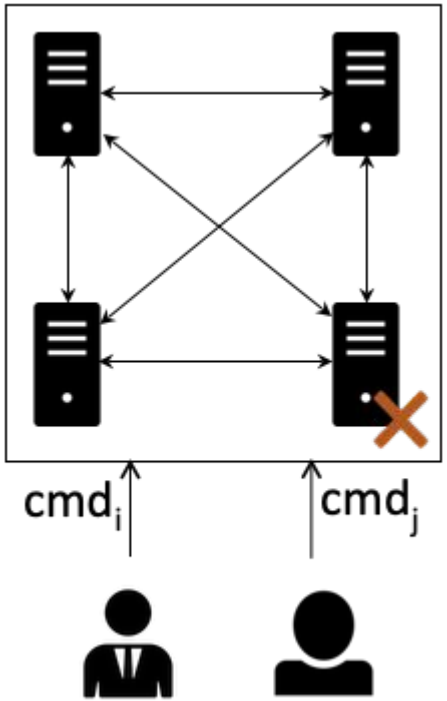
Sailfish: Towards Improving the Latency of DAG-based BFT SMR

Nibesh Shrestha, Rohan Shrothrium, Aniket Kate, and Kartik Nayak

State Machine Replication



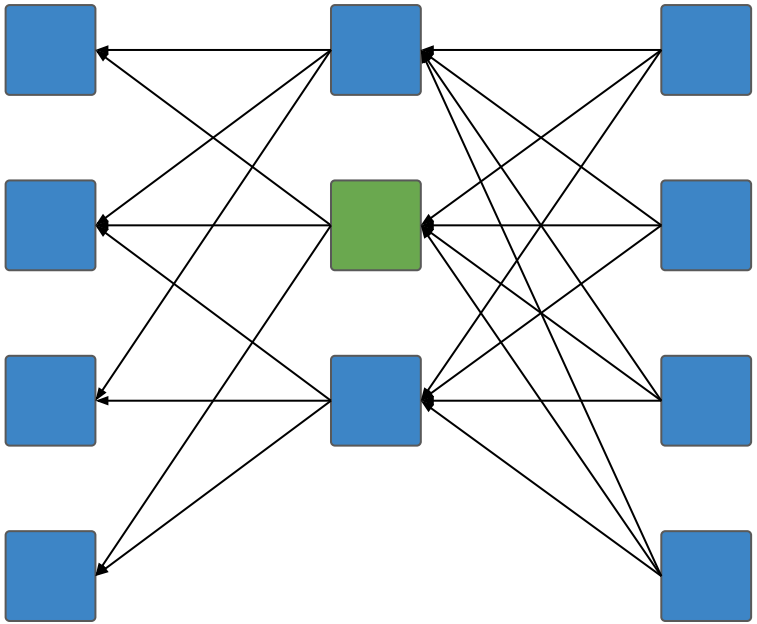
Safety: All non-faulty servers must commit on same sequence of commands

Liveness: Commands issued by clients must eventually be committed by non-faulty servers

Model:

- Partially synchronous model
- $f < n/3$ Byzantine faults

Overview of DAG-based BFT SMR



Round r-1 r r+1

All nodes propose a vertex in each round using an RBC

A round *r* vertex references *2f+1* round *r-1* vertices.

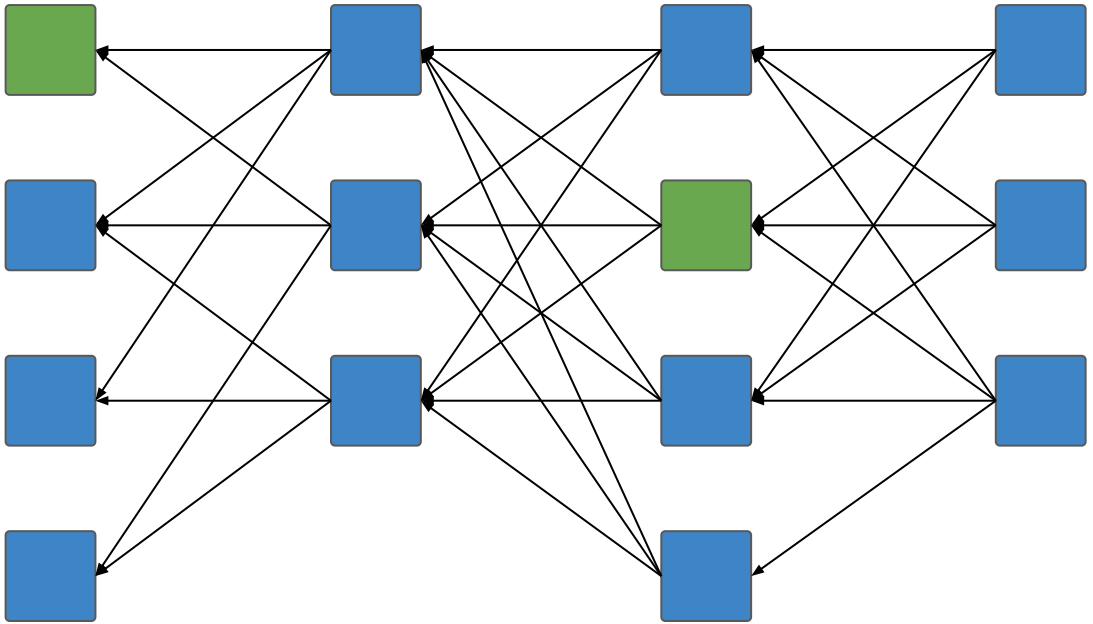
A leader in a round; Only leader vertices are committed. Non-leader vertices are ordered as when leader vertex is committed.

f: No. of Byzantine faults, $f < n/3$

Leader vertex

Non-leader vertex

Latency in Existing DAG-based BFT



Round r-1 r r+1

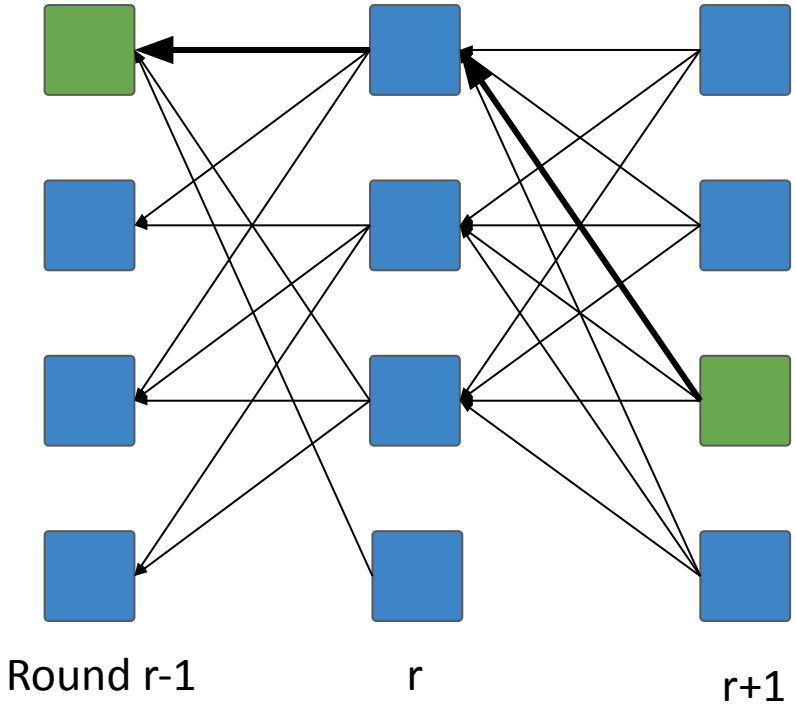
Leader every 2 rounds: Increases latency of non-leader vertices

Latency of Leader vertex: 2 RBCs

Latency of non-leader vertices: 4 RBCs

Leader vertex Non-leader vertex

Key Technical Challenge

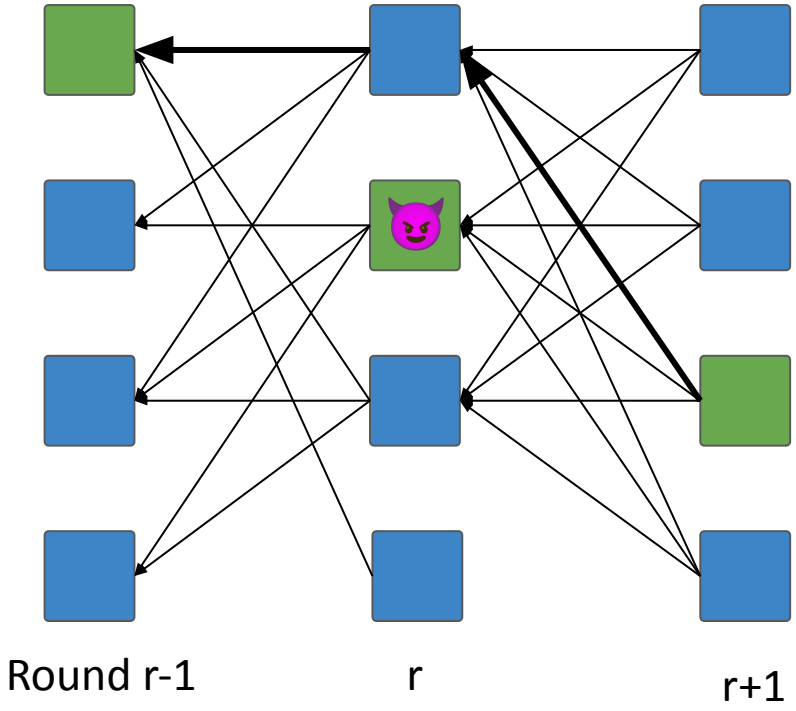


Safety property:
If the round *r-1* leader vertex is committed, the subsequent leader vertices must have a path to the round *r-1* leader vertex

Easy to satisfy if leaders are every 2 rounds

Leader vertex Non-leader vertex

Key Technical Challenge

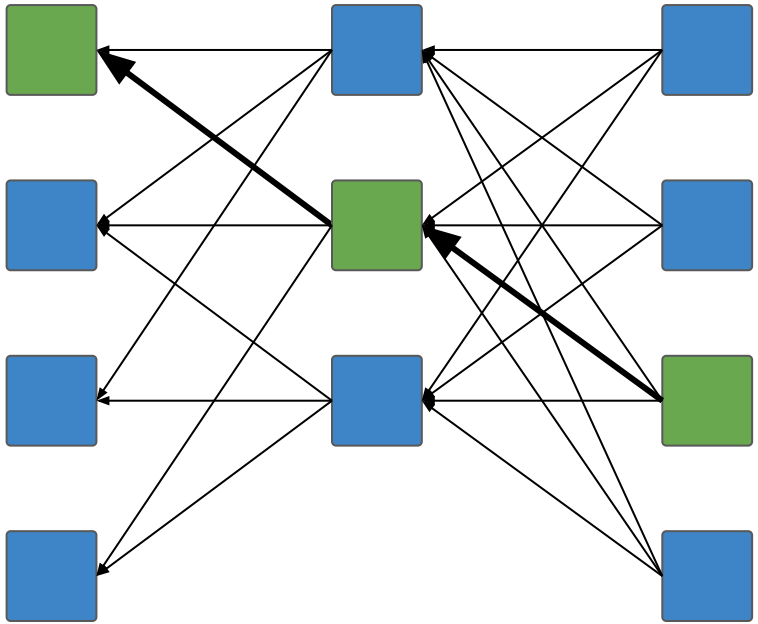


If leaders in every round:
A round r leader vertex may not point to round r-1 leader vertex; violates the safety property

Leader vertex Non-leader vertex

Is it possible to support a leader vertex in each round and reduce the latency of DAG-based protocols?

Sailfish



Round r-1 r r+1

Supports leaders in each round

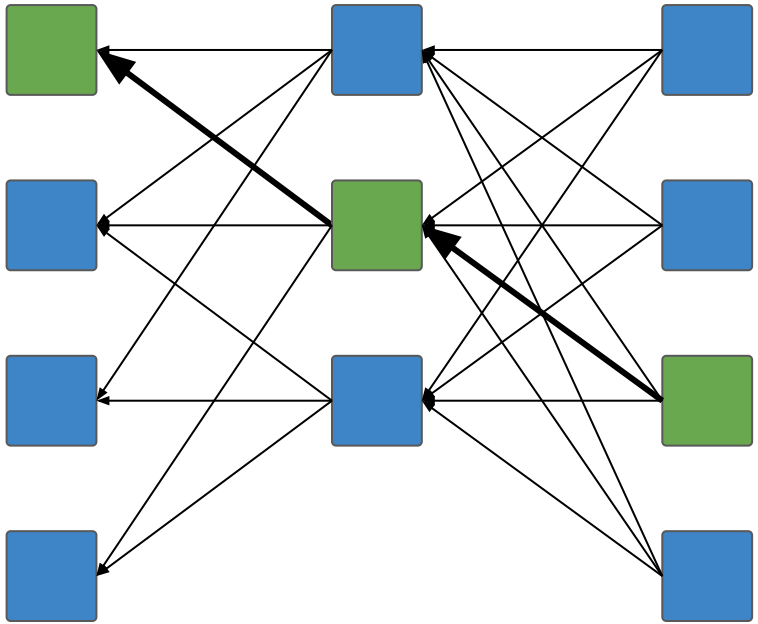
Commits leader vertex with **1RBC+1δ**
Non-leader vertices with **2RBC+1δ**

δ : Actual transmission delay of message

Leader vertex

Non-leader vertex

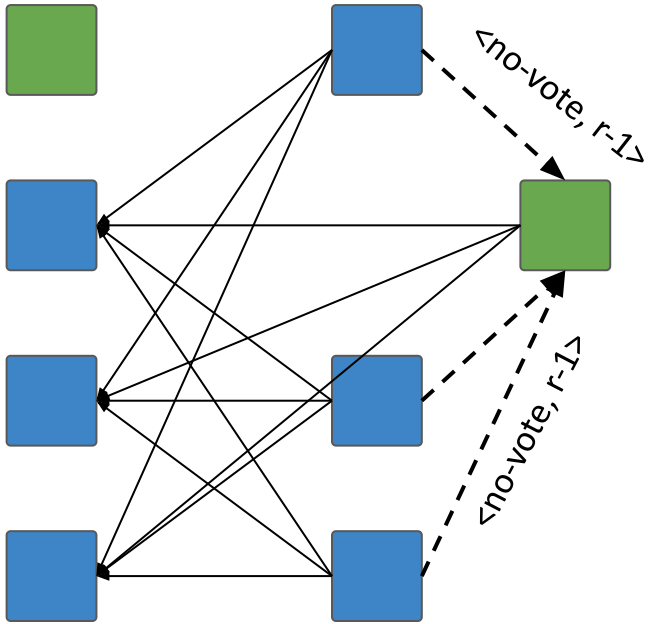
Supporting a Leader Vertex in Each Round



The round *r* leader vertex must reference the round *r-1* leader vertex **OR** …

Leader vertex Non-leader vertex

Supporting a Leader Vertex in Each Round



The round *r* leader vertex must show a proof that the round *r-1* leader vertex could not have been committed.

In our protocol, nodes send <*no-vote, r-1*> message if they did not vote for the round *r-1* leader vertex.

2f+1 <*no-vote, r-1*> messages constitute **no-vote certificate**; serves as a proof that round r-1 leader vertex could not have been committed.

Leader vertex Non-leader vertex

Improving Commit Latency to 1RBC + 1δ

Property of RBC:
When the sender is honest, the value is the first message of RBC is the value that is (eventually) delivered

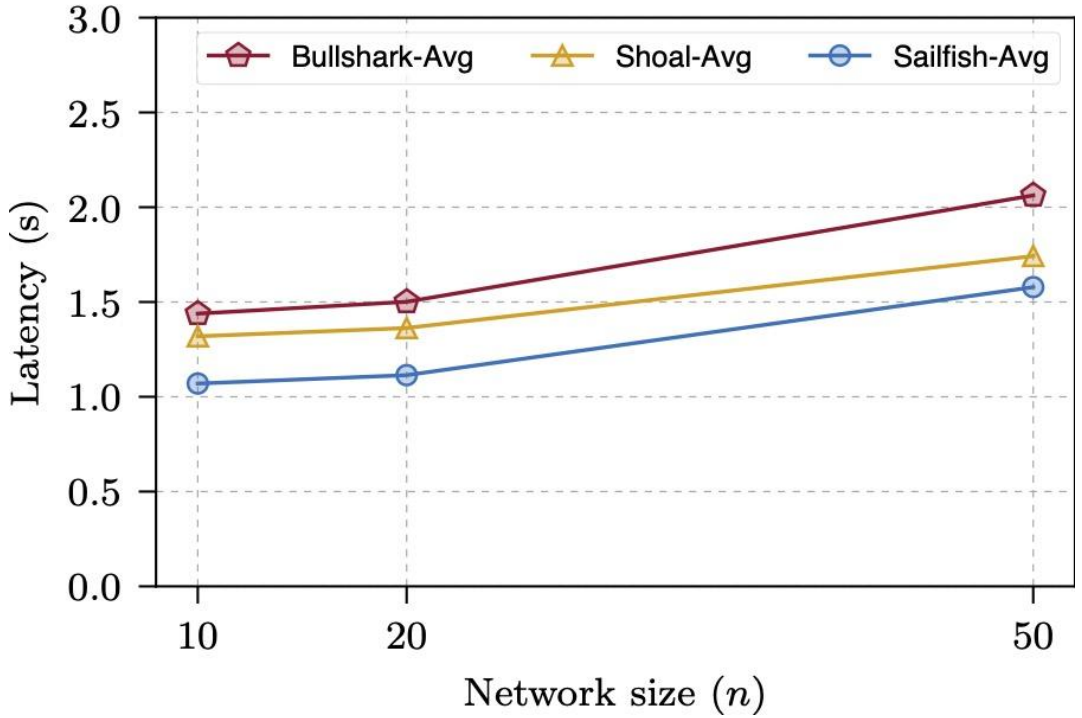
Commit rule for round *r-1* leader vertex:
If *2f+1* first message of round *r* vertices refer to the round *r-1* leader vertex.

Safety argument:
At least *f+1* first messages are sent by honest parties.

- At least *f+1* round *r* vertices with path to round *r-1* leader vertex
- no-vote certificate cannot exist

δ : Actual transmission delay of message

Experimental Evaluation

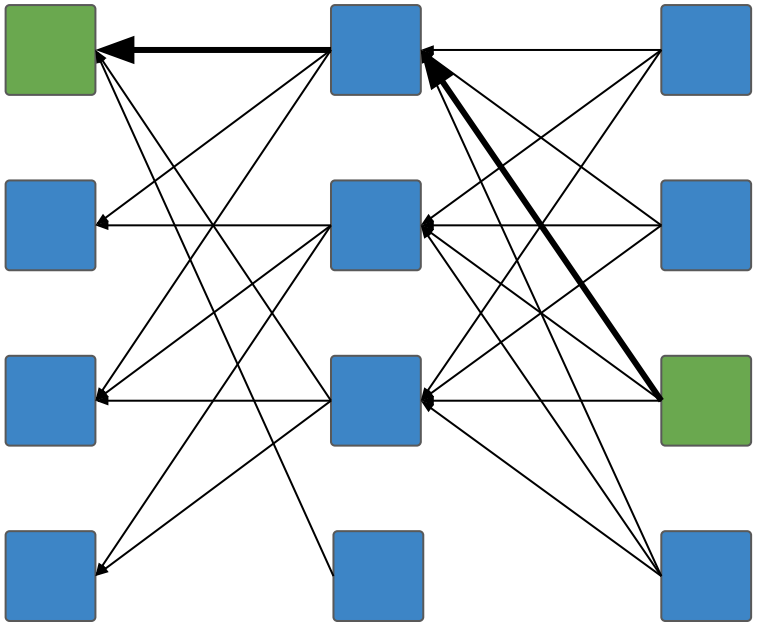


Thank you!

Paper Link

Contact: n.shrestha@supraoracles.com

Key Technical Challenge



Easy if leaders are every 2 rounds:
A round r-1 leader vertex is committed when it is referenced by at least f+1 round r vertices.

The round r+1 vertices reference 2f+1 round r vertices. So, every round r+1 vertex will have a path to round r-1 leader vertex.

Similarly, every round r' > r+1 vertex will have a path to round r-1 leader vertex.

Leader vertex Non-leader vertex