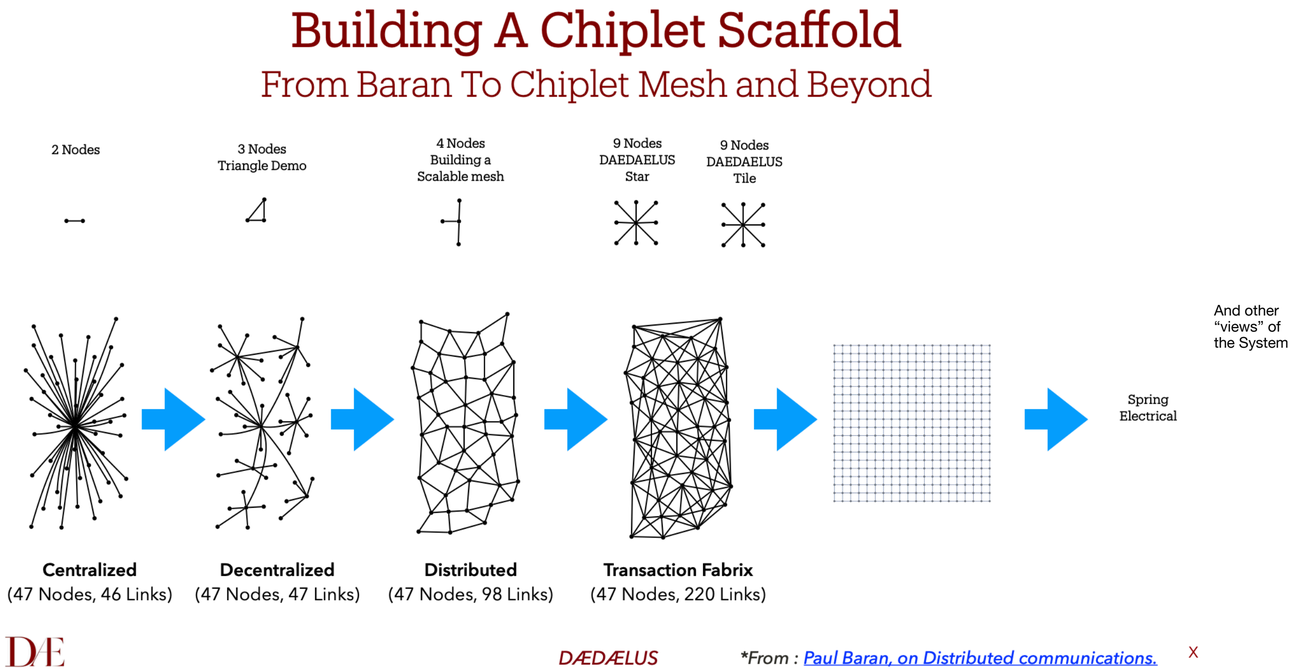
Following Rust rules.

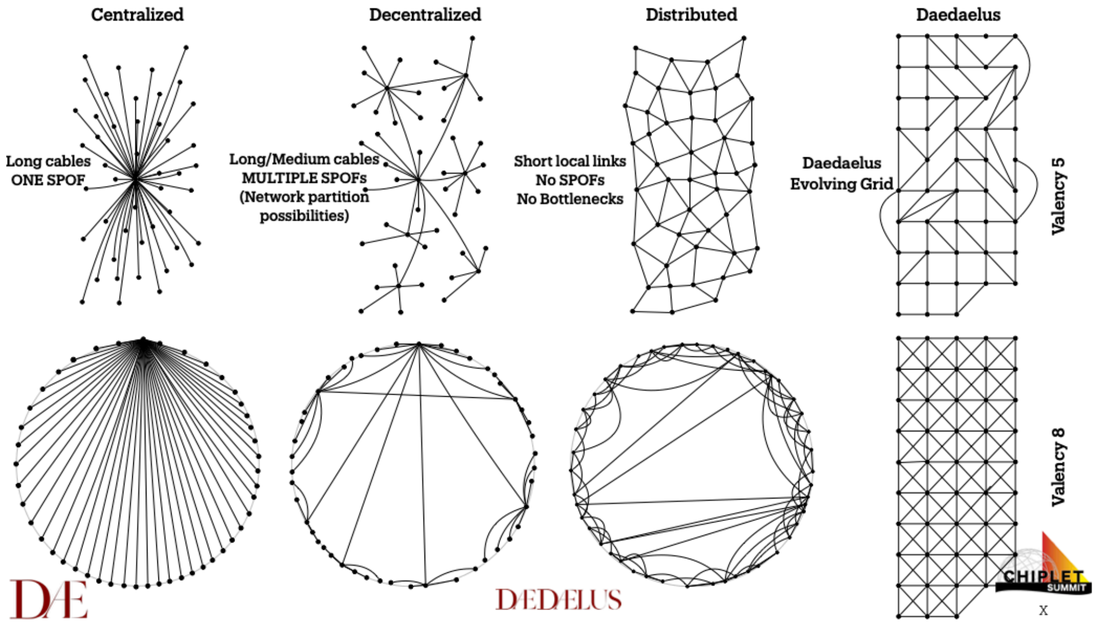
Graph Metrics

“Discovering” the Graph from Local Observer

Resources:

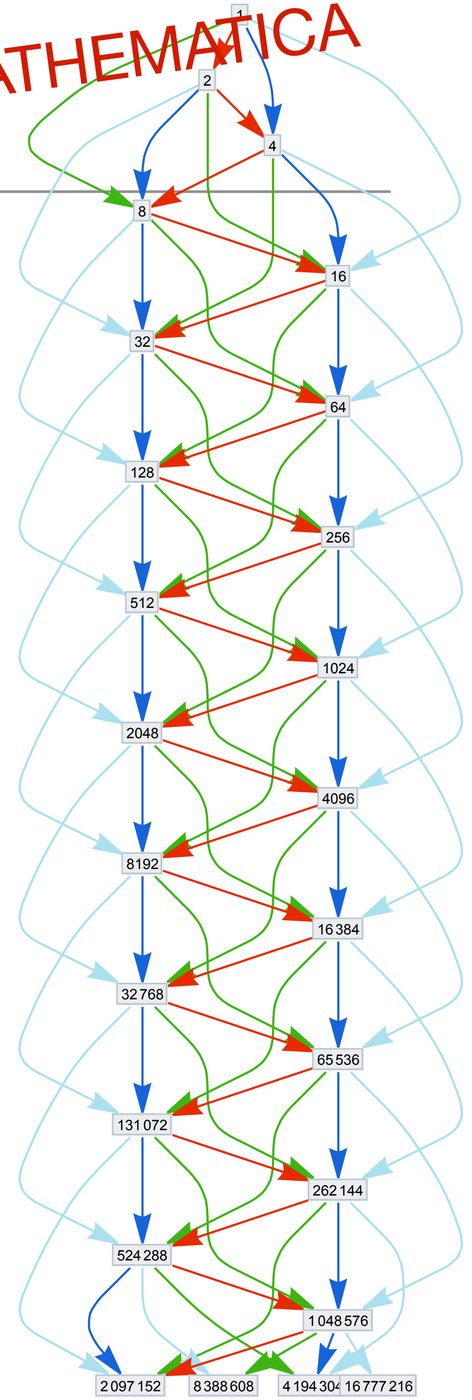
1. Classification of Graph Metrics
9. Dijkstra (from LinkedIn Mathematica Demo of Shortest Path)
10. Graph Laplacian (Counting Spanning Trees)





MultiWay View of Link

WOLFRAM MATHEMATICA



An example of a dynamic yet stable causal structure (multicomputation reducibility) using a dual token Petri net, spanning a single bipartite link. Quantum Ethernet. Dual SAW-Petri-Spekkens Protocol.

Imagine an Ethernet cable, with each end connected to a SmartNIC -- each with a server attached to the other side of it's PCIe or CXL bus. Conventional heartbeats enables *liveness* on the link with a symmetric question and answer dialog: *Are you there?* <-> *Yes I am!*

The link continuously circulates fixed frames of shared information. The figure to the right shows two independently circulating shared frames green & red. These two frames are shown as concurrent in this graph, but they will serialize, *convoying*: with `alternating causality' on a pair of Ethernet transmit/receive channels. Each frame may be modified only by the receiver when it has the frame in its possession. There is only one serializability focus per frame, but there are two frames: one owned by Alice, one owned by Bob.

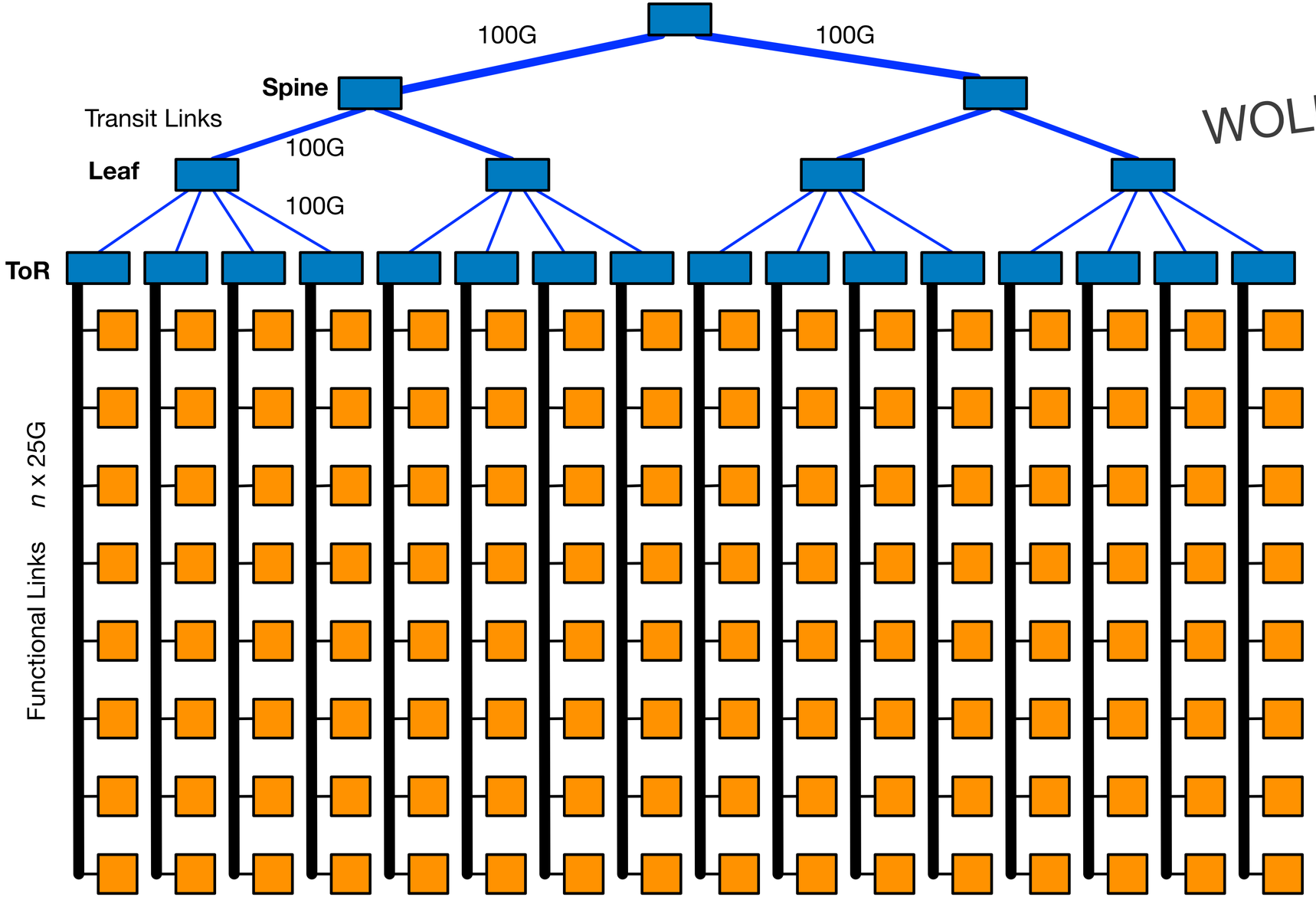
There is thus one circulating (direction) of causality.

Each causal frame has an identity (Alice or Bob) designated in the protocol after an appropriate symmetry breaking. In Alice's frame we say that Alice's token is \emph{owned,} but Bob's token is \emph{borrowed}. In Bob's frame we say that Bob's token is \emph{owned,} but Alice's token is \emph{borrowed}.

\subsection{Alice/Bob Independently initialized with a Petri Token}

\paragraph{[1] Begin with Alice} Alice sends two pieces of information: Red to the other side of the link, Blue to self (to merge with future operations).

\paragraph{[2] Begin with Bob} Bob sends two pieces of information: Red to the other side of the link, Blue to self (to merge with future operations).



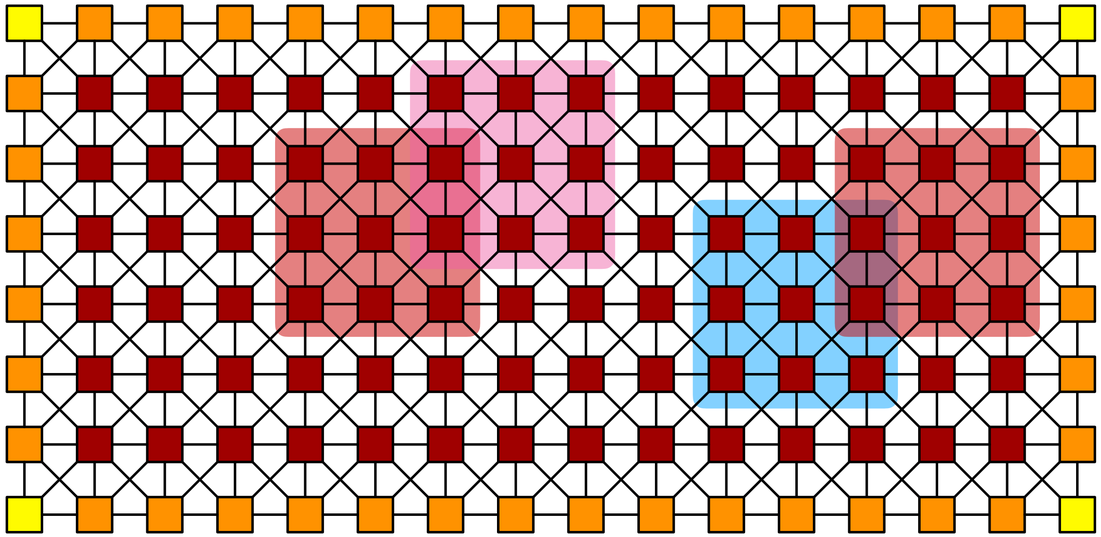
Conventional Clos

WOLFRAM MATHEMATICA

Graph (Network) Automata Foundations

8-port (Moore) neighborhood exploits physical link topology substructure within racks. But with *fragile* cabling (where links and nodes must be healed around dynamically with Local Observer View (LOV) information only)

- Paves the way for highly dynamic computational strata in layers above
- Leads to a 9-Cell deployable unit of distributed computation we call a hypercell (Tile)*



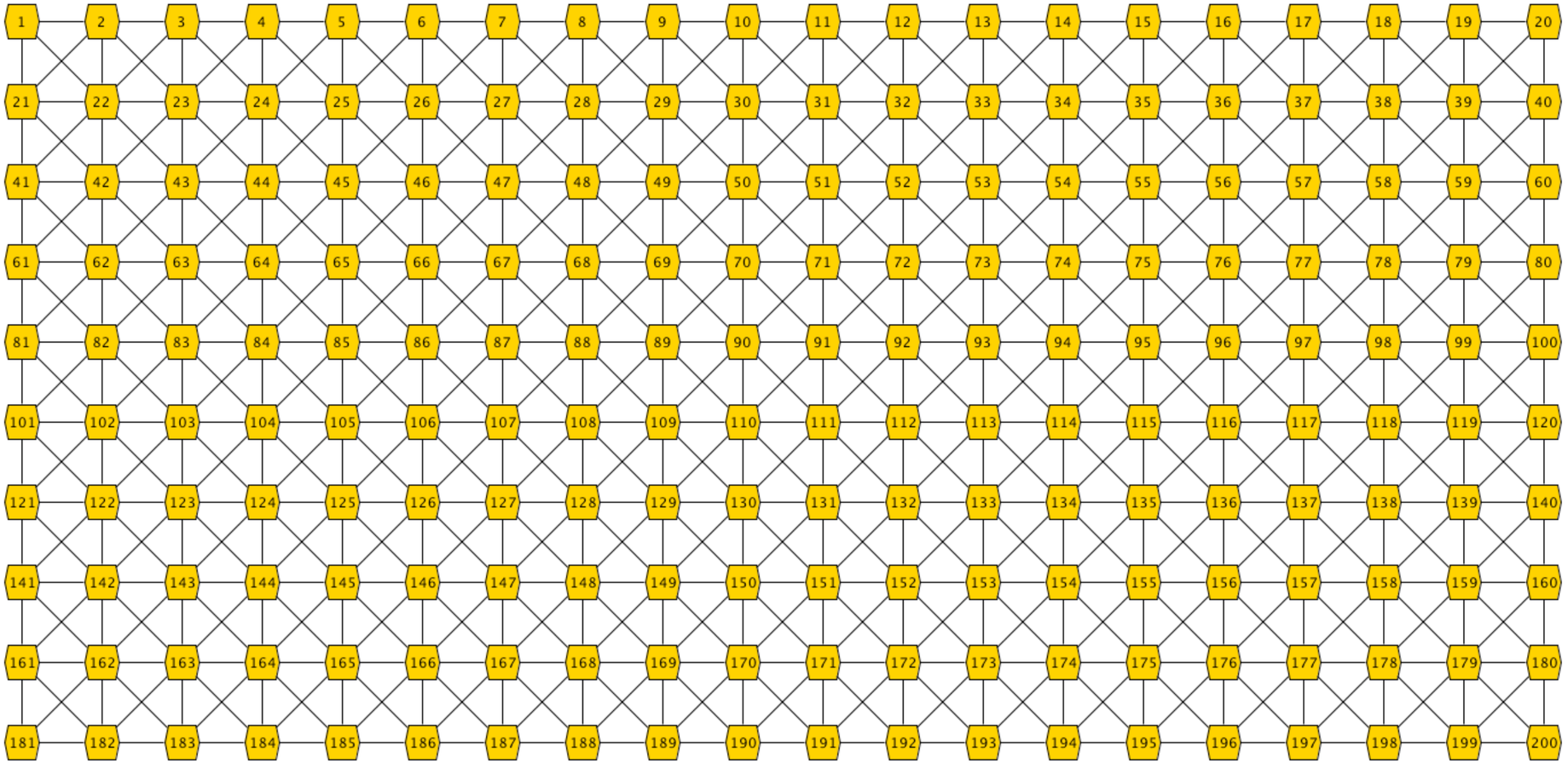
WOLFRAM MATHEMATICA

**This physical structuring must not be conflated with a regular lattice on a cartesian coordinate system because our graph algorithms are highly dynamic, adapting to both the addition of new resources, or the deletion (i.e. failure or removal) of old ones. A fixed cartesian coordinate system (hypercube-like or otherwise) would be far too fragile as all labels throughout the graph would have to be replaced on failures and reconfigurations*

Regular or Irregular Lattice?

Real Infrastructures are messy

WOLFRAM MATHEMATICA

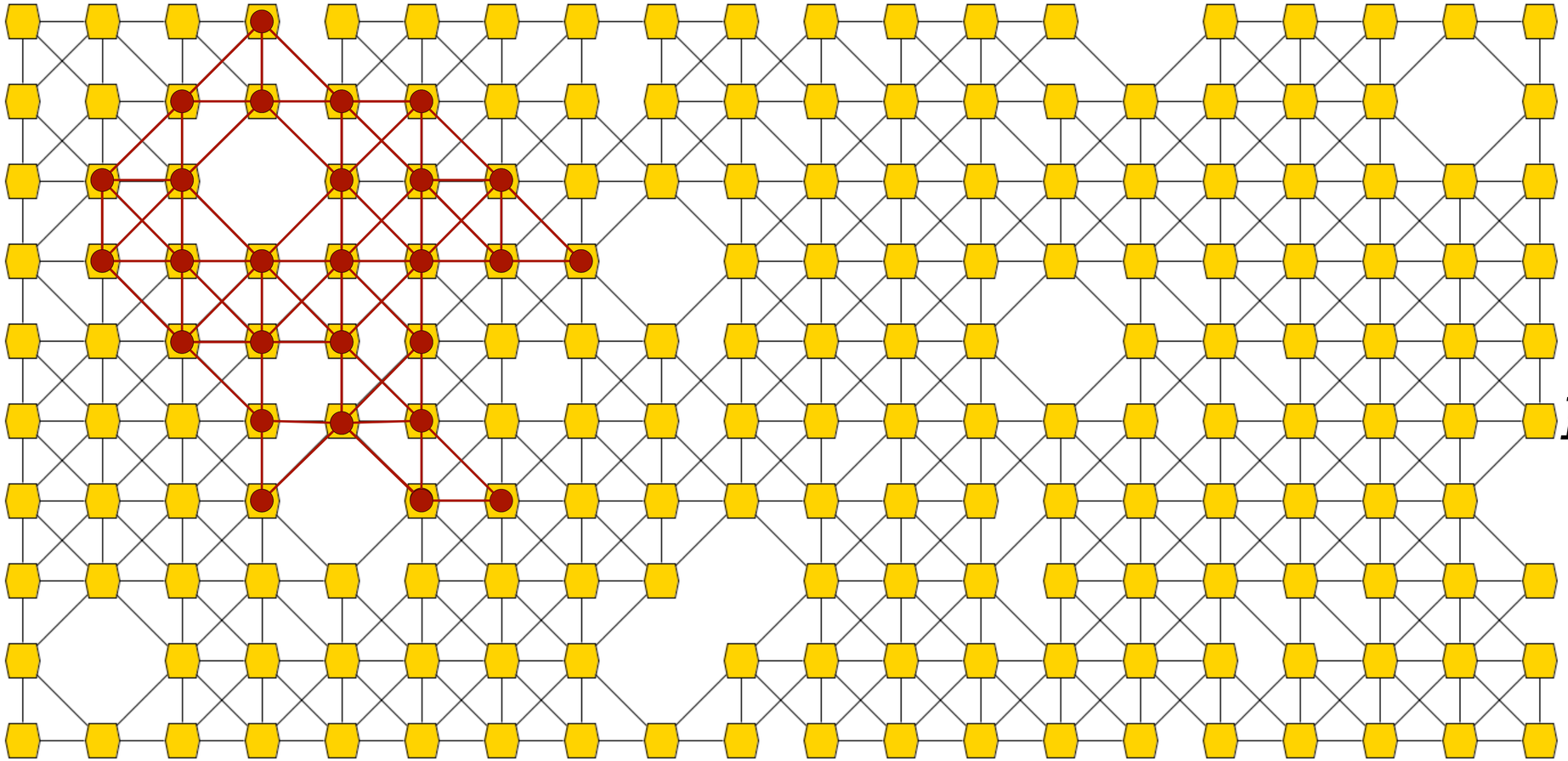


~~*Absolute / Cartesian Coordinate Addressing*~~

Neighbor to Neighbor (N2N) Lattice

Manage on a Tree Compute on a Graph

WOLFRAM MATHEMATICA

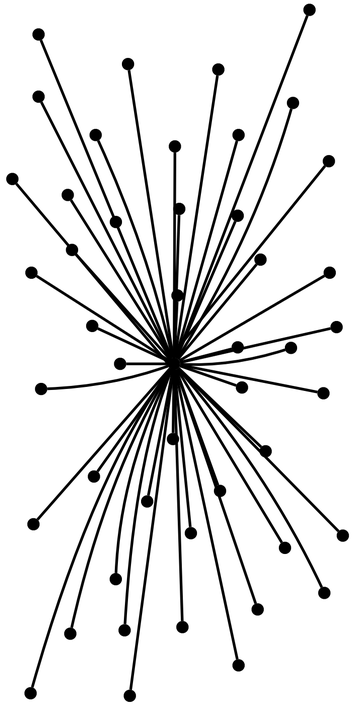


Relative Address-Free Routing

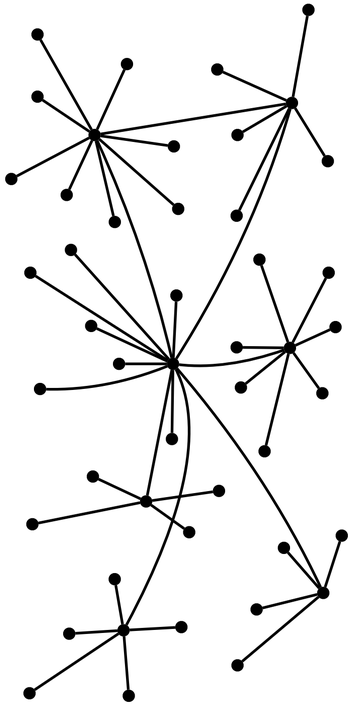
Graph Virtual Machine (GVM)

WOLFRAM MATHEMATICA

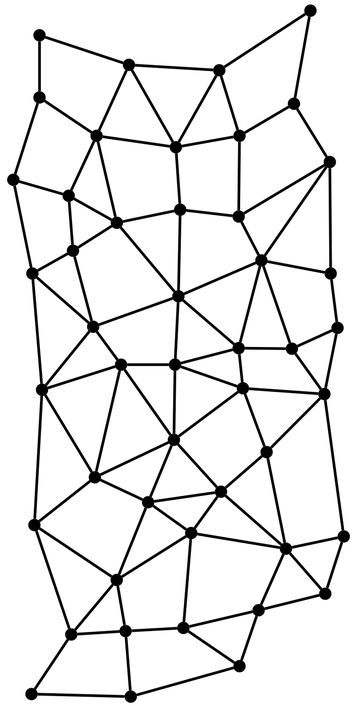
Build and tear down ‘named’ graph relationships (sets/graph covers) based on properties of a cell or hypercell. Demonstrate using Paul Baran examples*:



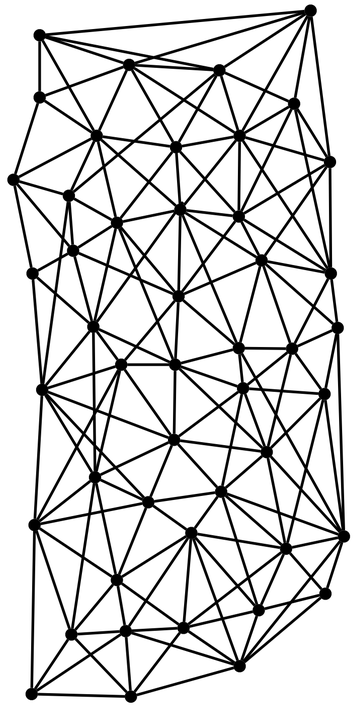
Centralized
(47 Nodes, 46 Links)



Decentralized
(47 Nodes, 47 Links)



Distributed
(47 Nodes, 98 Links)



Transaction Fabrix
(47 Nodes, 220 Links)

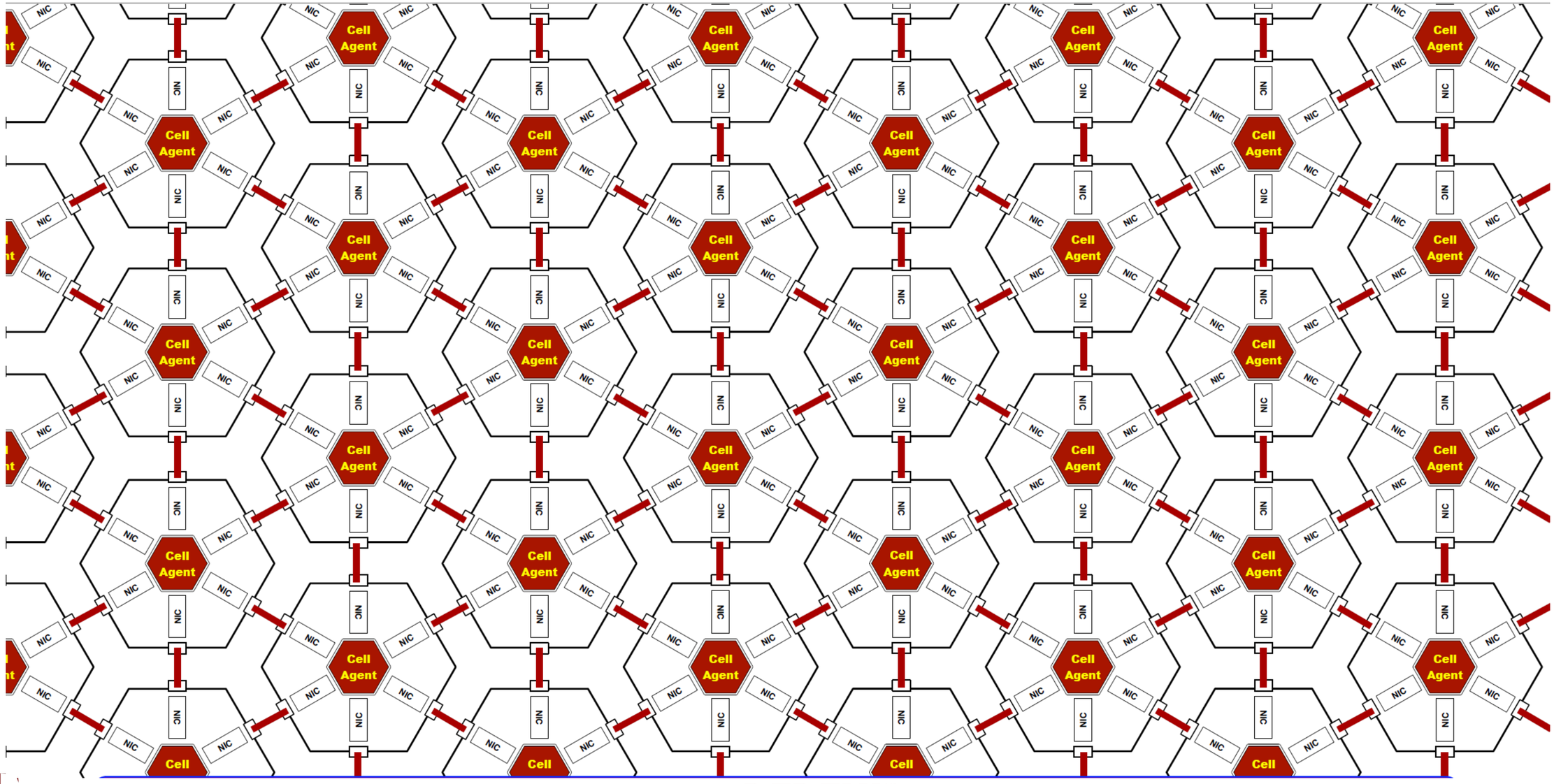
Count connectivity using: Matrix Tree Theorem & Graph Laplacian

*From : Paul Baran, on Distributed communications.

Physical Plane (mesh) of servers

Not a causal network yet (those are stacked on top)

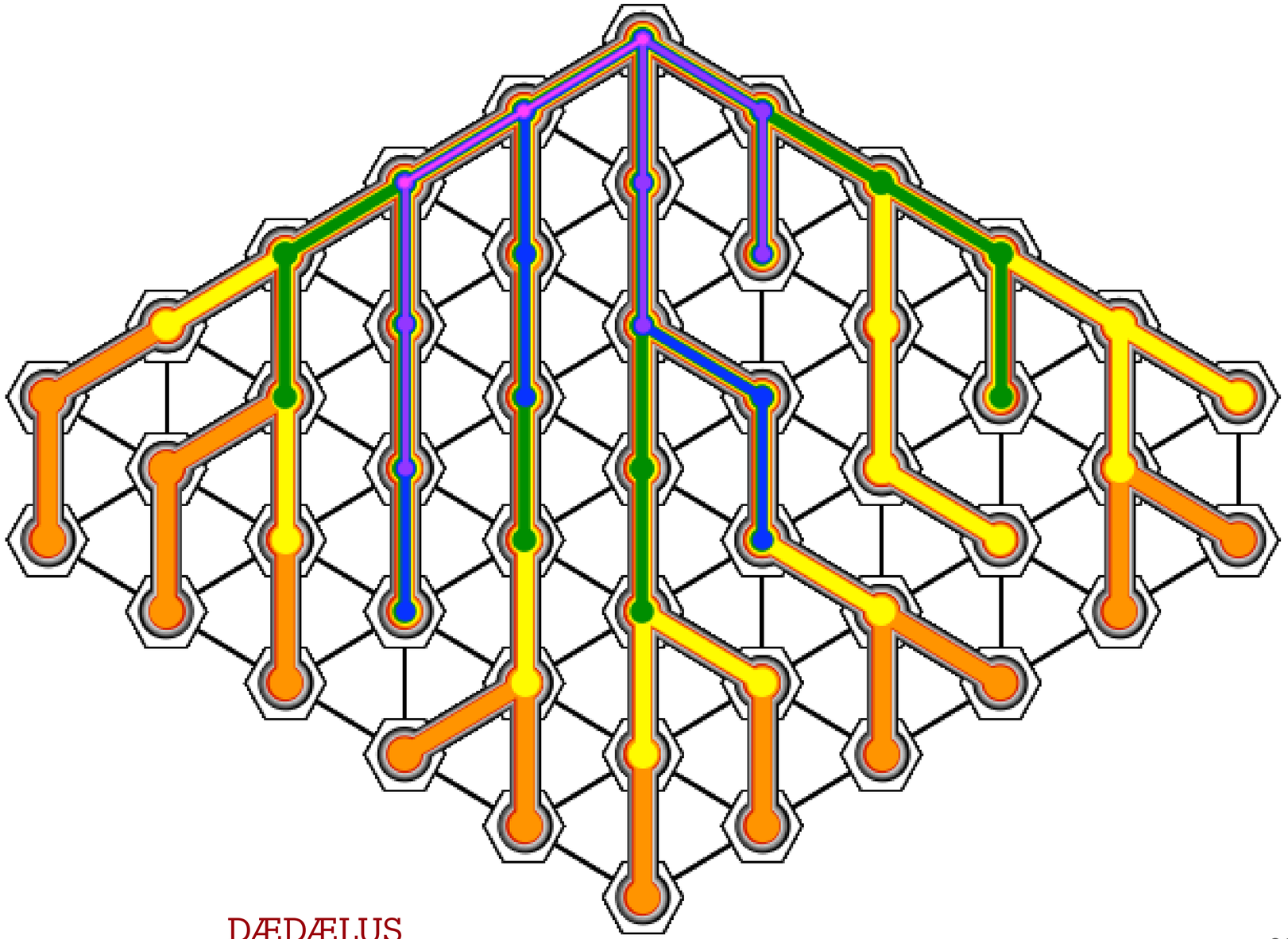
WOLFRAM MATHEMATICA



Stacked Trees

WOLFRAM MATHEMATICA

- One “stacked tree”
  - Rooted at “top”. Graph covers shrink as you go higher up.
  - They are “subsets” for provisioning & control
- Each node in this picture controls a similar set of trees, rooted on itself
- These graphs are specified by the Graph Virtual Machine (GVM)



Composable (Stacked) TRAPHs

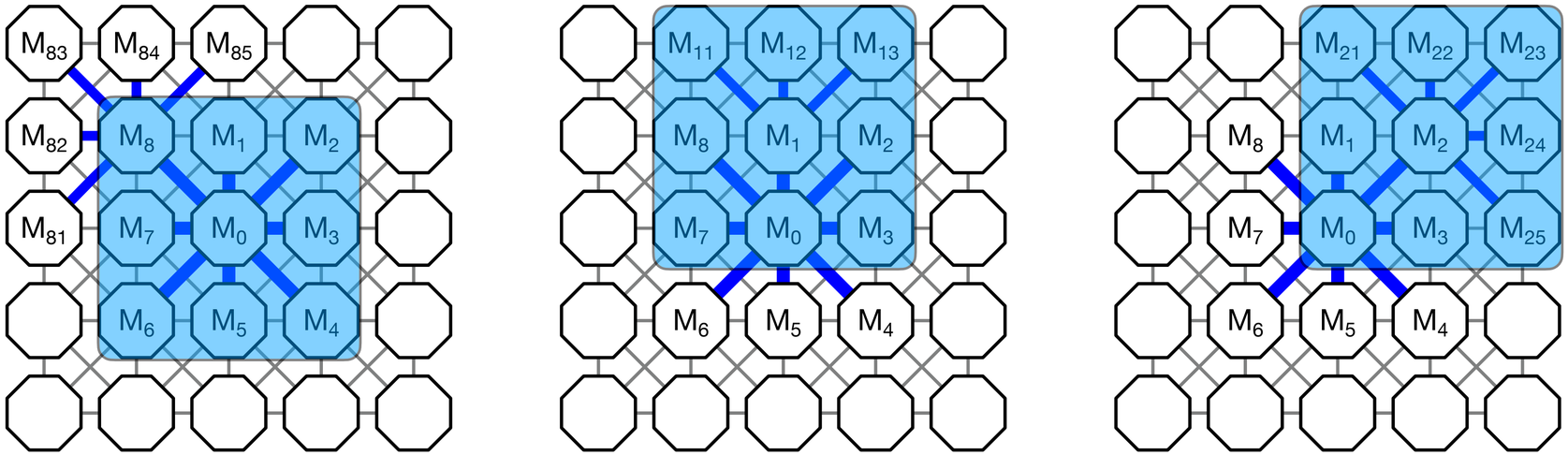
WOLFRAM MATHEMATICA

- Composable (Stacked) TRAPHs (TRee-grAPHs) Select one, or a small number of strata for experiment and visualization
- Use Cluster Liveness model to demonstrate spatial confinement
- Visualize TRAPHs for stacked trees, show physical, logical, virtual
- Demonstrate (local) distributed consensus within the tile. This 9-cell Tile also gives us a natural unit of fault-tolerance. Each cell is (by definition) the center of it's own tile
- The self-cell is thus pre-allocated in various algorithms as the center (Captain) of its own world for the purpose of initiating distributed algorithms, such as routing, consensus, autoscaling and load balancing

Hypercell Tile (Local) Addressing

WOLFRAM MATHEMATICA

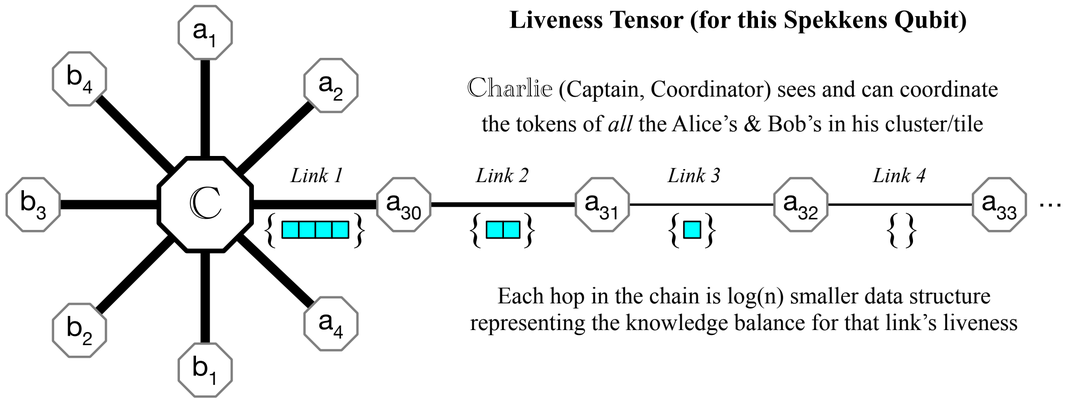
- Demonstrate Multiway routing protocol between tiles on DAGs
- Demonstrate healing around individual link failures
- Demonstrate remote consensus between tiles distributed throughout the graph, in the presence of injected failures



Tree Addressing from M8 | Tree Addressing from M1 | Tree Addressing from M2

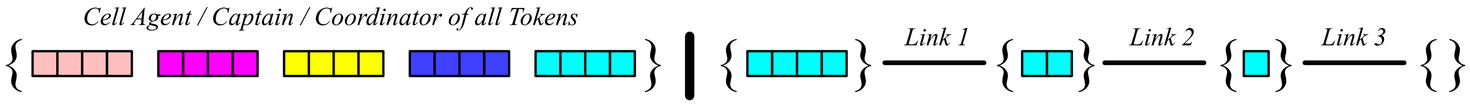
Tree Addressing is *Relative* — It looks different from each Tree's perspective

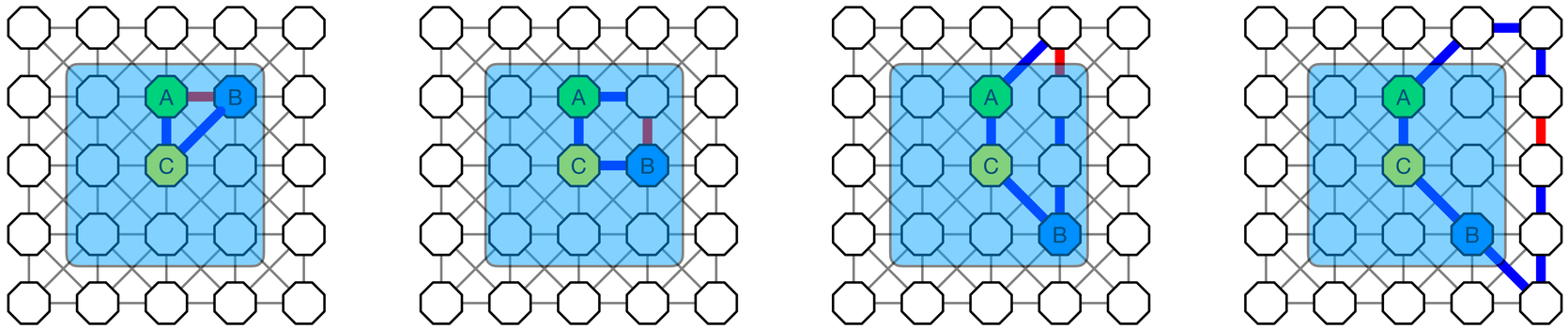
Cluster Liveness (Sensing your environment)



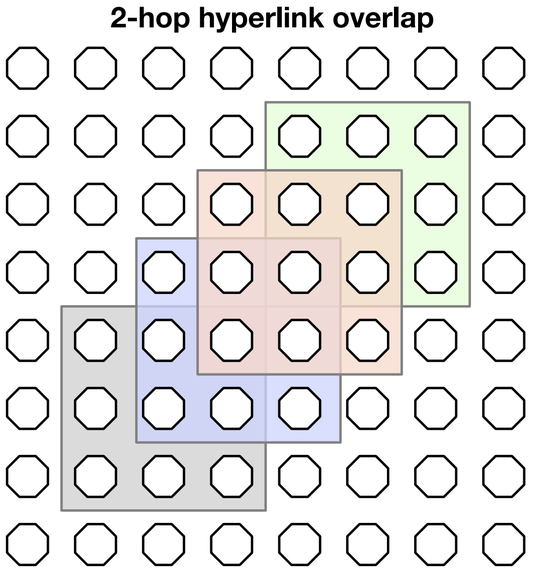
Local topology awareness 1, 2 hops out, as basis for GVM, routing (and healing), and consensus engine. Demonstrate Raft and Paxos consensus protocols using single and multi- token Petri net. Demonstrate Reversible Constructor for 2PC, 3PC & 4PC

WOLFRAM MATHEMATICA



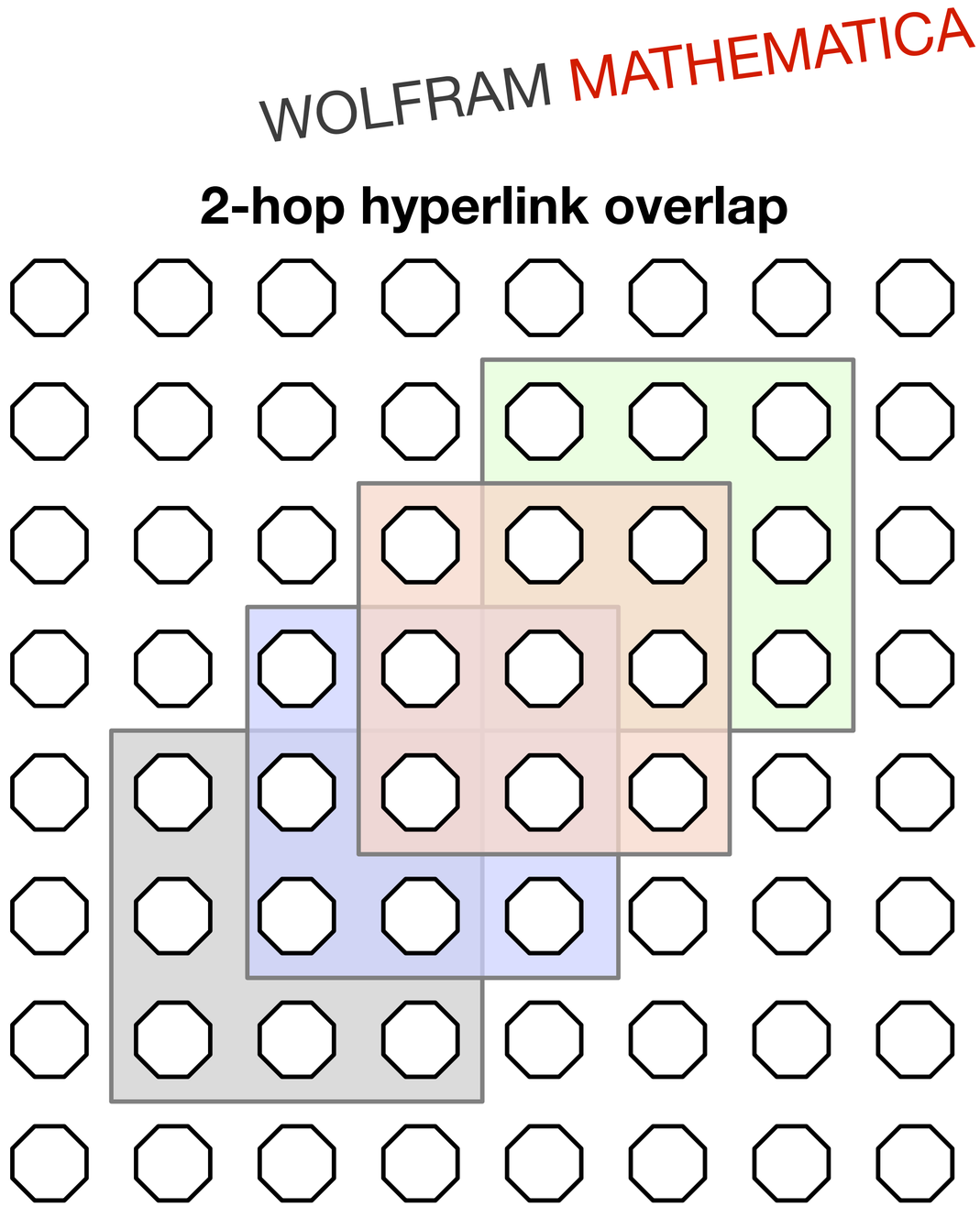


(a) 1st Hop Failure and Bypass (b) 2nd Hop Failure and Bypass (c) 3rd Hop Failure and Bypass (d) Hop Failure and Bypass

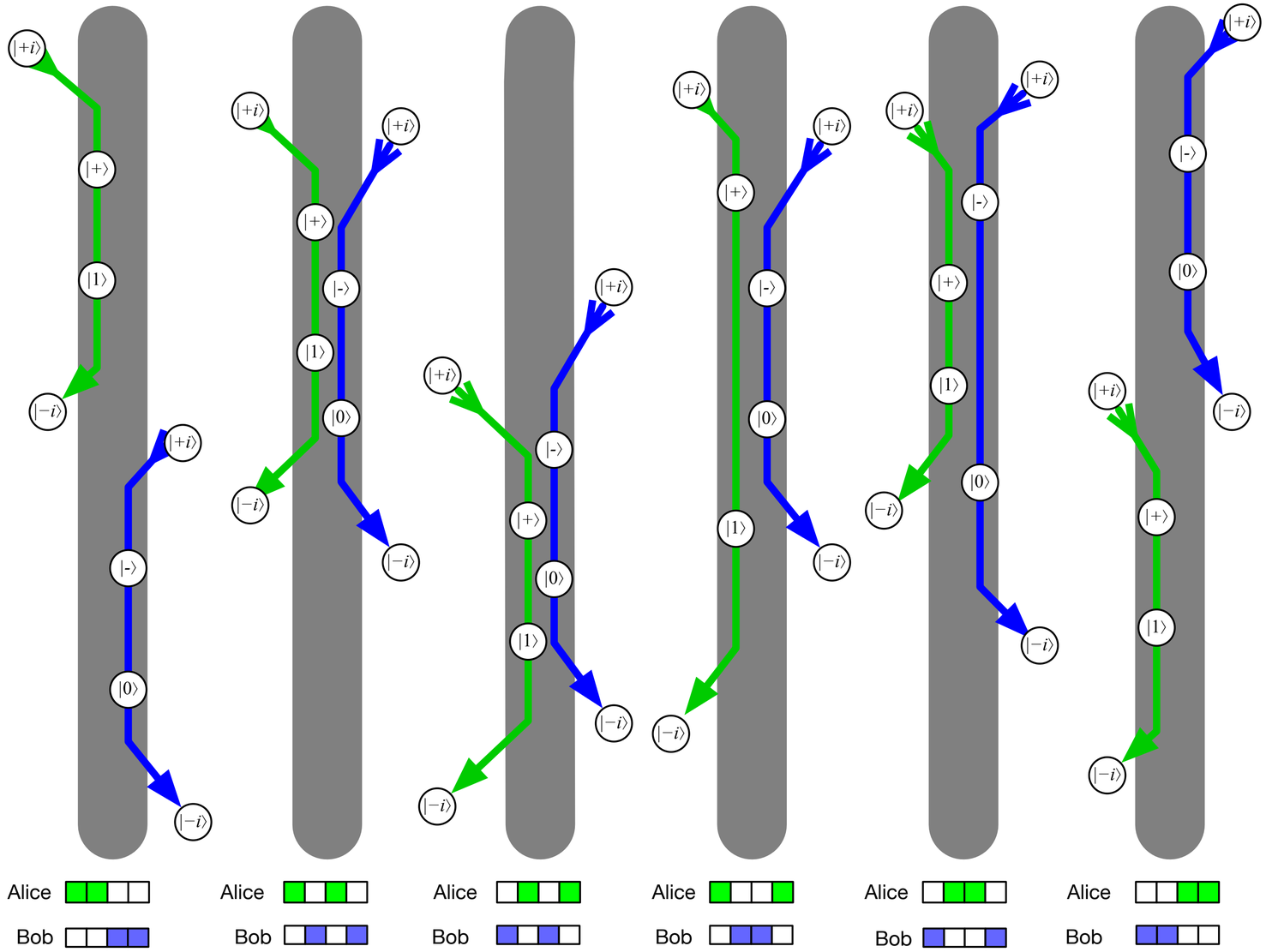


Cluster Liveness On Hypergraphs

- On the Link
- In the Cell
- In the Hypercell (8-ports)
- On the Trees
- On the Graph



Serialization in the FPGA SerDes



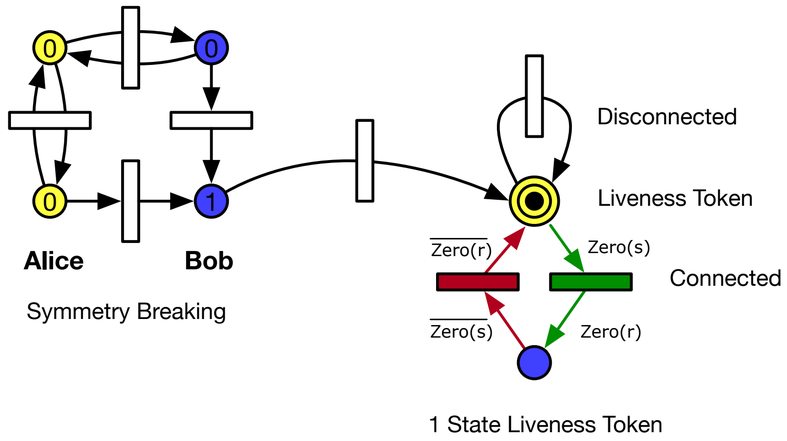
Petri-Spekkens Token Orderings in the FPGA SerDes and Disjoint Pairs

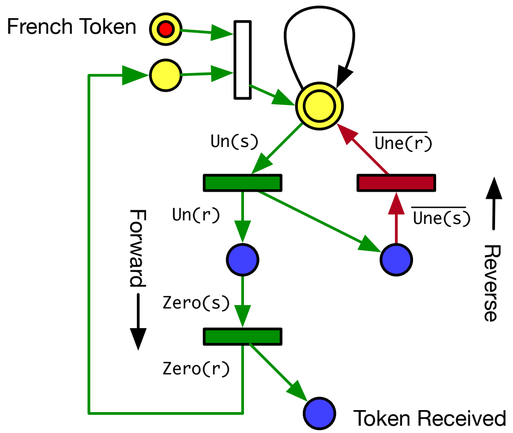
WOLFRAM MATHEMATICA

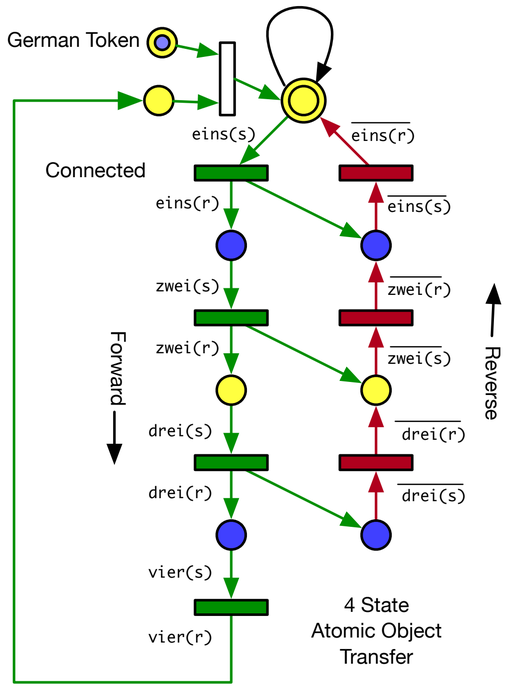
We proactively ***reorder events*** to match application constraints, eliminate race conditions, and prevent write skews before the database/application sees them as an error

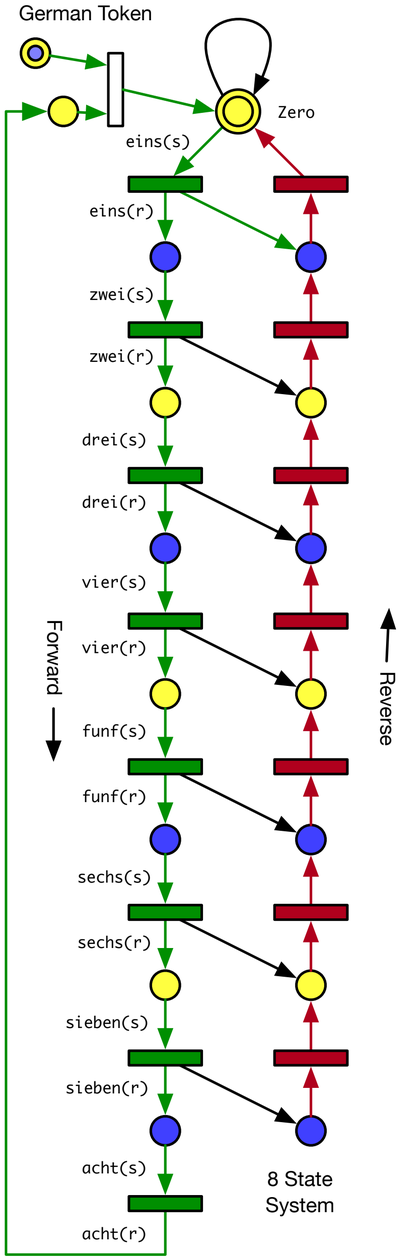
Reversible State Machines

Conserved quantities/exchanged quantities/reversal recovery/lost packets on *TRAPHs*. Include measurements such as graph resiliency, total counts for each token color. Include consistency checks to be tested after fault-injection of each kind of hazard (magnitude and phase errors in Spekkens states)









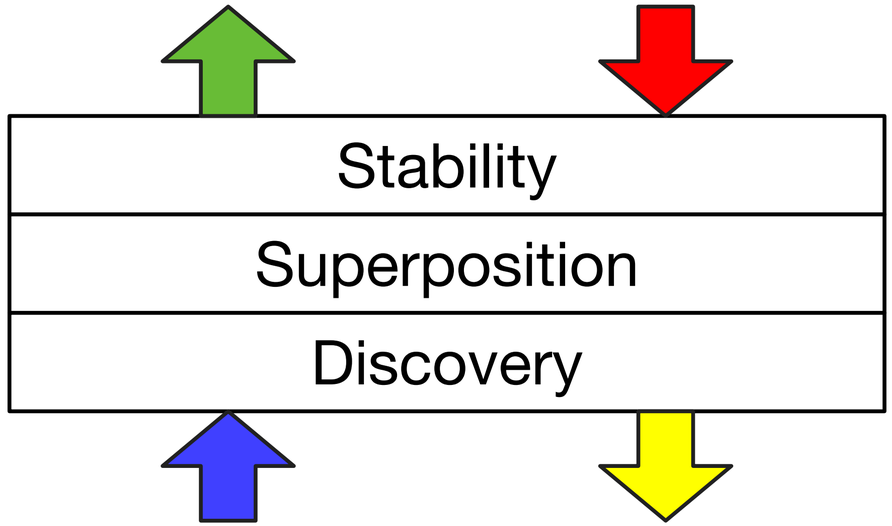
Uses Wolfram Petri-Net Package

WOLFRAM MATHEMATICA

Causal Dynamics

WOLFRAM MATHEMATICA

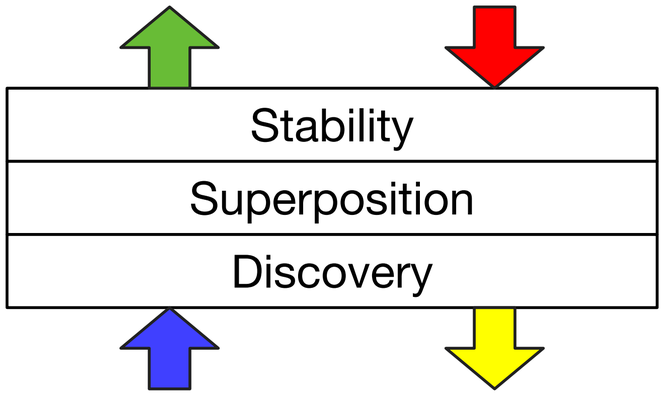
- An important goal of this design is to enable the setup, manipulation, and metastable strata of multi-level causal structures for Microservices. Jonathan Gorard was cofounder of the [Wolfram Physics Project]
- We see an opportunity to revolutionize distributed systems in datacenters everywhere, by developing multiway algorithms to predictably ‘re-order causality’ invisibly and indivisibly in the FPGA substructure (SmartNICs)
- This “solves” the vast majority of database Transaction Errors, improving reliability, performance, and security of transactions

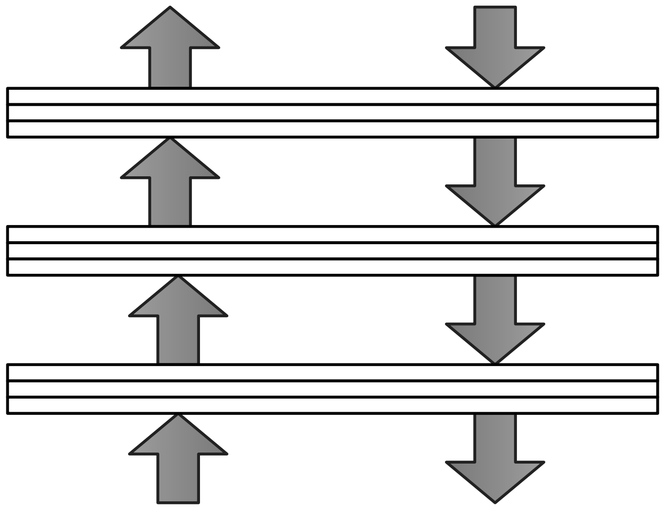


See Emily Adlam: Is There Causation in Fundamental Physics? New Insights from Process Matrices and Quantum Causal Modelling

Graph Stratification Model in Mathematica

- Demonstrate coarse graining where hypergraphs represent multiple physical nodes 3 × 3 tiles which represents cooperative computational and failover entities
- ‘Zoom out’ to whole datacenter (all nodes), or ’Zoom in’ to smaller sets of nodes in hypergraph style consolidation based on the same set of physical vertices and physical edges. Hypergraph can also ‘go down’ into the physical nodes and connect with sets of execution entities in a potentially hierarchical structure such as virtual machines, containers and lambdas
- Demonstrate ‘Kubernetes replacement’ for configuration and deployment

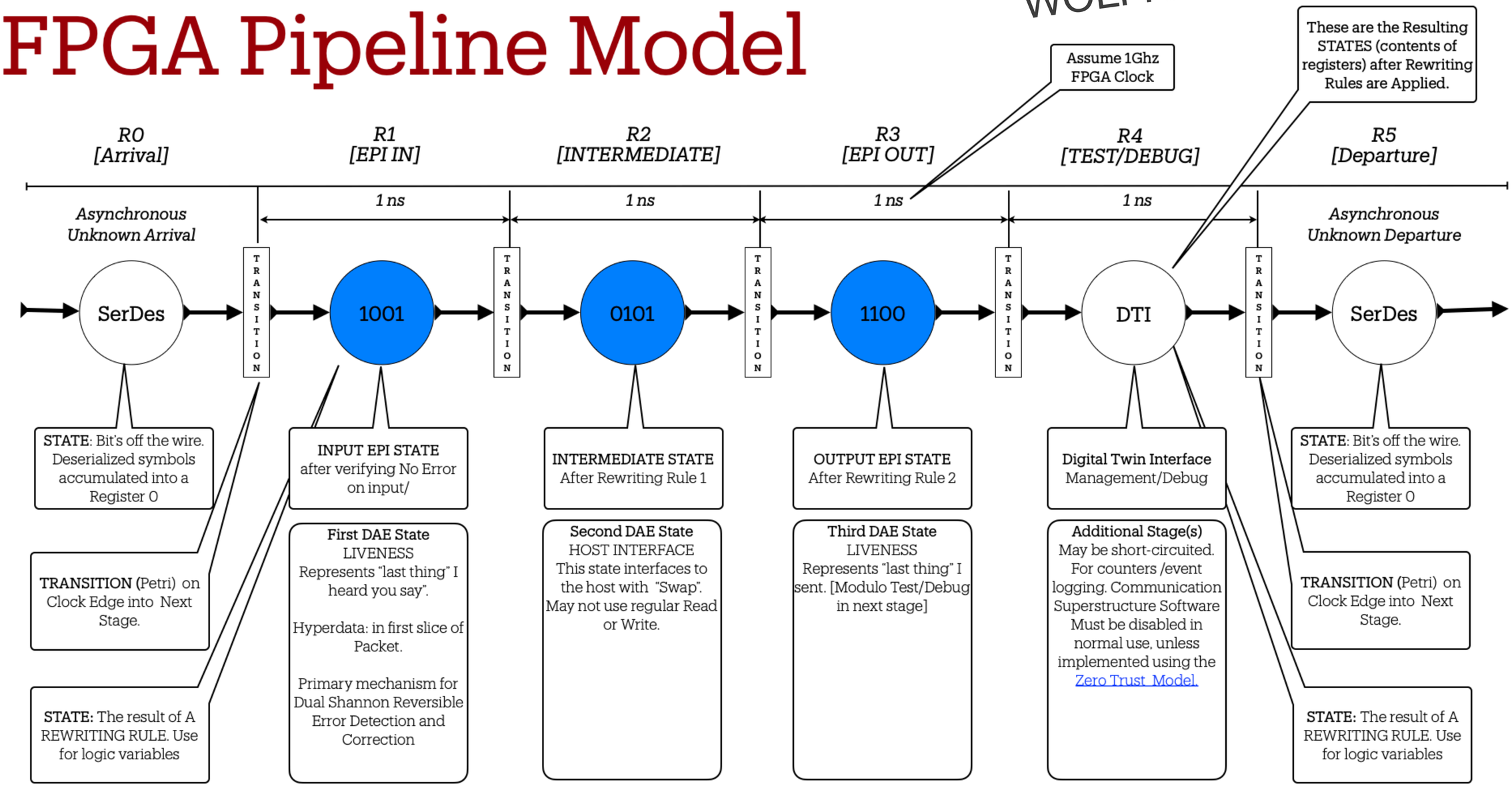




WOLFRAM MATHEMATICA

FPGA Pipeline Model

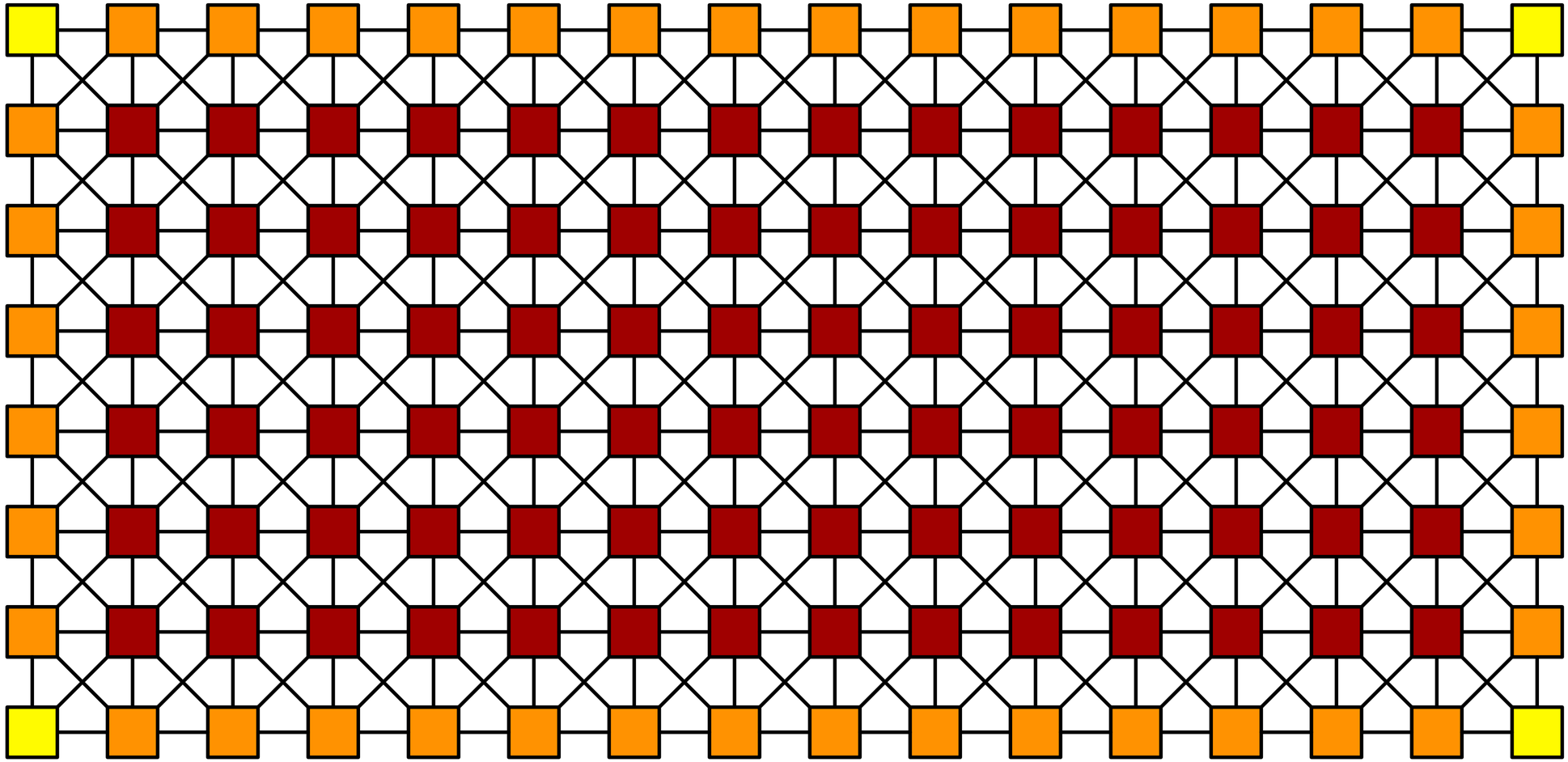
WOLFRAM MATHEMATICA



Rack & Chiplet Scale Microdatacenters

Software Infrastructure Model: Mesh Network of Microservers + Chiplet IPUs

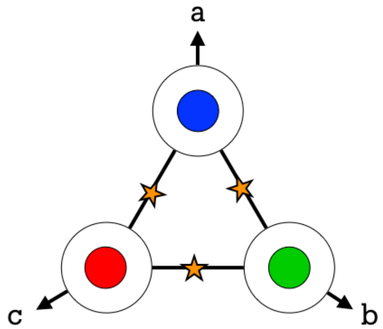
- Scale Independent, Homogeneous core of compute, memory storage and packet processing cells
- Edge rows and columns connecting to next motherboard, or next rack - for extreme bandwidth [East-West Traffic]
- Gateway to the IP world - Top Of Rack (ToR) switches [North/South Traffic]

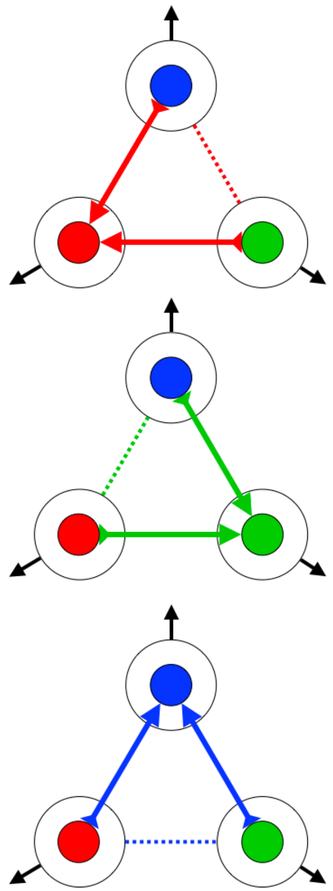


From Quantum Networks

Triangle Network

- It takes two to tango. It takes three to party
- Problem - status of each link is ‘unknown’ ★
- Use case: PTP Trees. How do you spot the “signatures of loss of synchronization”?
- Solution: Self-Healing communications trees*
- Three Trees (Blue, Green, Red) are shown here
- IEEE 1588 has 256 Trees





* TO DO - get References from Swiss Authors

* Patented (2005)

Packet Loss Model

WOLFRAM MATHEMATICA

From [1] "*Our long-term goal is to not only implement a link-local retransmission scheme that can fully recover from packet corruption losses, but one that also preserves packet ordering. In this paper, we present the results of our early-stage implementation that* ***detects the packets lost due to corruption and naïvely retransmits them out-of-order.*** *Even with this basic implementation,* ***LinkGuardian is able to mitigate the impact of corruption for both throughput-sensitive flows and latency-sensitive flows.*** *We found that the key reason why out-of-order retransmission works well is that TCP has a built-in "****reordering tolerance****" i.e. a TCP sender waits until it receives triple duplicate ACKs (or SACK-equivalent) [5, 8] before considering a packet lost. Since we are able to retransmit packets fast enough, we can effectively convert the packet loss events into* reordering events, thereby preventing cwnd reduction from being triggered at the TCP sender. Our current implementation is a work in progress that achieves good performance for the common case but does not work so well beyond the 95th percentile for latency-sensitive flows. We believe that this is because we are yet to implement mechanisms to deal with (i) **consecutive losses**, (ii) **tail losses**, and (iii) **preserving packet ordering.** Moving forward, we will extend LinkGuardian so that the end hosts can be made completely oblivious to packet corruption losses. Doing so would require additional packet buffering and we argue with back-of-the-envelope calculations that there is sufficient packet buffer in existing dataplane-programmable switches to support the same."

Resources:

1. LinkGuardian - Mitigating the impact of packet corruption loss with link local retransmission.
2. Understanding and Mitigating Packet Corruption in Data Center Networks.

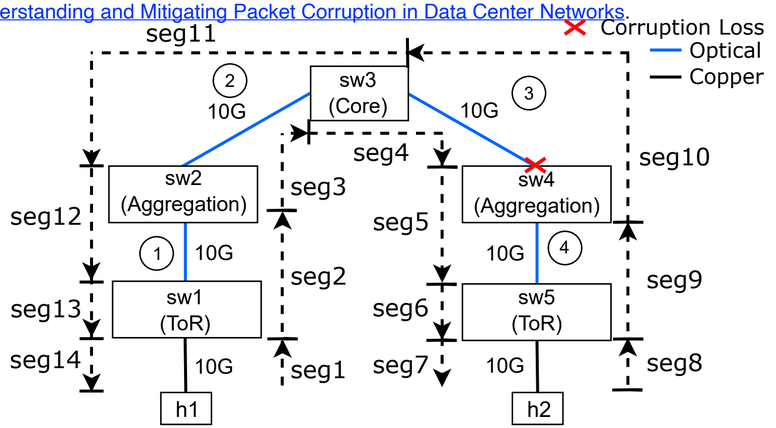


Figure 1: Testbed Setup: 3-stage Clos network with unidirectional corruption on an inter-pod path.

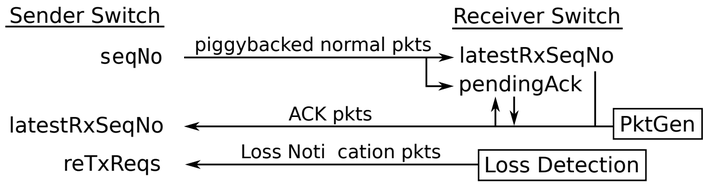


Figure 2: Protocol Overview: Arrows indicate the state variables read and updated by different packet types.

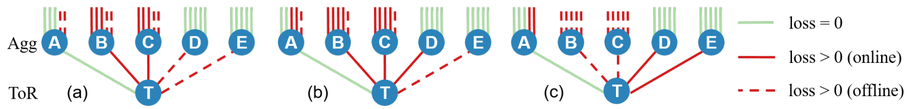


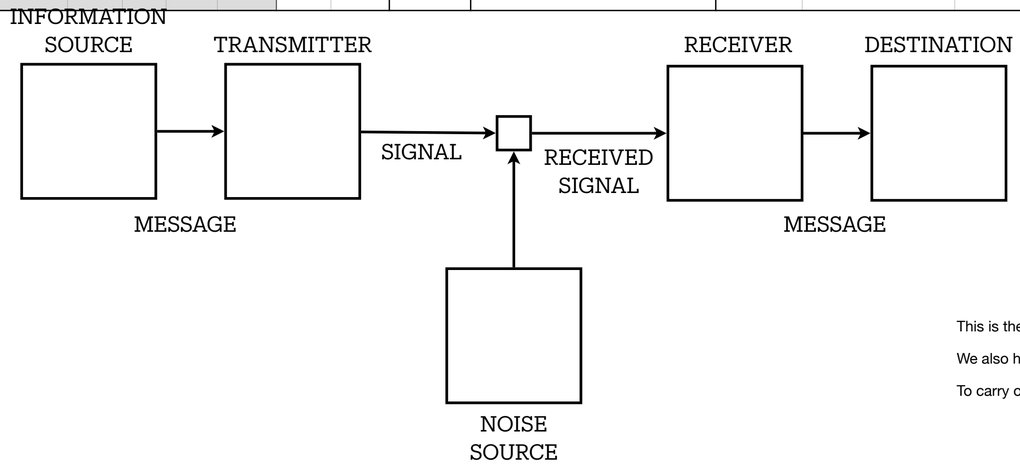
Figure 10: Example of problems with switch-local checking, with ToR capacity constraints of c=60%. (a): Every switch keeps s_c=c=60% of its uplinks alive, resulting in 8 disabled links, but only 9 out of 25 paths to the spine are still available for T, far below the constraint of 60%. (b): When s_c=$\sqrt{c}$ = 0.77 of the links are kept online, the ToR capacity constraint is met, but only 4 links can be disabled. (c): The optimal solution, which has 12 disabled links offline and meets the capacity constraints.

Dual Shannon 1

7. Microsoft/uChicago /Microsoft [If at First You Dont Succeed.](#)
8. [FEC Killed the Cut-Through Switch](#) (why we don't talk about FEC anymore)
9. Link Metrics (see Packet Loss Analysis Tab)

Project #1 The Alternating Multi-bit Protocol, from ABP to the AOP

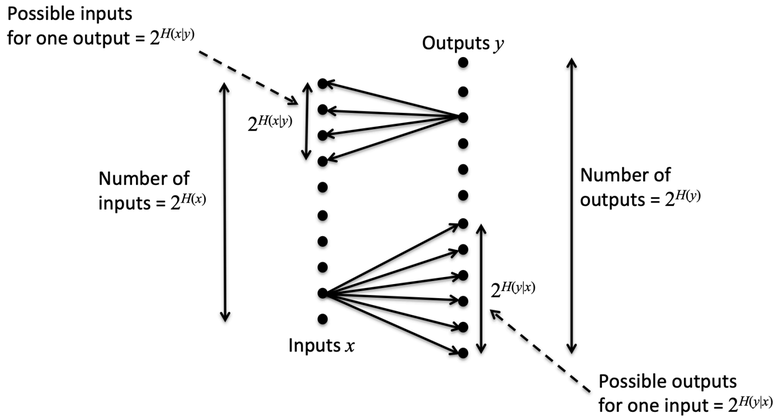
			EPISTEMIC		BITs		ONTIC	POWER SET	ALICE	ALICE	ALICE	ALICE
			From	To	ALICE	BOB	PRODUCT	Variations the Ontic State can take	2 Bits	2 Bits	2 Bits	2 Bits
PROTOCOL		n	0	2^n-1					From			To
Single Bit	ABP	1	0	1	1	1	1	2	0	EXCLUDED MIDDLE		1
Dual Bit	ADP	2	0	3	2	2	4	16	00	01	10	11
Quad Bit	AQP	4	0	15	4	4	16	65536	0000	INCLUDED MIDDLE		1111
Oct Bit	AOP	8	0	255	8	8	64	1.84467440737096E+19	00000000	INCLUDED MIDDLE		11111111



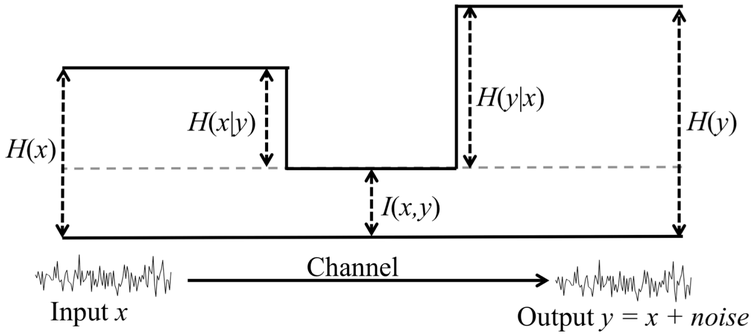
This is the Forward Channel.

We also have a feedback channel

To carry our mutual information



$$I(x,y) \quad = \quad H(y) - H(\eta) \text{ bits.}$$



Clocks and Mutual Information

The Alternating Bit Protocol [Wikipedia, Do your own Web search on "Alternating Bit Protocol"]

A single bit {0,1} forms the Alternating Bit Protocol (ABP) that initiated the first slice gets lost after the 3rd iteration.

Alice could send a 1, and Bob 'resets' it (turns it into a zero). We call this "smash and restart" because whatever value we begin with, it ends up being reset: the information is 'lost' (erased, causing energy dissipation). This is not our protocol. This is just a model to assemble to tools to do the project.

To build some kind of clock, both sides need to do some counting. Whatever size (number of bits) we use for the counter, it will overflow and all the ones flip back to being zero. Another smash and restart, but at least we can kick the can down the road and put off the reset to some point in the future where we don't care about it anymore. This is what we tried to do with Unix time and various forms of logical time used in simulators. But whether our smallest bit represents a second or a femto second, we are still sweeping the problem under the rug.

This description is about counting and how it can be used for sequence operations (including transactions) and how they can be misused to form problematic mechanisms that relate to 'time'.

Two bits is the smallest possible counter in which a 'direction' of counting is visible. It cycles one way around an ordered set, S {00, 01, 10, 11}, or if we choose the reverse direction: {11, 10, 01, 00}. The power set is $2^4 * 16$, possible alternatives to order the set.

The first thing we notice about 2 bits is that, similar to the 1 bit scenario, there appears to be no provision to express negative numbers. Adding an

EXTREMELY IMPORTANT (From JV STONE)

"Because we want to explore channel capacity in terms of channel noise, we will pretend to reverse the direction of data along the channel. Accordingly, before we 'receive' an input value, we know that the output can be one of 6 values, so our uncertainty about the input value is summarised by its entropy H(y) = 2.58 bits."

...

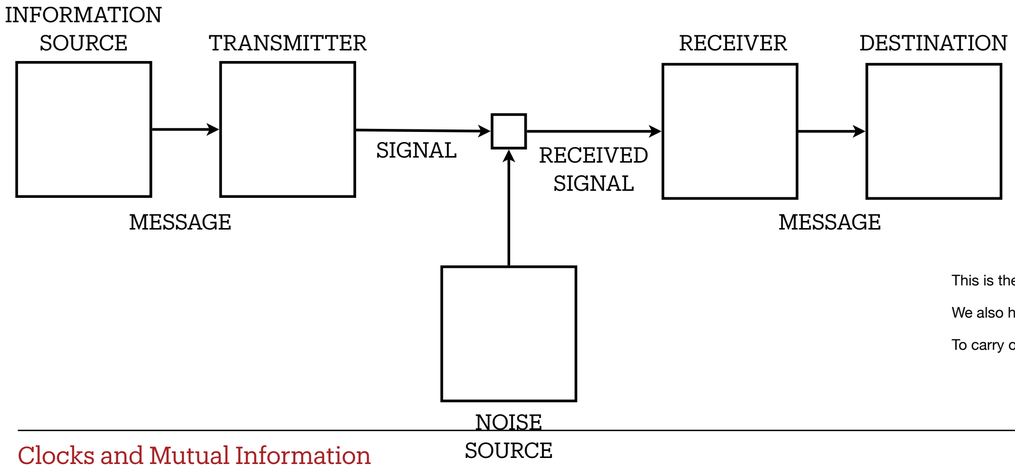
"Somewhat counter-intuitively, the average amount of

Dual Shannon 2

7. Microsoft/uChicago /Microsoft If at First You Dont Succeed.
8. FEC Killed the Cut-Through Switch (why we don't talk about FEC anymore)
9. Link Metrics (see Packet Loss Analysis Tab)

Project #1 The Alternating Multi-bit Protocol, from ABP to the AOP

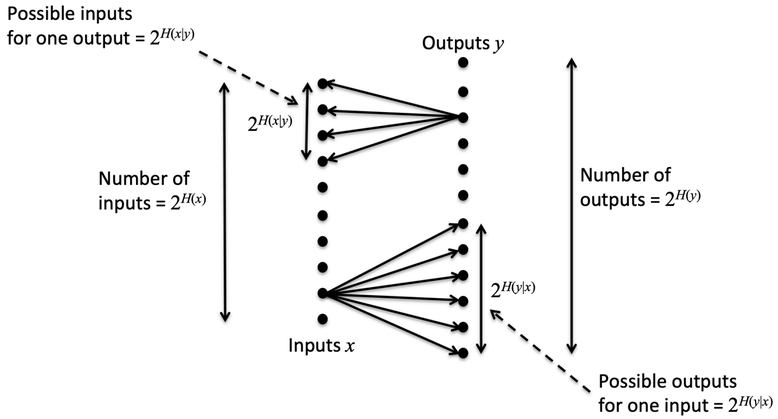
			EPISTEMIC		BITs		ONTIC	POWER SET	ALICE	ALICE	ALICE	ALICE
			From	To	ALICE	BOB	PRODUCT	Variations the Ontic State can take	2 Bits	2 Bits	2 Bits	2 Bits
PROTOCOL		n	0	2^n-1					From			To
Single Bit	ABP	1	0	1	1	1	1	2	0	EXCLUDED MIDDLE		1
Dual Bit	ADP	2	0	3	2	2	4	16	00	01	10	11
Quad Bit	AQP	4	0	15	4	4	16	65536	0000	INCLUDED MIDDLE		1111
Oct Bit	AOP	8	0	255	8	8	64	1.84467440737096E+19	00000000	INCLUDED MIDDLE		11111111



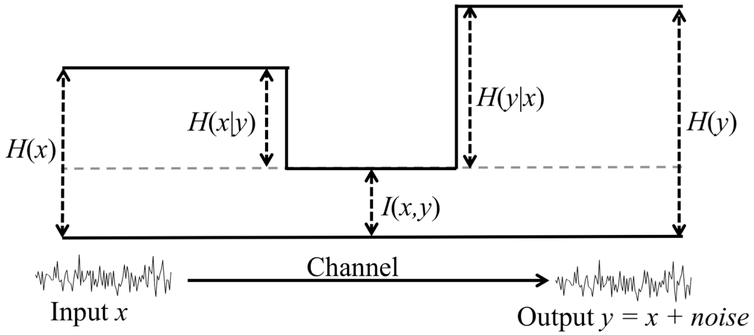
This is the Forward Channel.

We also have a feedback channel

To carry our mutual information



$$I(x,y) \quad = \quad H(y) - H(\eta) \text{ bits.}$$



Clocks and Mutual Information

The Alternating Bit Protocol [Wikipedia, Do your own Web search on "Alternating Bit Protocol"]

A single bit {0,1} forms the Alternating Bit Protocol (ABP) that initiated the first slice gets lost after the 3rd iteration.

Alice could send a 1, and Bob 'resets' it (turns it into a zero). We call this "smash and restart" because whatever value we begin with, it ends up being reset: the information is 'lost' (erased, causing energy dissipation). This is not our protocol. This is just a model to assemble to tools to do the project.

To build some kind of clock, both sides need to do some counting. Whatever size (number of bits) we use for the counter, it will overflow and all the ones flip back to being zero. Another smash and restart, but at least we can kick the can down the road and put off the reset to some point in the future where we don't care about it anymore. This is what we tried to do with Unix time and various forms of logical time used in simulators. But whether our smallest bit represents a second or a femto second, we are still sweeping the problem under the rug.

This description is about counting and how it can be used for sequence operations (including transactions) and how they can be misused to form problematic mechanisms that relate to 'time'.

Two bits is the smallest possible counter in which a 'direction' of counting is visible. It cycles one way around an ordered set, S {00, 01, 10, 11}, or if we choose the reverse direction: {11, 10, 01, 00}. The power set is $2^4 * 16$, possible alternatives to order the set.

EXTREMELY IMPORTANT (From JV STONE)

"Because we want to explore channel capacity in terms of channel noise, we will pretend to reverse the direction of data along the channel. Accordingly, before we 'receive' an input value, we know that the output can be one of 6 values, so our uncertainty about the input value is summarised by its entropy H(y) = 2.58 bits."

...

"Somewhat counter-intuitively, the average amount of

Team & Advisors

Daedaelus: Architecting the Future with Enduring Excellence

We address fundamental problems in distributed systems using protocols, data structures and algorithms inspired by Quantum Information Theory and Multiway Systems.
Our market is secure, reliable, distributed computing on the edge. Use-cases: Transaction Processing, Infrastructure Failover, Digital Twins, and Interface to Quantum Computers

Team

- Founder/CEO: Paul Borrill (NASA, Sun Microsystems, Quantum, VERITAS Software, Apple)
- Networking Story: Steve Chalmers (HPE)
- Protocol Engineering: Sahas Mulamala (AWS, Daedaelus)
- Graph Models : Dugan Hammock (Wolfram Institute, Mathematician, Artist)
- Simulation and Visualization: Casildo Romero (Physicist, Programmer, LANL)
- CFO: Roger Marlin

Current & Past Advisors

- Business Development: Chuck Sobey (ChannelScience)
- Jim Bole (Oracle EVP of Products)
- Database Transactions: Charlie Johnson (Tandem/HP NonStop)
- Mathematical Physics: Jonathan Gorard (Cambridge,Wolfram,IAS)

Open Atomic Ethernet and WOLFRAM MATHEMATICA

We would like to promote the use of Mathematica for all the specifications for the new OAE Standards Project.
Starting with:
1. Metcalfe+Boggs Original 1976 Paper.
2. Briggs 1978 Paper that defines Exactly once semantics and the alternating bit protocol

OPEN Compute Project®

About · Marketplace · Solution Providers · Contributions · Projects · Events · Membership

Open Atomic Ethernet

Projects > Time Appliances Project (TAP) > Open Atomic Ethernet

Project Leads
Sahas Munamala, Paul Borrill

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.

Open Atomic Ethernet will look ahead to the next 50 years of Ethernet networking. It will focus on ways to achieve higher performance, guaranteed service, lower latency, fewer errors, and greater scalability.

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.

Details

Wiki

Get Involved

Sahas Munamala
Paul Borrill

subscribe to mailing list

Events

2025 OCP EMEA Summit
2025 OCP Global Summit