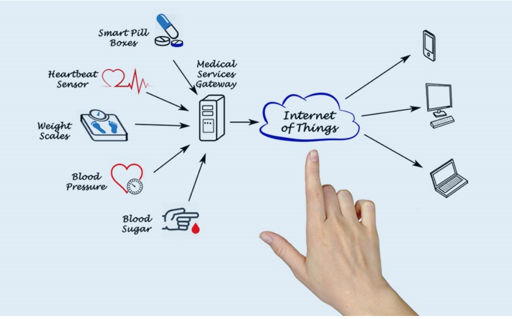
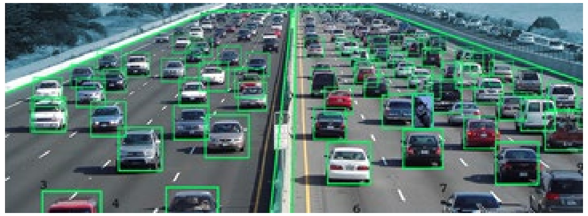
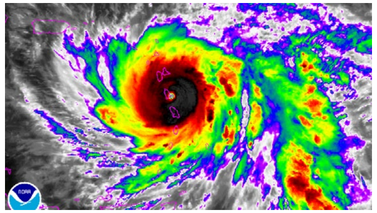
DERECHO IS A PLATFORM TO ENABLE MACHINE INTELLIGENCE FOR THE "INTERNET OF THINGS"



IoT devices simply don't have enough power and lack the big picture.

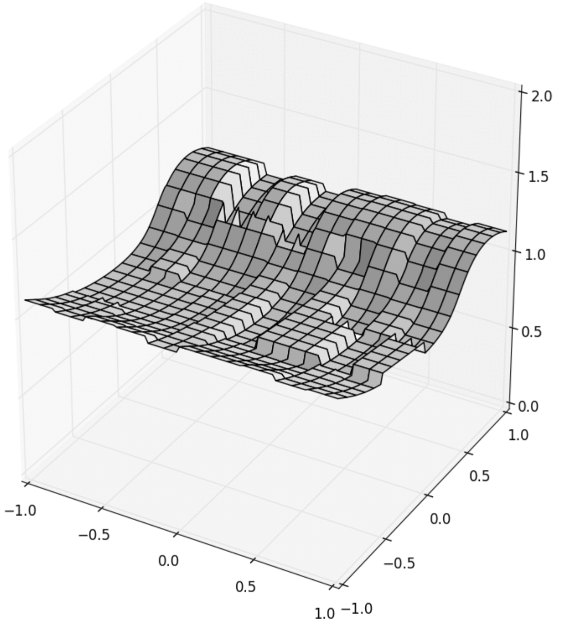
Use the cloud-edge could host machine intelligence, enabling real-time reactivity using consistent, recently-acquired context.



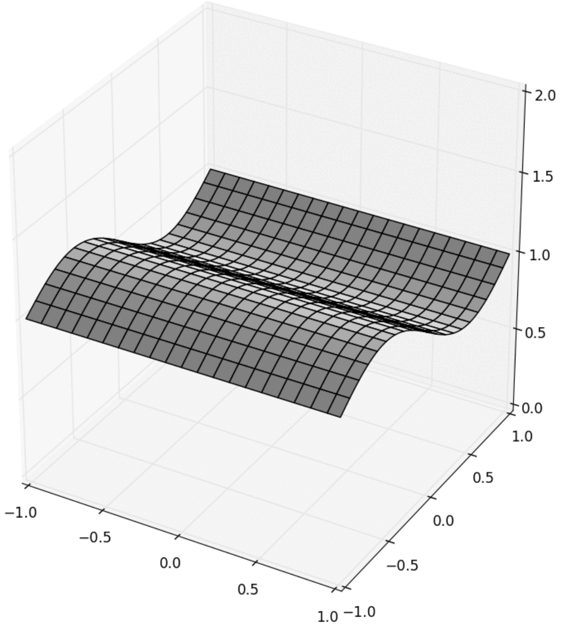


REQUIRES SPEED + CONSISTENCY

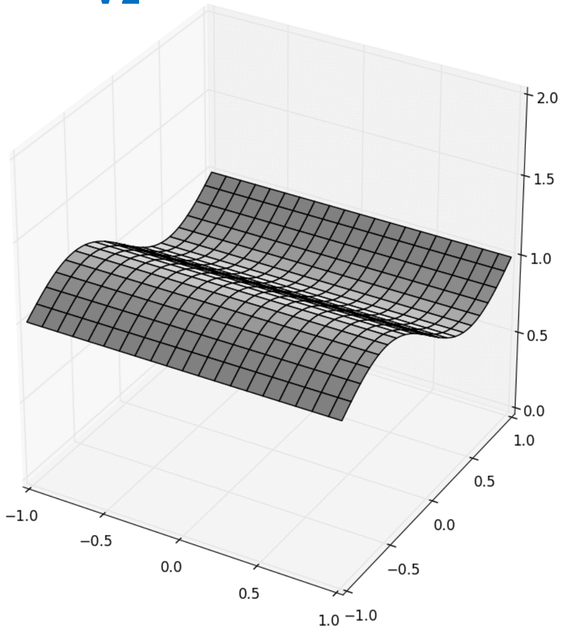
HDFS



FFFS$_{V2}$+SERVER TIME



FFFS$_{V2}$+SENSOR TIME



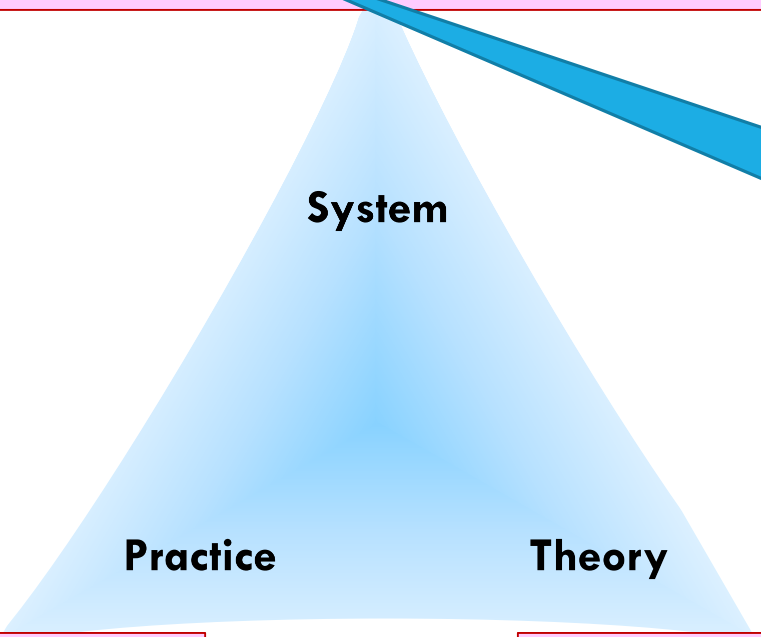
For real-time IoT data, the Derecho-based storage service (FFFS$_{v2}$) offers optimal temporal accuracy and strong read consistency, lock free.

DERECHO: UNDERLYING PLATFORM

A Derecho

New RDMA software framework for distributed programming (a C++ library)

100x to 10,000x faster than other options



RDMA = "Remote direct memory access".

Zookeeper, HDFS, blob store (FFFS$_{v2}$), BlockChains, DDS…

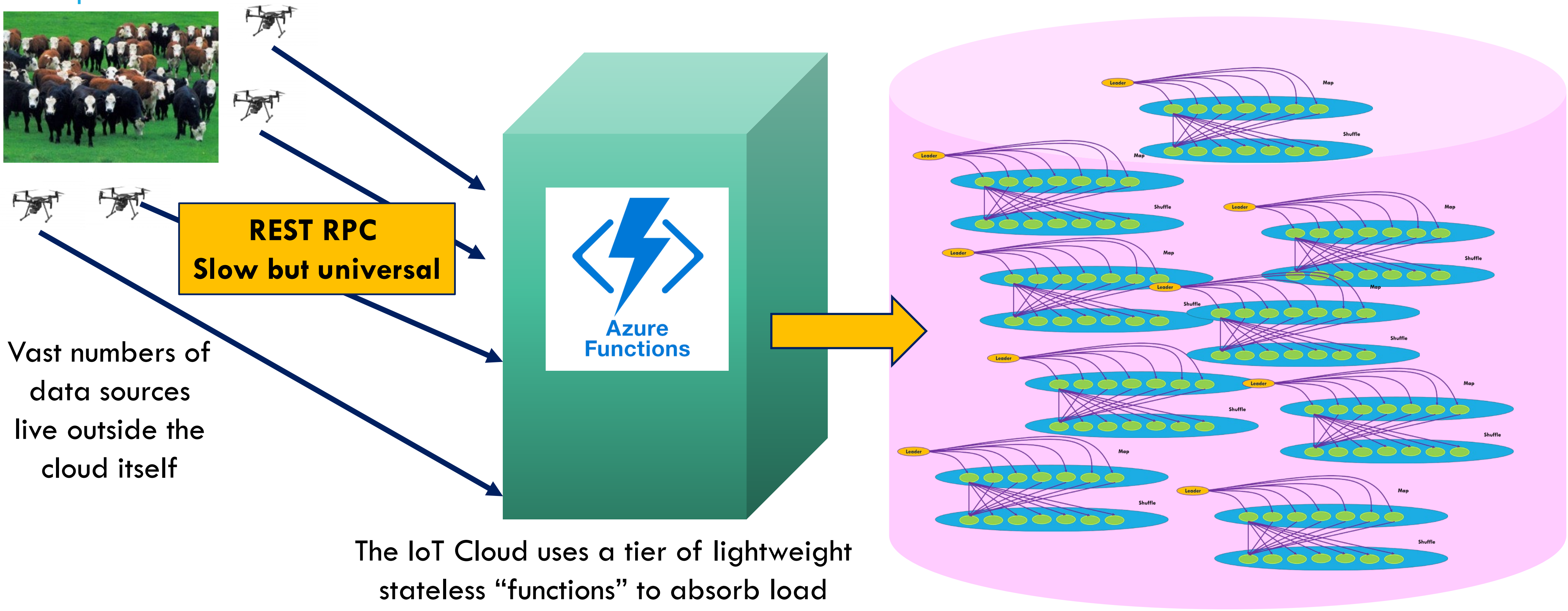
… or your spiffy new smart IoT services

It can be used to improve existing cloud μ-services

The first provably optimal Paxos/Atomic Multicast

Applying Derecho's asynchronous programming methodology to Paxos led us to the fastest possible protocol

MASSIVELY PARALLEL REAL-TIME USES



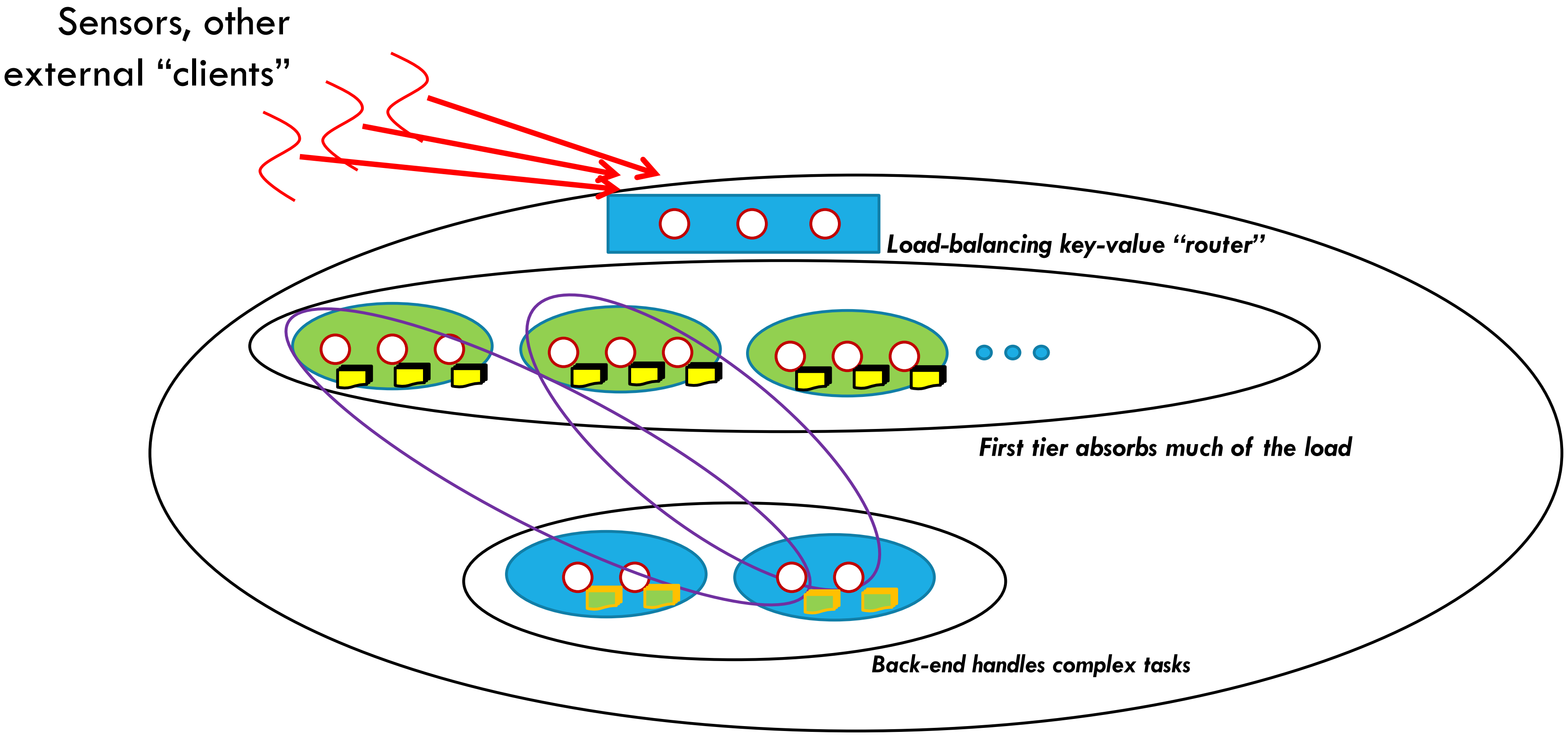
REST RPC
Slow but universal

Vast numbers of data sources live outside the cloud itself

The IoT Cloud uses a tier of lightweight stateless “functions” to absorb load

Derecho is a tool for creating intelligent stateful μ-services, like the Freeze Frame File Server, or this “MapReduce” service

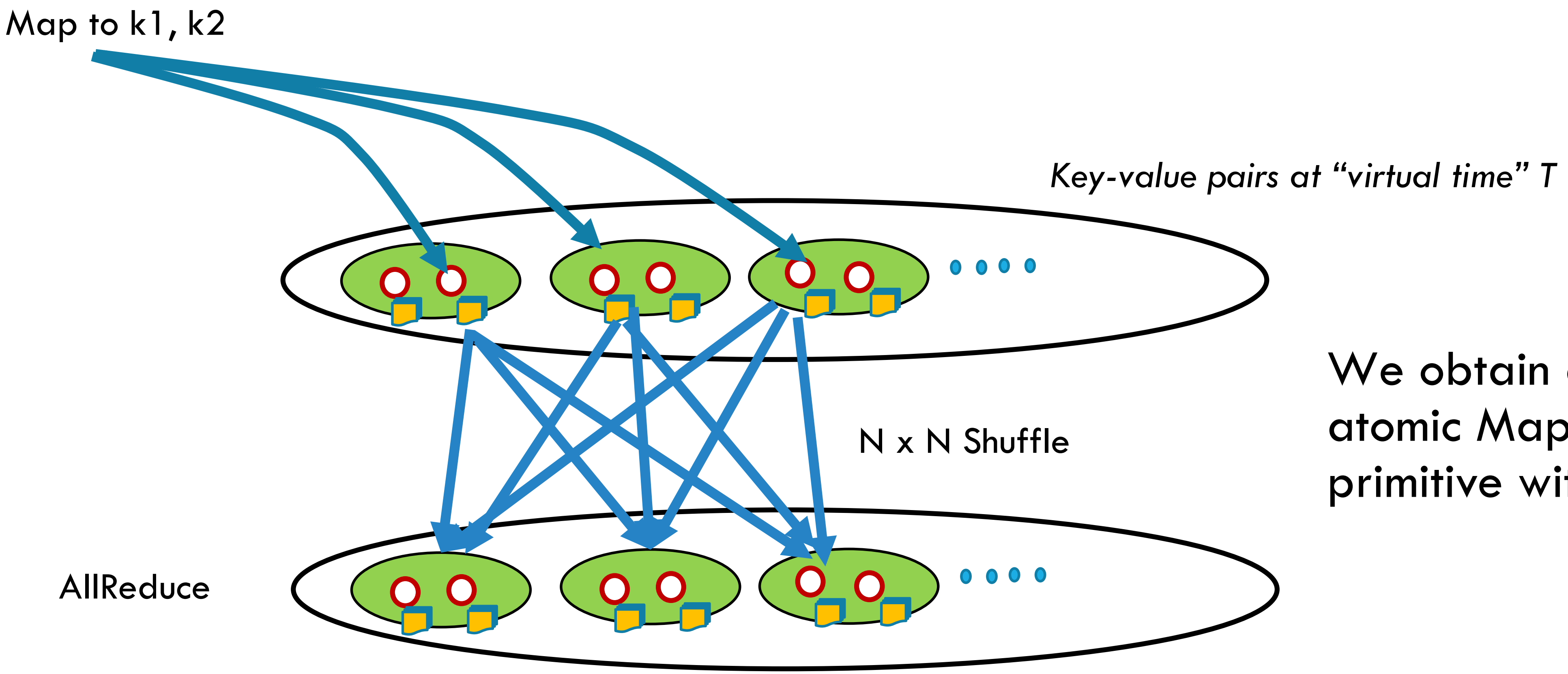
… OUR MODEL: STATE MACHINE REPLICATION IN GROUPS (ATOMIC MULTICAST OR DURABLE LOGGING)


Sensors, other external "clients"
Load-balancing key-value "router"
First tier absorbs much of the load
Back-end handles complex tasks

This is just an example.

The developer defines subgroups, controls layout and "shard" pattern

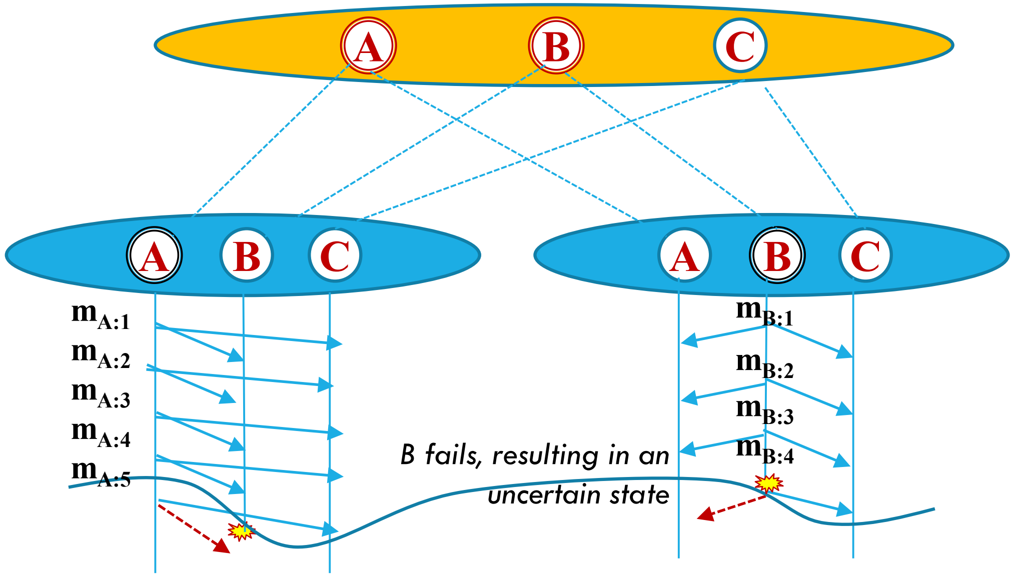
MAP-REDUCE IN A SHARDED GROUP



We obtain a completely atomic MapReduce primitive within Derecho!

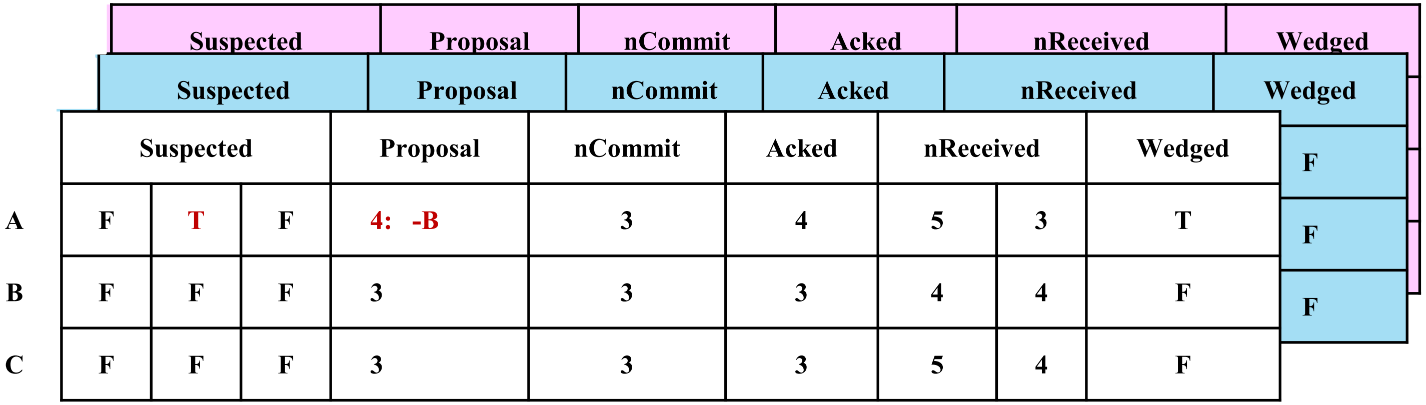
IMPLEMENTATION: DERECHO = RDMC/SMC + SST

Derecho group with members {A, B, C}
in which C is receive-only



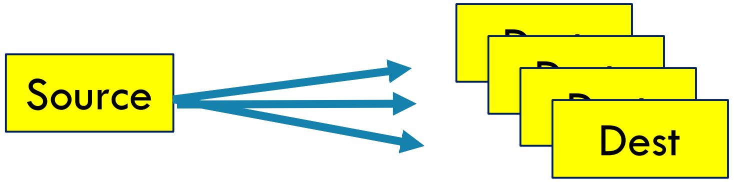
Data moved on RDMA multicast

A, B and C each have a replica of the SST



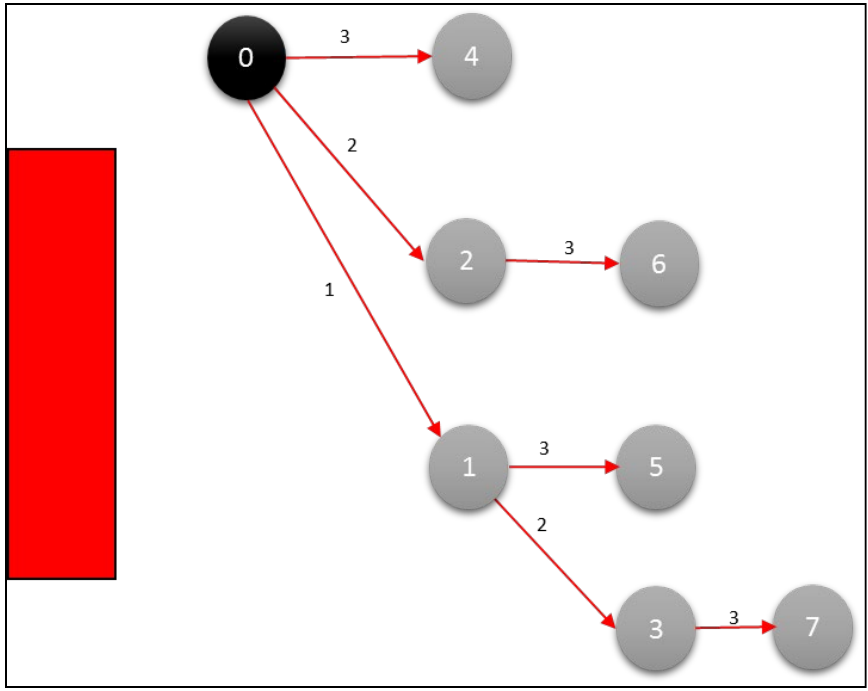
	Suspected			Proposal	nCommit	Acked	nReceived		Wedged
A	F	T	F	4: -B	3	4	5	3	T
B	F	F	F	3	3	3	4	4	F
C	F	F	F	3	3	3	5	4	F

Control is done using knowledge programming on the SST

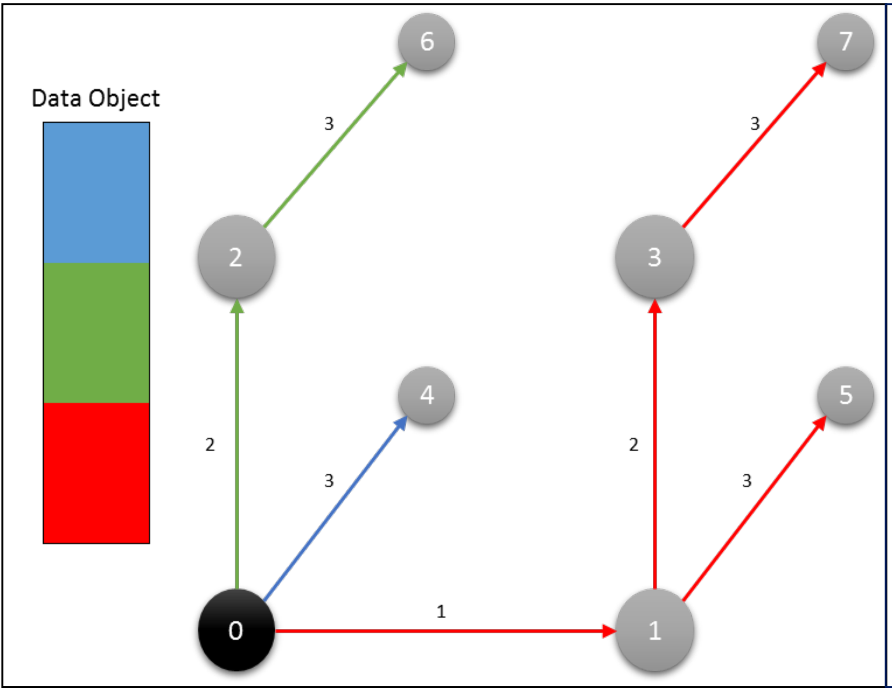


Multicast

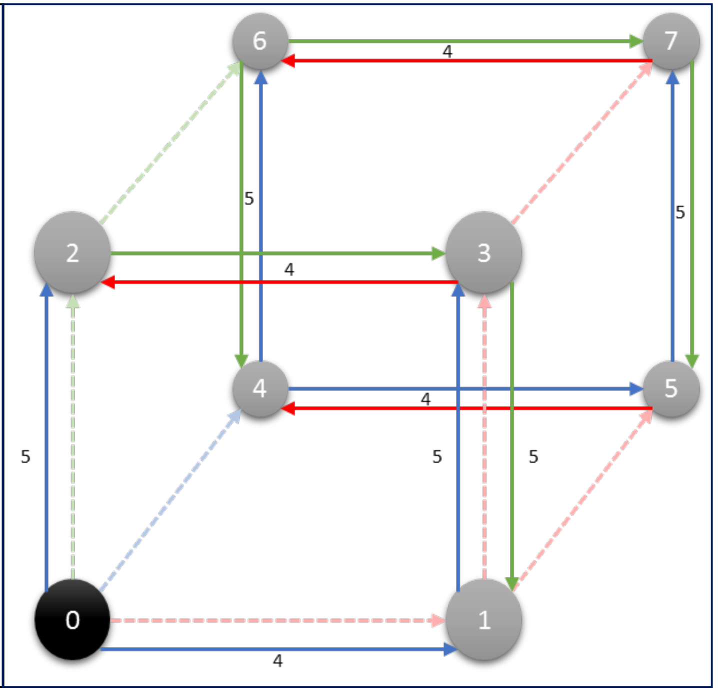
RDMC: AN RDMA MULTICAST



Binomial Tree

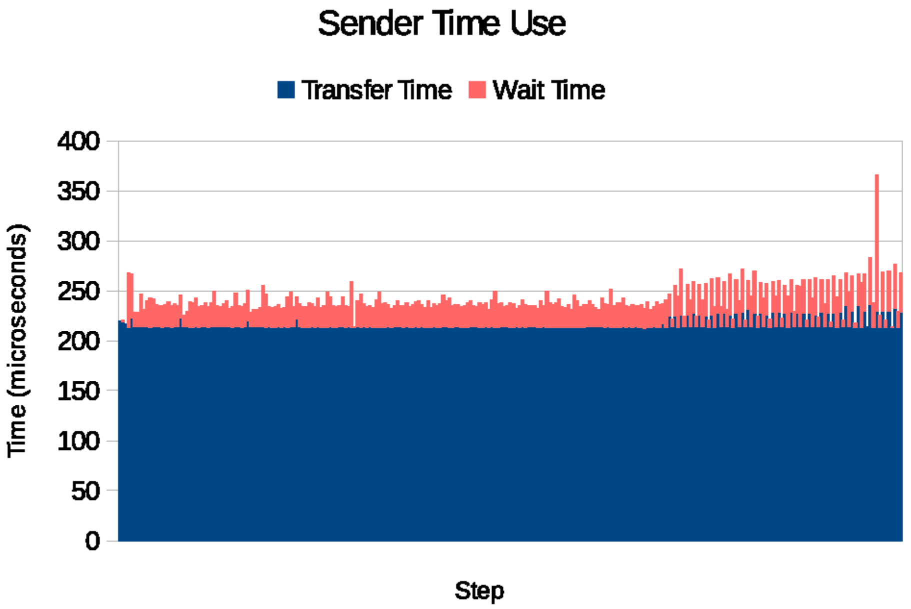


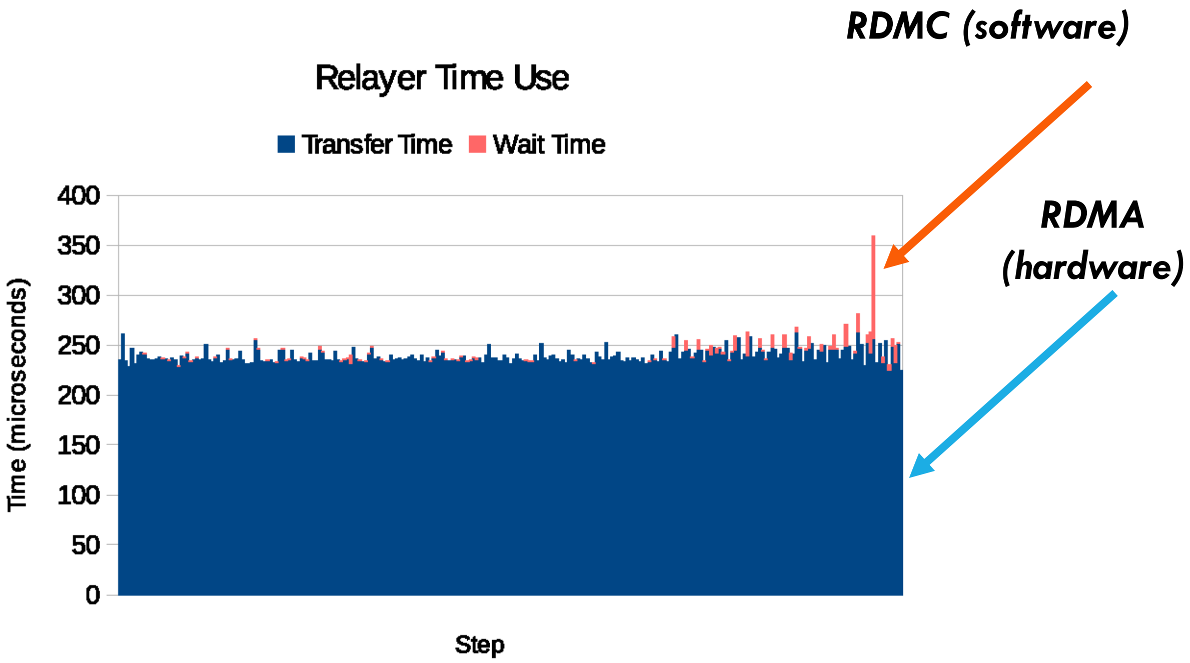
Binomial Pipeline



Final Step

RDMC SUCCEEDS IN OFFLOADING WORK TO HARDWARE





RDMC (software)

RDMA (hardware)

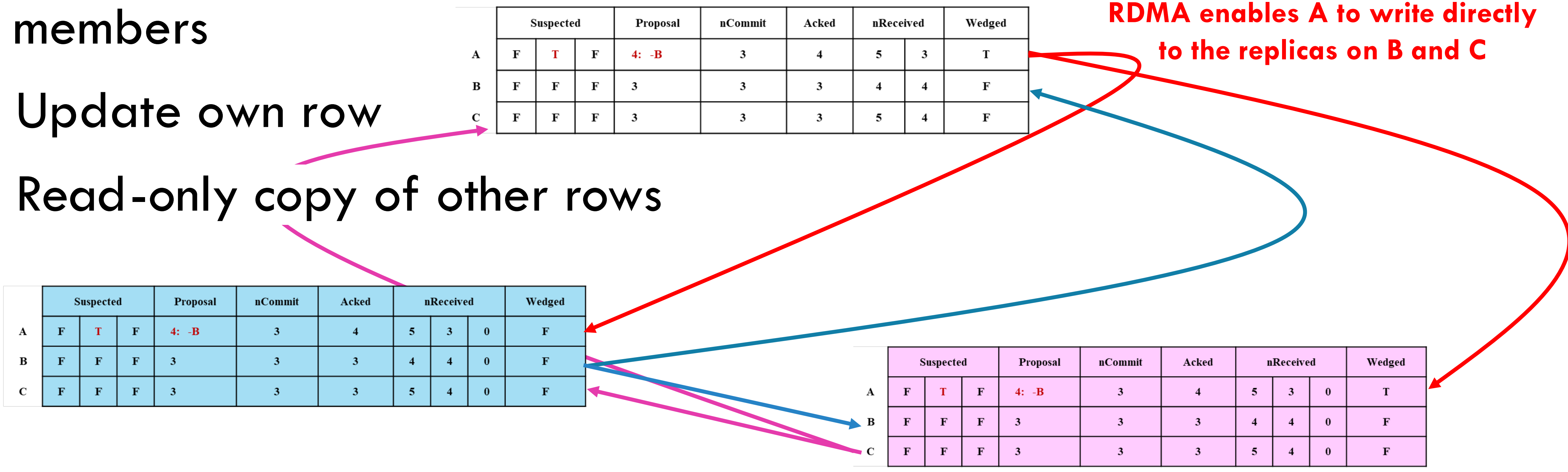
Trace a single multicast through our system… Orange is time "waiting for action by software". Blue is "RDMA data movement".

SHARED STATE TABLE: DIRECT RDMA WRITES WITH NO LOCKING (SEQUENTIAL CACHE-LINE CONSISTENCY)

Replicated at members

Update own row

Read-only copy of other rows



RDMA enables A to write directly to the replicas on B and C

	Suspected			Proposal	nCommit	Acked	nReceived		Wedged
A	F	T	F	4: -B	3	4	5	3	T
B	F	F	F	3	3	3	4	4	F
C	F	F	F	3	3	3	5	4	F

	Suspected			Proposal	nCommit	Acked	nReceived			Wedged
A	F	T	F	4: -B	3	4	5	3	0	F
B	F	F	F	3	3	3	4	4	0	F
C	F	F	F	3	3	3	5	4	0	F

	Suspected			Proposal	nCommit	Acked	nReceived			Wedged
A	F	T	F	4: -B	3	4	5	3	0	T
B	F	F	F	3	3	3	4	4	0	F
C	F	F	F	3	3	3	5	4	0	F

SST PROGRAMMING MODEL

Lock-free, but we store monotonic values in the cells. If you miss some updates you can still deduce that they occurred.

Enables monotonic aggregation and even a monotonic form of knowledge-based reasoning ($\mathrm{K}(\mathscr{P})$, $\mathrm{K}^1(\mathscr{P})$, …).

Result? Highly efficient *batched receiver-side decision-making*.

FORMAL VERIFICATION

We recently explored use of a theorem prover (Ivy, from Tel Aviv University and Microsoft, really a front-end to Z3 solver)

Formalized and proved our protocols correct.

Many open formal questions remain both related to things we have proved on paper and also to the larger question of proving properties of systems that use these tools.

DERECHO IS *FAST*

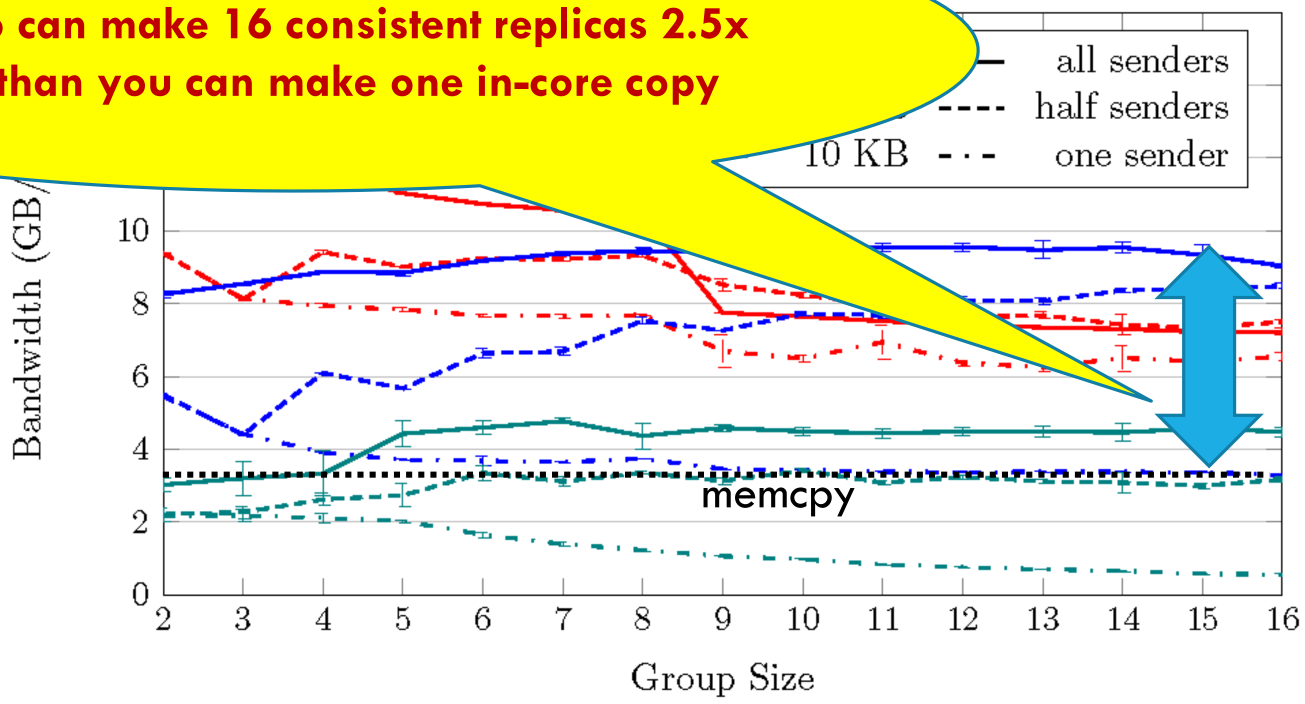
Mellanox 100Gbps RDMA on ROCE (fast Ethernet)

100Gb/s = 12.5GB/s

Comparisons:

- Derecho: 16GB/s
- Memcpy: 3.75GB/s
- Zookeeper: 0.75GB/s
- LibPaxos: 0.25GB/s

Derecho Atomic Multicast (Vertical Paxos)



Derecho can make 16 consistent replicas 2.5x faster than you can make one in-core copy

Cool discovery: Derecho outperforms *even on standard TCP.*

DERECHO BLOCKCHAIN

Edwar

Event e is locally stable if we have K_L identical logs that contain e.

It is WAN-stable if we have K_w WAN-mirror copies of those logs.

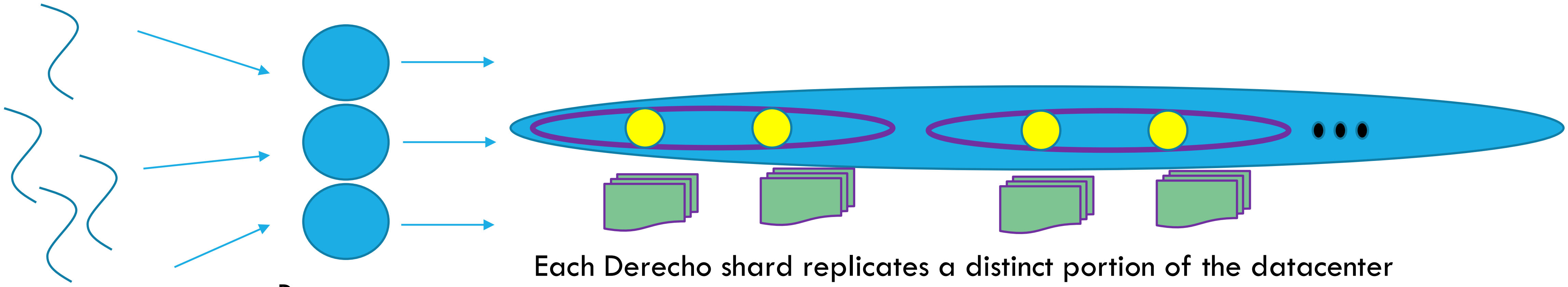
Key ic

1) Sta proof of work). It runs in one data center, which is connected to other da rs via high-speed WAN network links.

2) Add WAN mirroring [new concept: WAN bility]. Datacenter A can update its local "ledger" and has read-only mirrors of remote ledgers. This is a bit like Google Spanner, we could merge these ledgers using record timestamps (if we have true-time, that is)

3) Now provide a client API that mimics standard BlockChain, but without proof of work. Client can submit a record, and once accepted it will be visible in the global BlockChain. Client can read any W-stable records, and they will never be tampered-with

DERECHO BLOCKCHAIN HIGH-LEVEL PICTURE



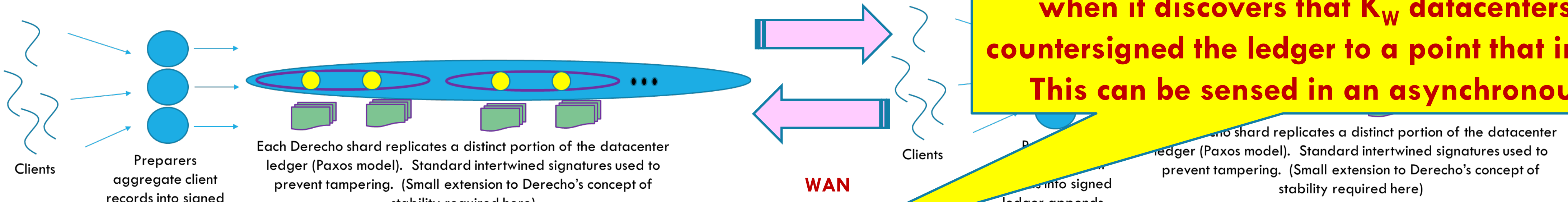
Clients

Preparers aggregate client records into signed ledger appends

Each Derecho shard replicates a distinct portion of the datacenter ledger (Paxos model). Standard intertwined signatures used to prevent tampering. (Small extension to Derecho's concept of stability required here)

DERECHO BLOCKCHAIN HIGH-LEVEL PICTURE

WAN datacenter W knows that event e is w-stable when it discovers that K_W datacenters have countersigned the ledger to a point that includes e. This can be sensed in an asynchronous way



Clients
Preparers aggregate client records into signed ledger appends
WAN Replication
W-stability counter-signature exchange
Clients
P
s into signed ledger appends

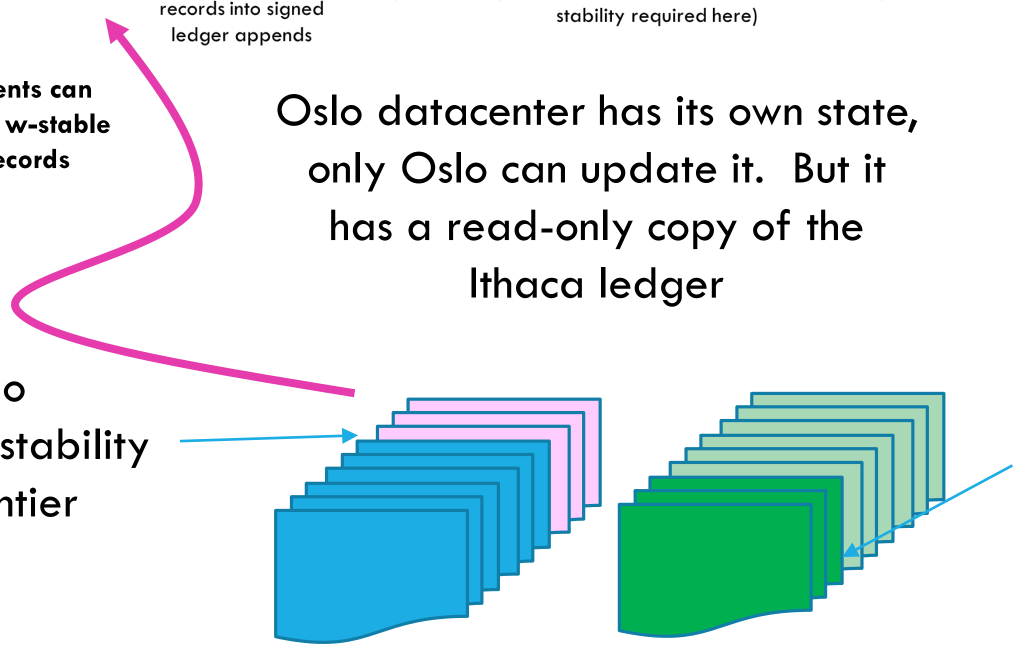
Each Derecho shard replicates a distinct portion of the datacenter ledger (Paxos model). Standard intertwined signatures used to prevent tampering. (Small extension to Derecho's concept of stability required here)

ho shard replicates a distinct portion of the datacenter edger (Paxos model). Standard intertwined signatures used to prevent tampering. (Small extension to Derecho's concept of stability required here)

Clients can read w-stable records

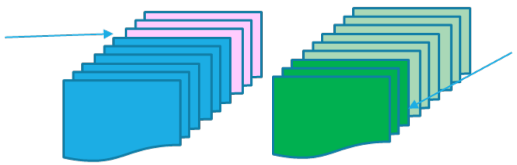
Oslo datacenter has its own state, only Oslo can update it. But it has a read-only copy of the Ithaca ledger

Ithaca datacenter has its own state, only Ithaca can update it. But it has a read-only copy of the Oslo ledger



Oslo records (r/w)

Ithaca records (r/o)



Oslo records (r/w)

Ithaca records (r/o)

DISCUSSION POINTS

How might we use Edward’s fast tamperproof audit ledgers? Can we call them “true” Blockchains, but for non-anonymous uses?

Edward and I also have a BFT differentially private distributed auditing and query algorithm. Would it be useful or is it unnecessary?

More broadly, if a small percentage of our clients are trying to cheat by deceiving the infrastructure, how can we use auditing to detect these actions (e.g. perhaps they “fool” some IoT sensors). Is there a method like double-entry bookkeeping we could explore?

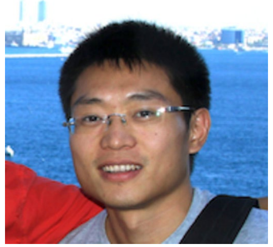
DERECHO PARTICIPANTS

Sagar Jha

Jonathan Behrens*

Matt Milano

Edward Tremel



Weijia Song

Theo Gkountouvas

Ken

Robbert

Derecho: Fast State Machine Replication for Cloud Services. S Jha, J Behrens, T Gkountouvas, M Milano, W Song, E Tremel, S Zink, K Birman, and R van Renesse. 2019. ACM Trans. Comput. Syst. (~March 2019).

RDMC: A Reliable Multicast for Large Objects. J Behrens, S Jha, K Birman, E Tremel. IEEE DSN ’18, Luxembourg, June 2018.

* Behrens was a Cornell ugrad, now at MIT pursuing his PhD