CANOPUS: A SCALABLE AND MASSIVELY PARALLEL CONSENSUS PROTOCOL

Bernard Wong
CoNEXT 2017

UNIVERSITY OF WATERLOO

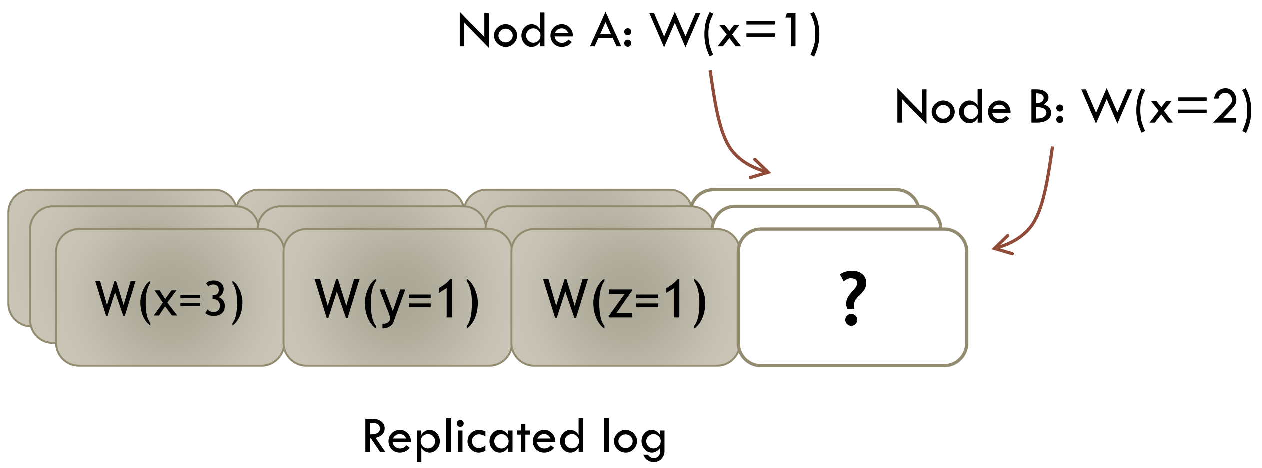
Joint work with Sajjad Rizvi and Srinivasan Keshav

CONSENSUS PROBLEM

Agreement between a set of nodes in the presence of failures

- Asynchronous environment

Primarily used to provide fault tolerance



A BUILDING BLOCK IN DISTRIBUTED SYSTEMS

A BUILDING BLOCK IN DISTRIBUTED SYSTEMS

System applications

Hadoop HBase Kafka Akka Mesos BookKeeper Spanner …

Coordination services

Current consensus protocols are not scalable

However, most applications only require a small number of replicas for fault tolerance

NetPaxos AllConcur NOPaxos …

PERMISSIONED BLOCKCHAINS

A distributed ledger shared by all the participants

Consensus at a large scale

- Large number of participants (e.g., financial institutions)
- Must validate a block before committing it to the ledger

Examples

- Hyperledger, Microsoft Coco, Kadena, Chain …

CANOPUS

Consensus among a large set of participants

- Targets thousands of nodes distributed across the globe

Decentralized protocol

- Nodes execute steps independently and in parallel

Designed for modern datacenters

- Takes advantage of high performance networks and hardware redundancies

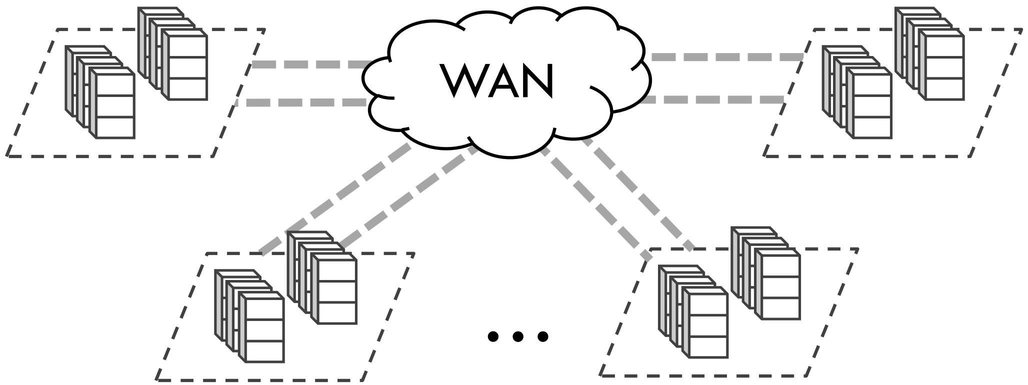
SYSTEM ASSUMPTIONS

Non-uniform network latencies and link capacities

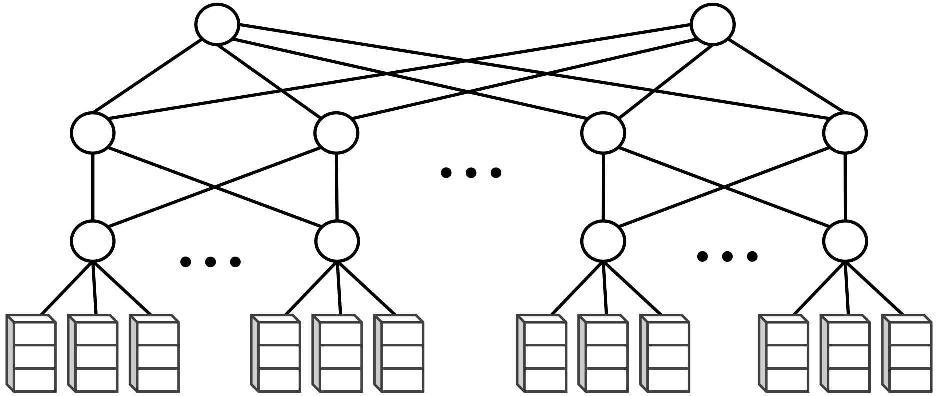
- Scalability is bandwidth limited
- Protocol must be network topology aware

Deployment consists of racks of servers connected by redundant links

- Full rack failures and network partitions are rare



Global view

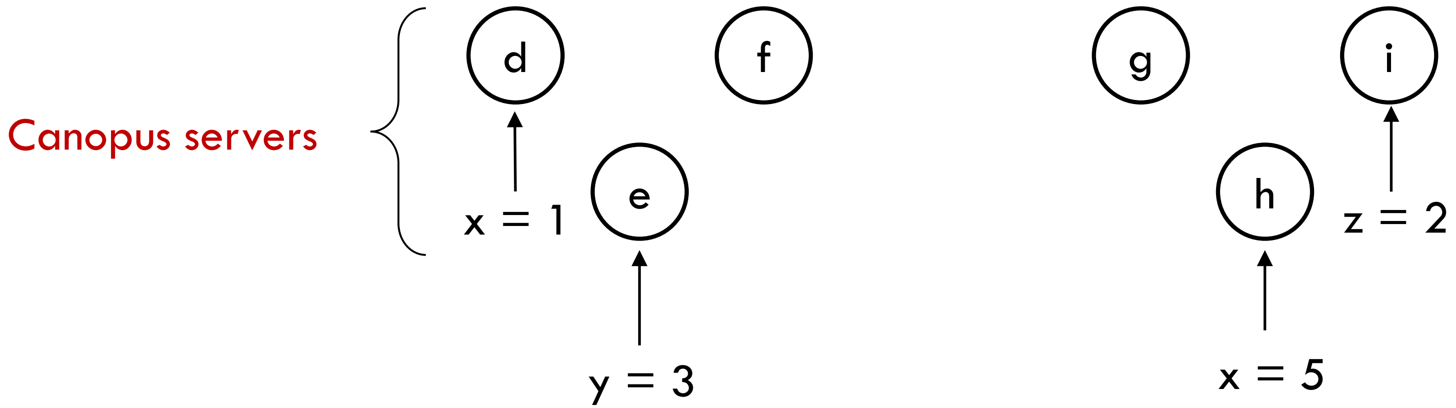


Within a datacenter

CONSENSUS CYCLES

Execution divided into a sequence of consensus cycles

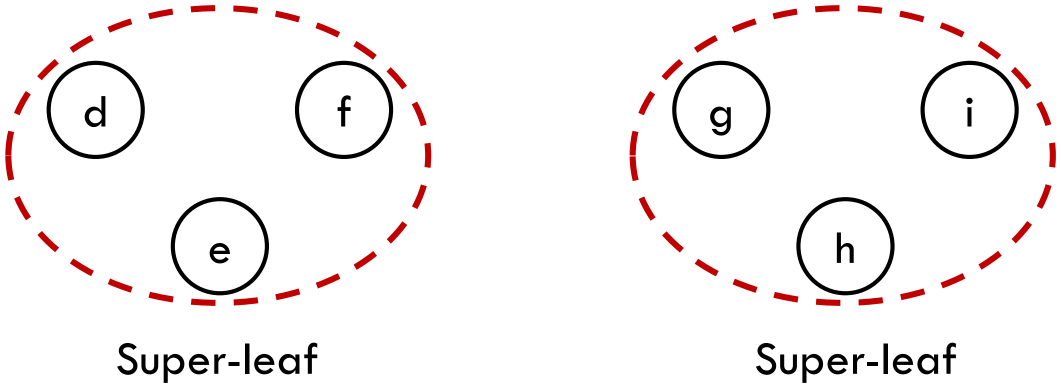
- In each cycle, Canopus determines the order of writes (state changes) received during the previous cycle



SUPER-LEAVES AND VNODES

Nodes in the same rack form a logical group called a super-leaf

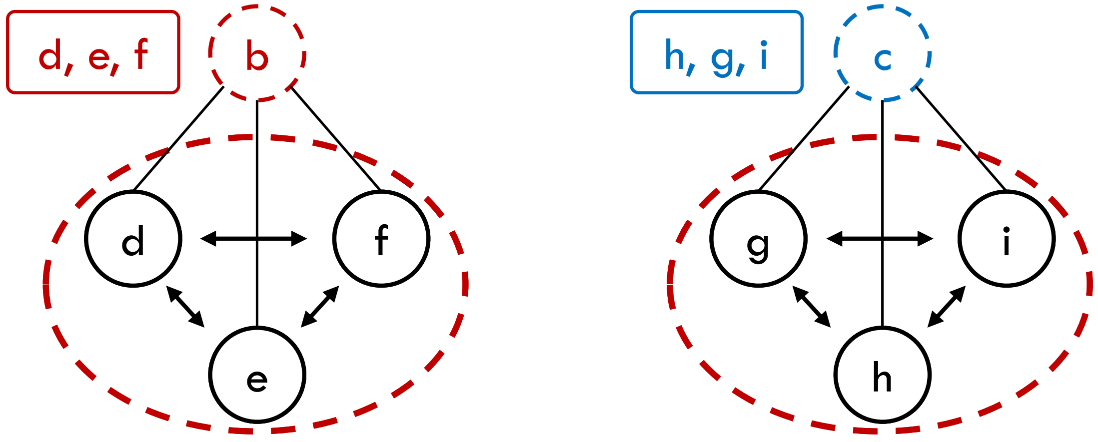
Use an intra-super-leaf consensus protocol to replicate write requests between nodes in the same super-leaf



SUPER-LEAVES AND VNODES

Nodes in the same rack form a logical group called a super-leaf

Use an intra-super-leaf consensus protocol to replicate write requests between nodes in the same super-leaf



Super-leaf

Super-leaf

Represent the state of each super-leaf as a height 1 virtual node (vnode)

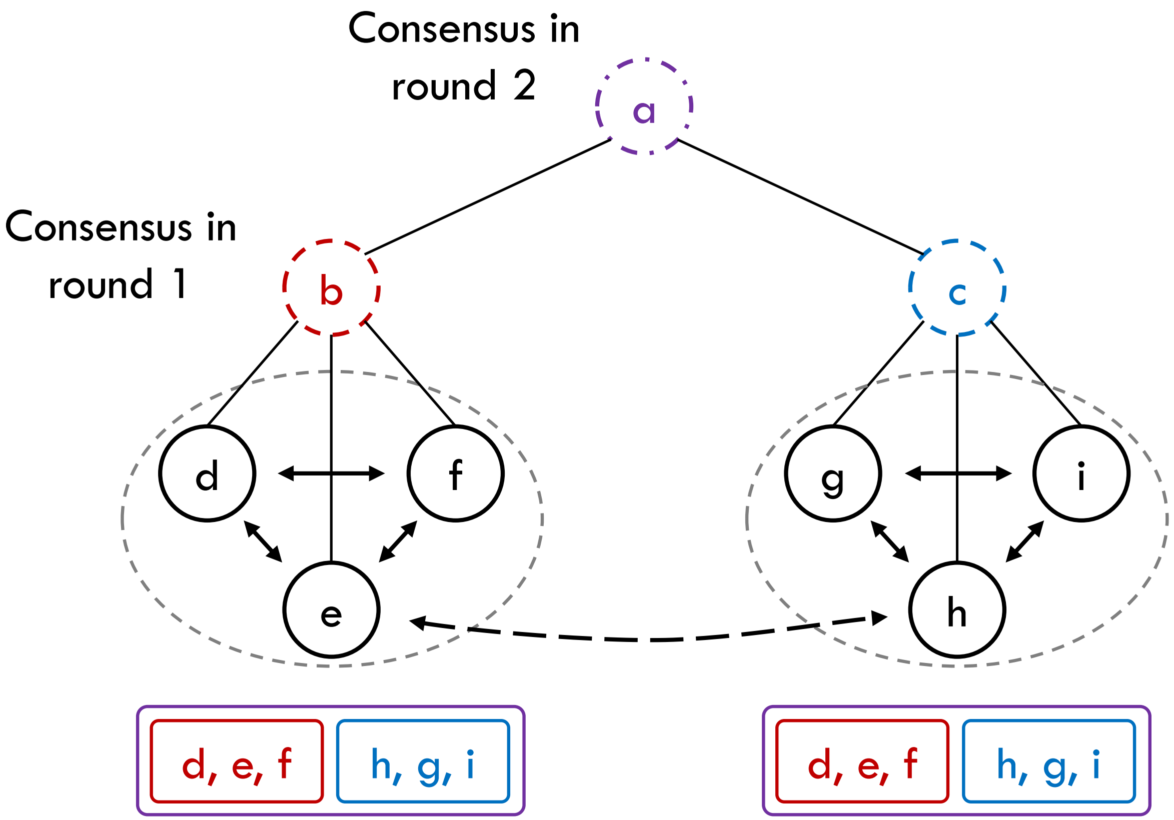
ACHIEVING CONSENSUS

Members of a height 1 vnode exchange state with members of nearby height 1 vnodes to compute a height 2 vnode

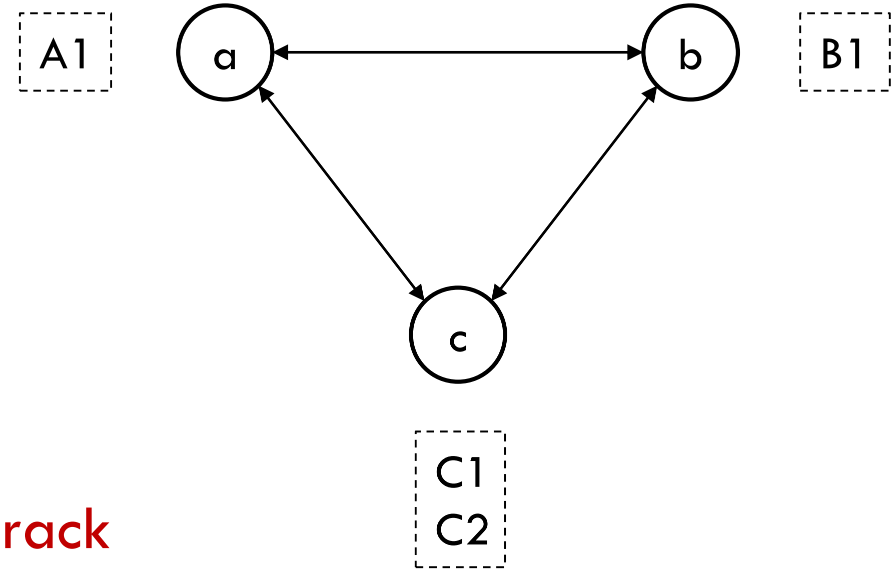
- State exchange is greatly simplified since each vnode is fault tolerant

h rounds in a consensus cycle

A node completes a consensus cycle once it has computed the state of the root vnode



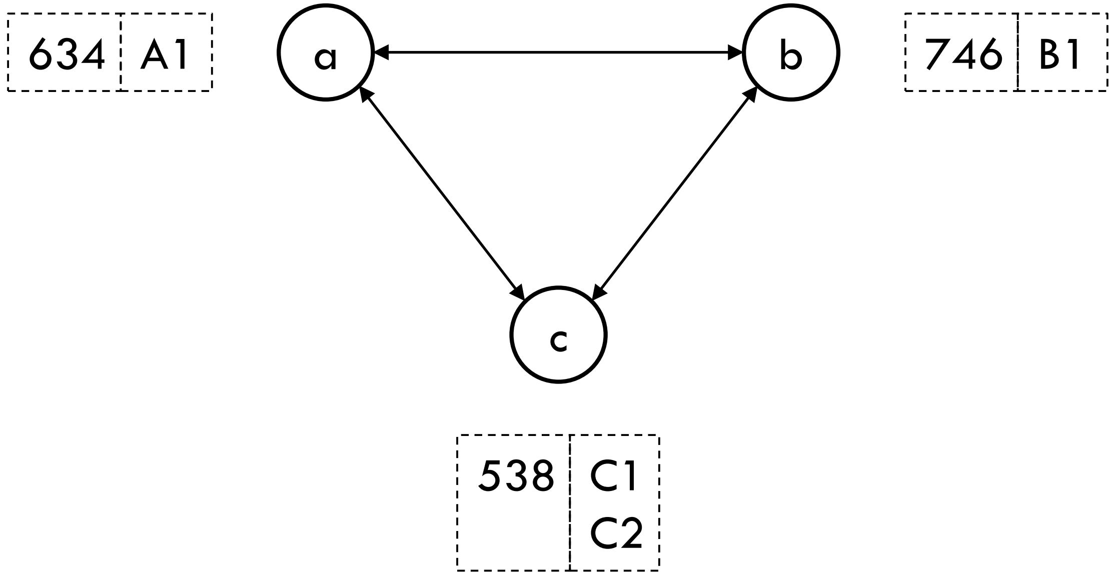
CONSENSUS PROTOCOL WITHIN A SUPER-LEAF



- Exploit low latency within a rack
  - Reliable broadcast
  - RAFT

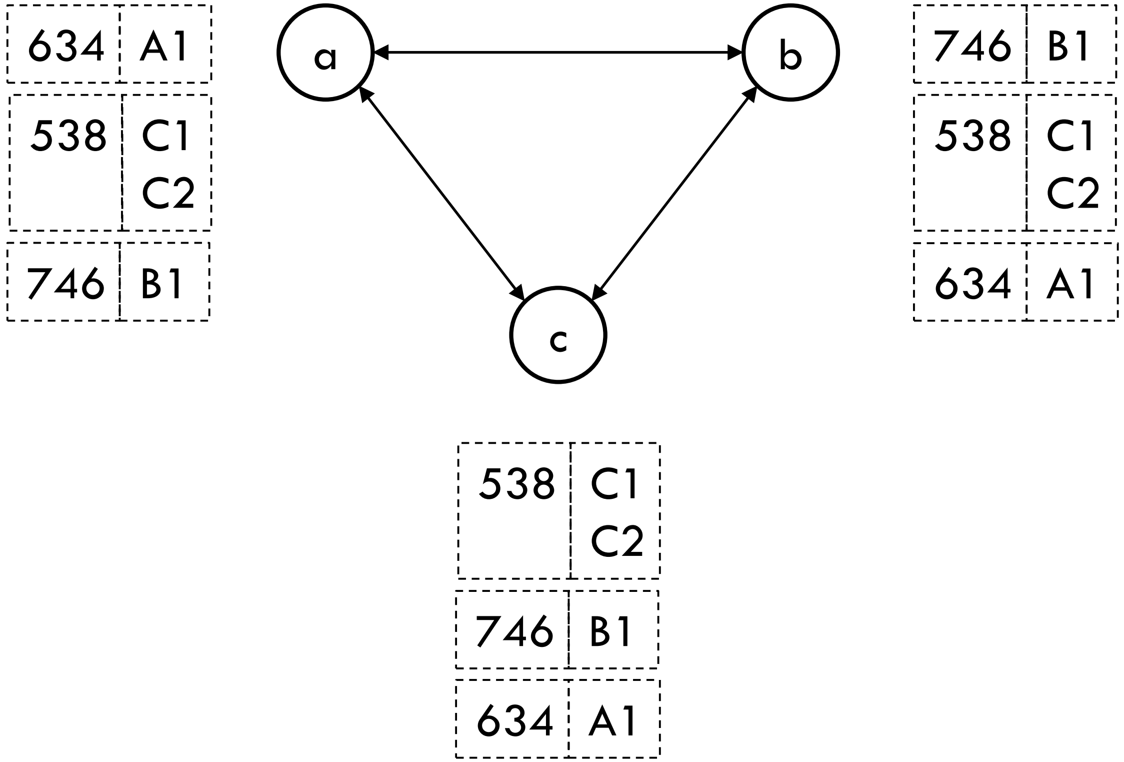
CONSENSUS PROTOCOL WITHIN A SUPER-LEAF

1. Nodes prepare a proposal message that contains a random number and a list of pending write requests



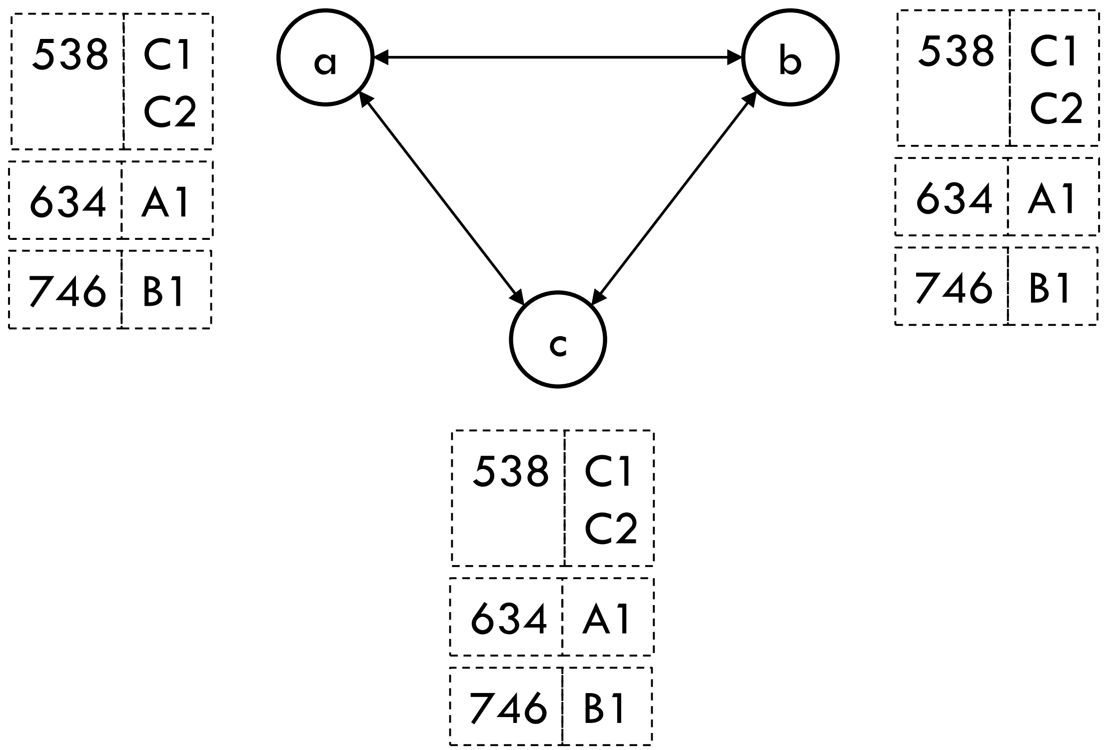
CONSENSUS PROTOCOL WITHIN A SUPER-LEAF

2. Nodes use reliable broadcast to exchange proposals within a super-leaf

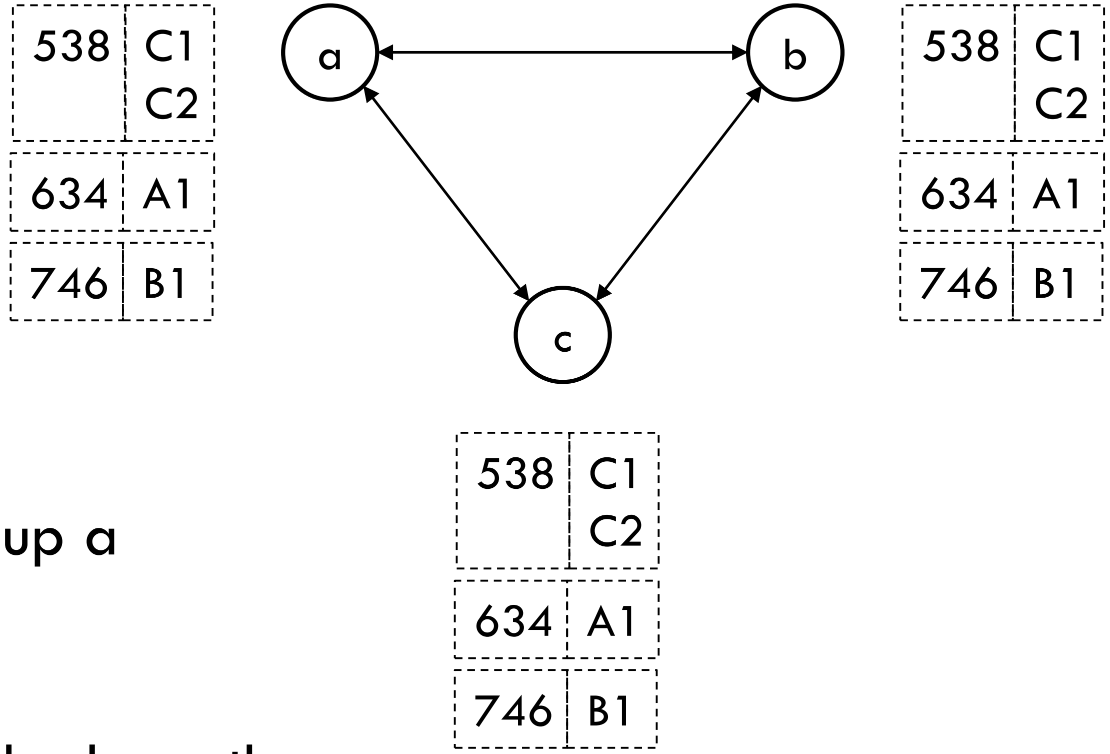


CONSENSUS PROTOCOL WITHIN A SUPER-LEAF

3. Every node orders proposals



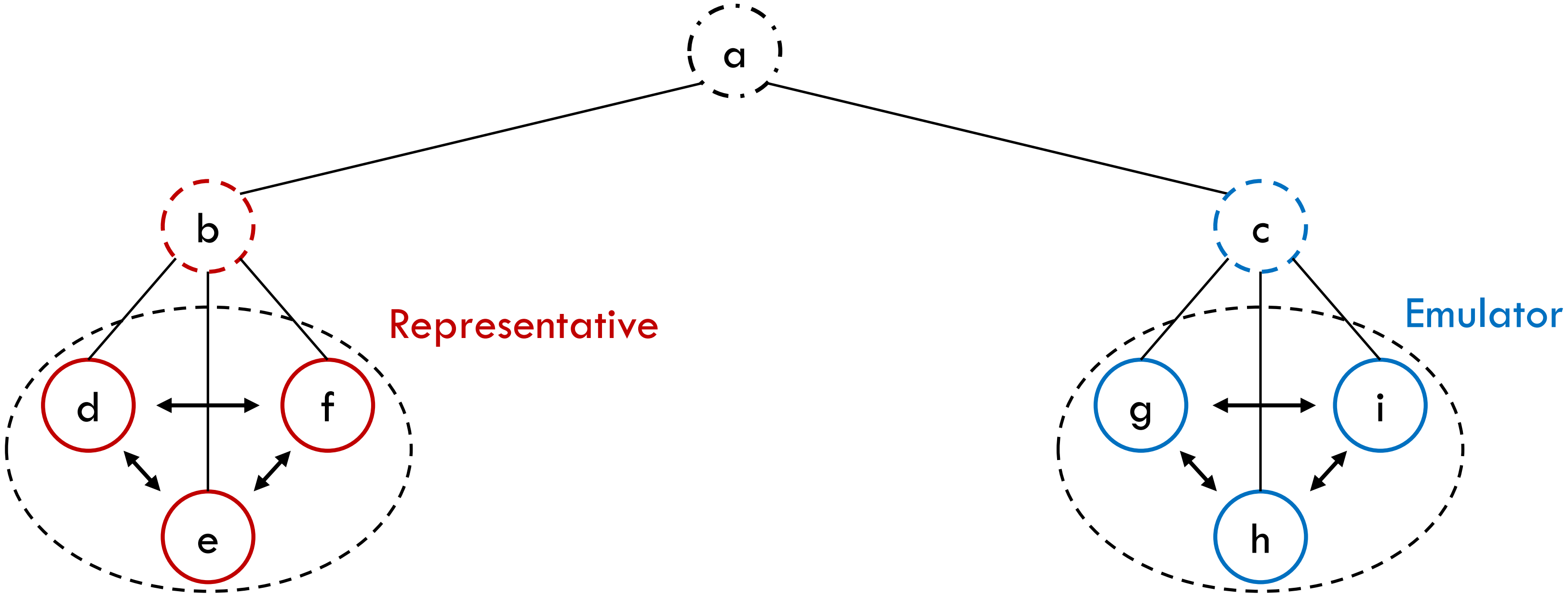
CONSENSUS PROTOCOL WITHIN A SUPER-LEAF



These three steps make up a consensus round.

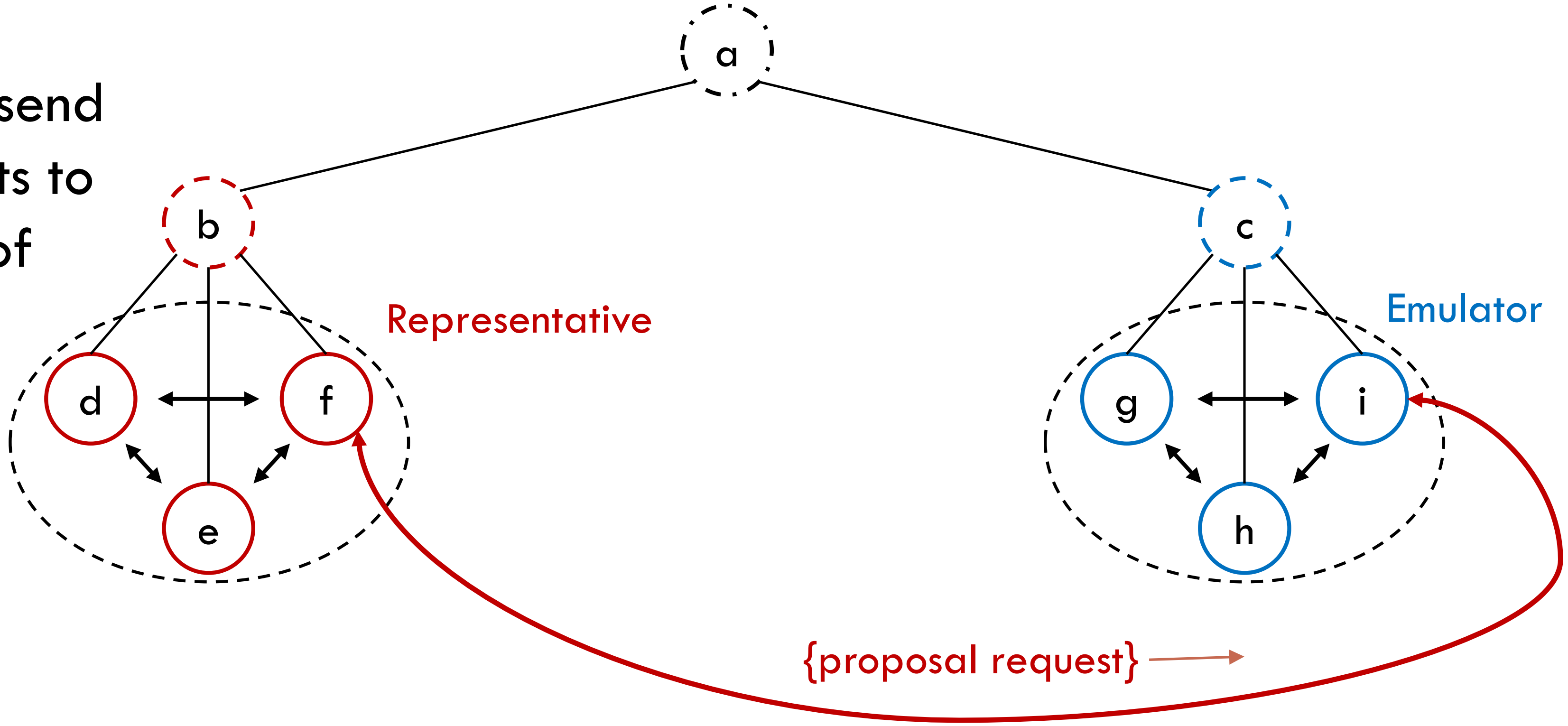
At the end, all three nodes have the same state of their common parent.

CONSENSUS PROTOCOL BETWEEN SUPER-LEAVES



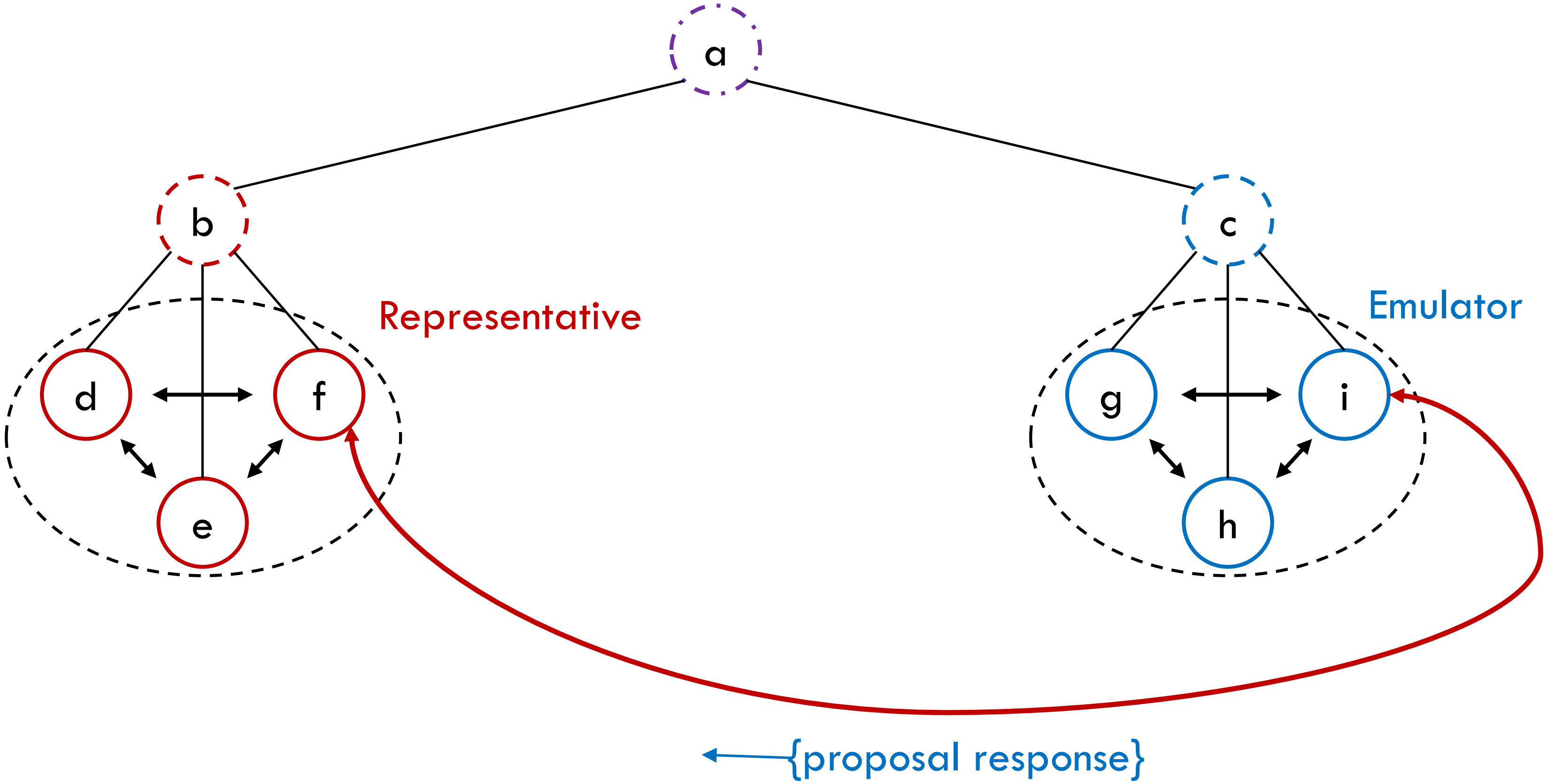
CONSENSUS PROTOCOL BETWEEN SUPER-LEAVES

1. Representatives send proposal requests to fetch the states of vnodes



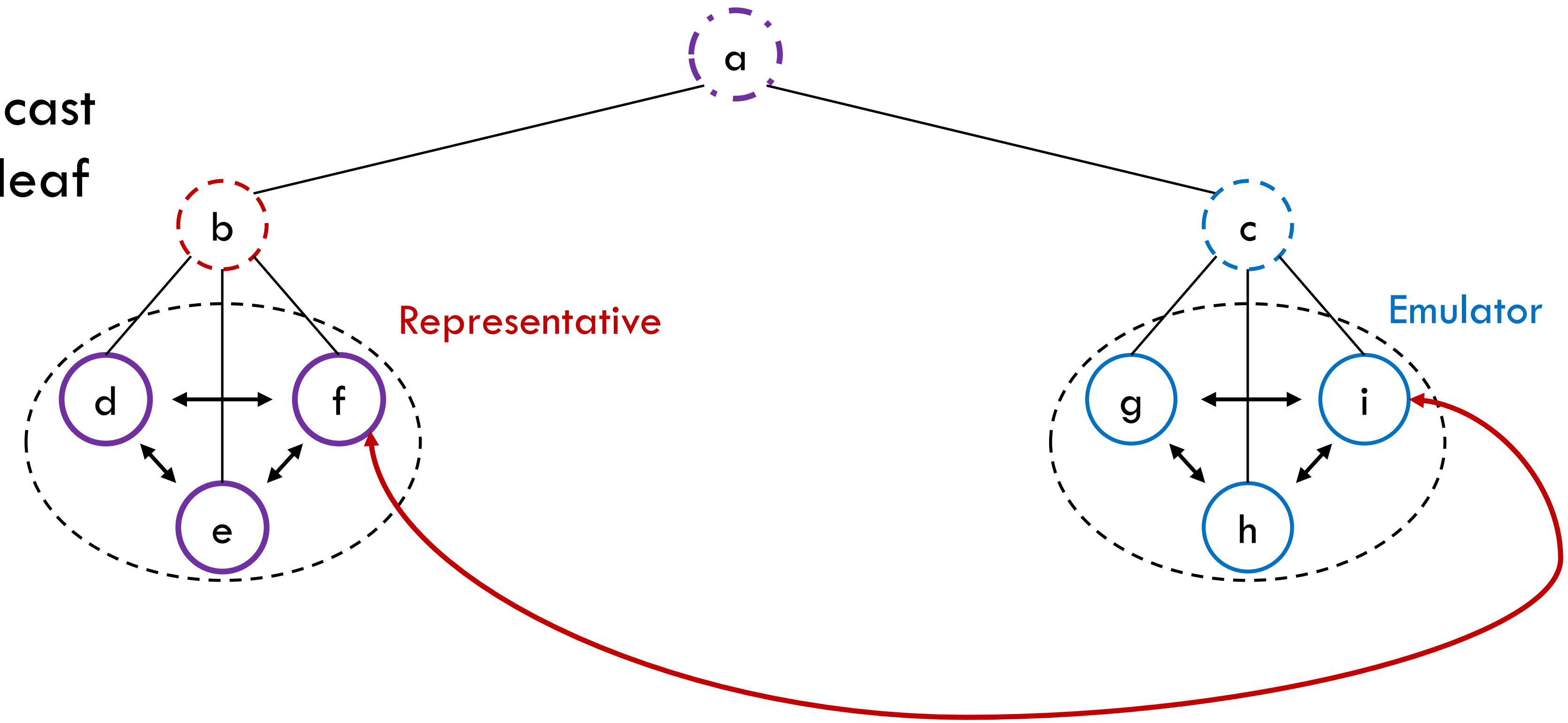
CONSENSUS PROTOCOL BETWEEN SUPER-LEAVES

2. Emulators reply with proposals



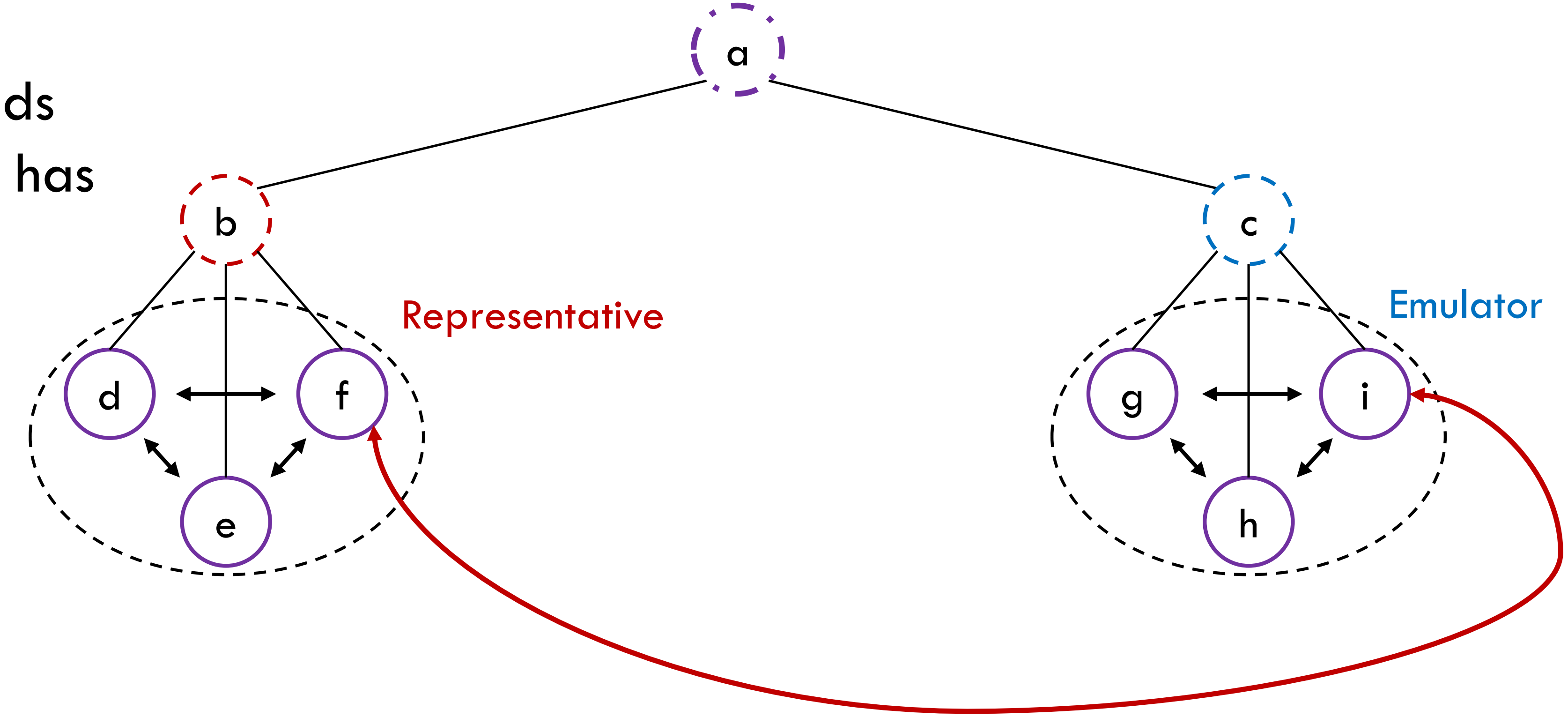
CONSENSUS PROTOCOL BETWEEN SUPER-LEAVES

3. Reliable broadcast within a super-leaf



CONSENSUS PROTOCOL BETWEEN SUPER-LEAVES

Consensus cycle ends for a node when it has completed the last round



READ REQUESTS

Read requests can be serviced locally by any Canopus node

- Does not need to disseminate to other participating nodes

Provides linearizability by

- Buffering read requests until the global ordering of writes has been determined
- Locally ordering its pending reads and writes to preserve the request order of its clients

Significantly reduces bandwidth requirements for read requests

Achieves total ordering of both read and write requests

ADDITIONAL OPTIMIZATIONS

Pipelining consensus cycles

- Critical to achieving high throughput over high latency links

Write leases

- For read-mostly workloads with low latency requirements
- Reads can complete without waiting until the end of a consensus cycle

EVALUATION: MULTI DATACENTER CASE

3, 5, and 7 datacenters

- Each datacenter corresponds to a super-leaf

3 nodes per datacenter (up to 21 nodes in total)

- EC2 c3.4xlarge instances

100 clients in five machines per datacenter

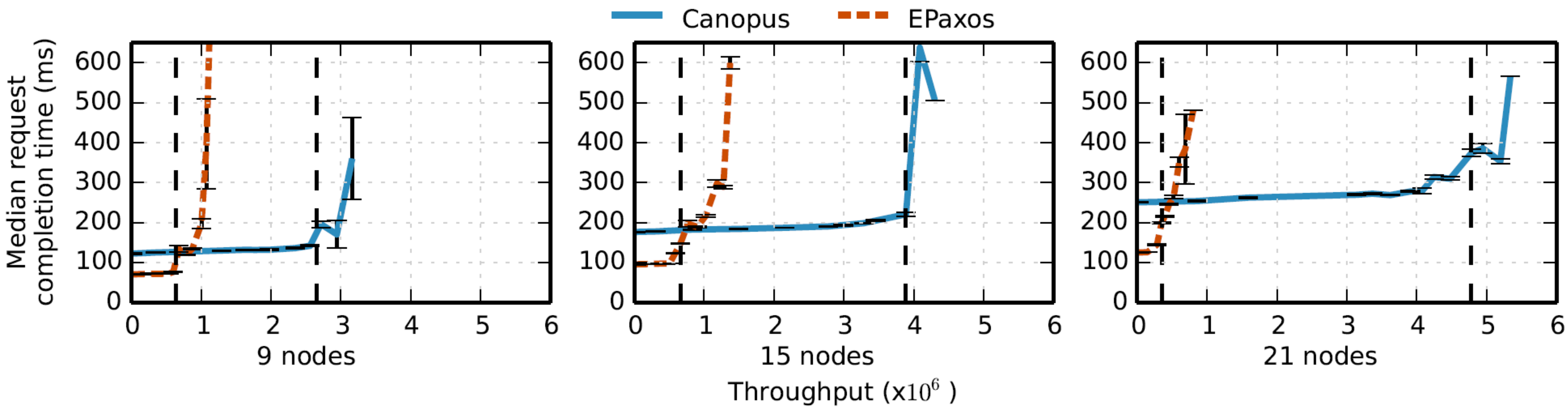
- Each client is connected to a random node in the same datacenter

Latencies across datacenters (in ms)

	IR	CA	VA	TK	OR	SY	FF
IR	0.2						
CA	133	0.2					
VA	66	60	0.25				
TK	243	113	145	0.13			
OR	154	20	80	100	0.26		
SY	295	168	226	103	161	0.2	
FF	22	145	89	226	156	322	0.23

Regions: Ireland (IR), California (CA), Virginia (VA), Tokyo (TK), Oregon (OR), Sydney (SY), Frankfurt (FF)

CANOPUS VS. EPAXOS (20% WRITES)



EVALUATION: SINGLE DATACENTER CASE

3 super-leaves of sizes of 3, 5, 7, 9 servers (i.e., up to 27 total servers)

- Each server has 32GB RAM, 200 GB SSD, 12 cores running at 2.1 GHz

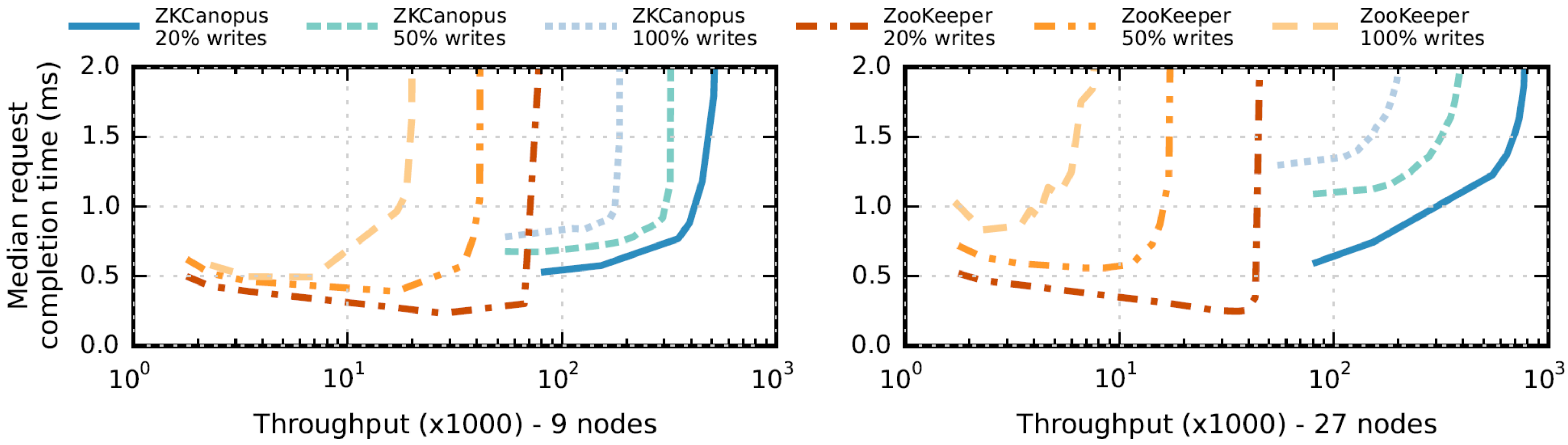
Each server has a 10G to its ToR switch

- Aggregation switch has dual 10G links to each ToR switch

180 clients, uniformly distributed on 15 machines

- 5 machines in each rack

ZKCANOPUS VS. ZOOKEEPER



LIMITATIONS

We trade off fault tolerance for performance and understandability

- Cannot tolerate full rack failure or network partitions

We trade off latency for throughput

- At low throughputs, latencies can be higher than other consensus protocols

Stragglers can hold up the system (temporarily)

- Super-leaf peers detect and remove them

ON-GOING WORK

Handling super-leaf failures

- For applications with high availability requirements
- Detect and remove failed super-leaves to continue

Byzantine fault tolerance

- Canopus currently supports crash-stop failures
- Aiming to maintain our current throughput

CONCLUSIONS

Emerging applications involve consensus at large scales

- Key barrier is a scalable consensus protocol

Addressed by Canopus

- Decentralized
- Network topology aware
- Optimized for modern datacenters