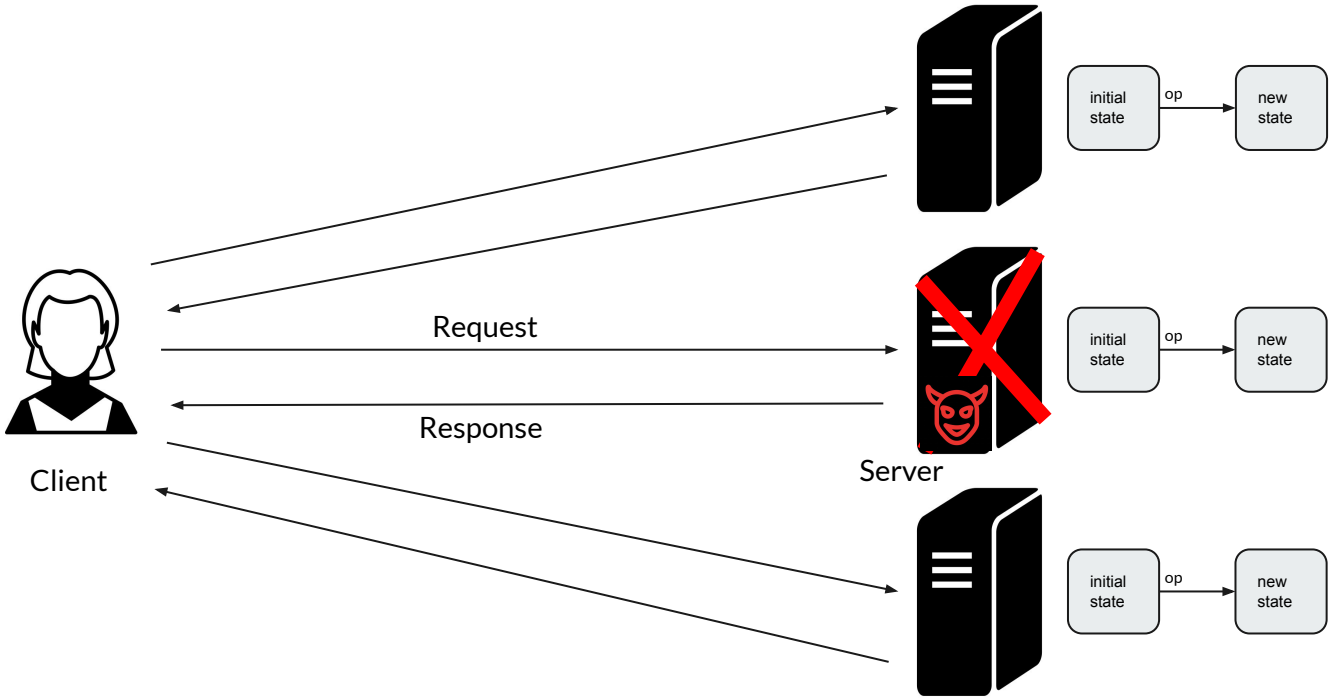
# Byzantine Ordered Consensus without Byzantine Oligarchy

Yunhao Zhang, Srinath Setty, Qi Chen, Lidong Zhou, and Lorenzo Alvisi

**Presented by:**
Harsha Tolani
Kaustubh Shete
Seema Upadhya
Shivam Rai Sharma
Shivani Kalamadi

# State Machine Replication



# BFT SMR

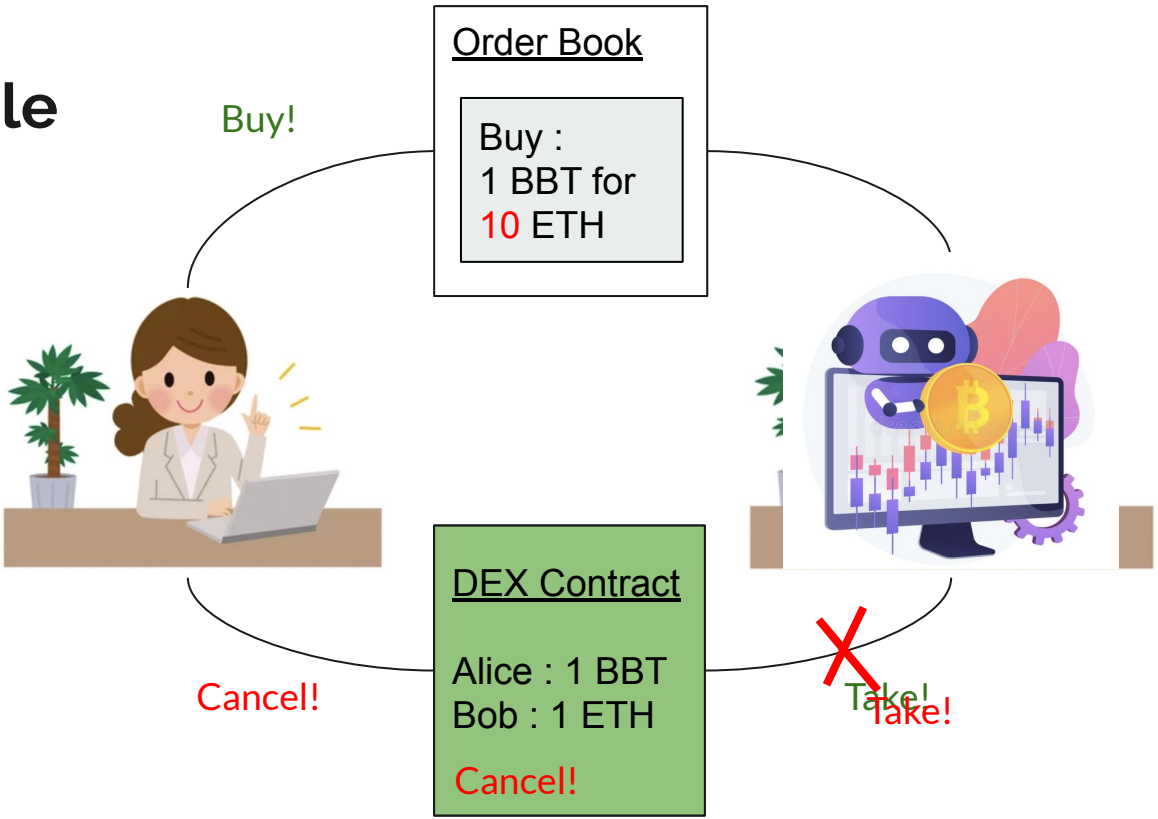
- Most of the practical solutions for **BFT SMR** are based on the **Primary-Backup paradigm.**
- In this approach, in **each view**, there is a **leader in charge** to drive decisions efficiently, until replaced by the next leader.
- The **Primary-Backup** approach for SMR exposes deep connections to ***broadcast***.
- Each **view** in BFT SMR is similar to an **instance of broadcast** where the **leader** taking on a similar role as the **broadcaster**.

## Why do we need ordering of requests?

In the permissioned blockchains that rely on Byzantine fault tolerant (BFT) SMR, however, nodes have a **stake** in the **specific sequence** that **ledger** records, as well as in **preventing** other **parties** from **manipulating** the **sequencing** to their advantage .

## More importantly, why do we need fair ordering of requests?

# Example


Order Book
Buy :
1 BBT for
10 ETH
Buy!
DEX Contract
Alice : 1 BBT
Bob : 1 ETH
Cancel!
Take!
Take!
Cancel!

# Example

Cancel !!

Gas price : 20

Take !

Gas price : 30

Ethereum network

Block

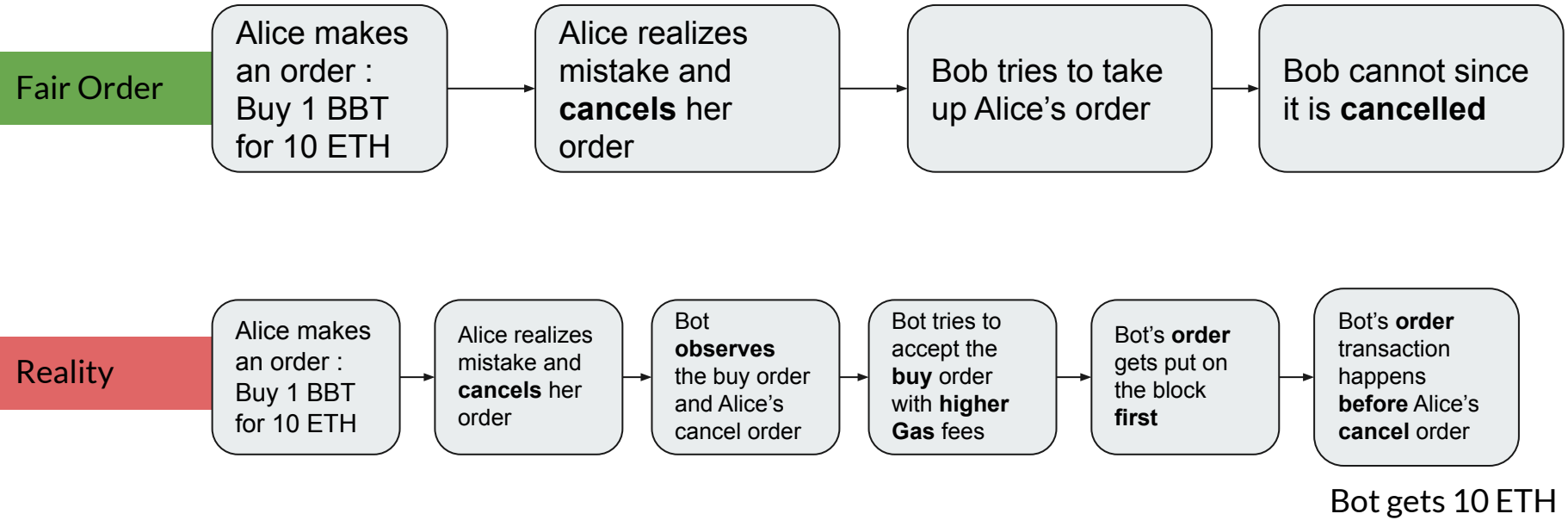
Take !

Cancel !!

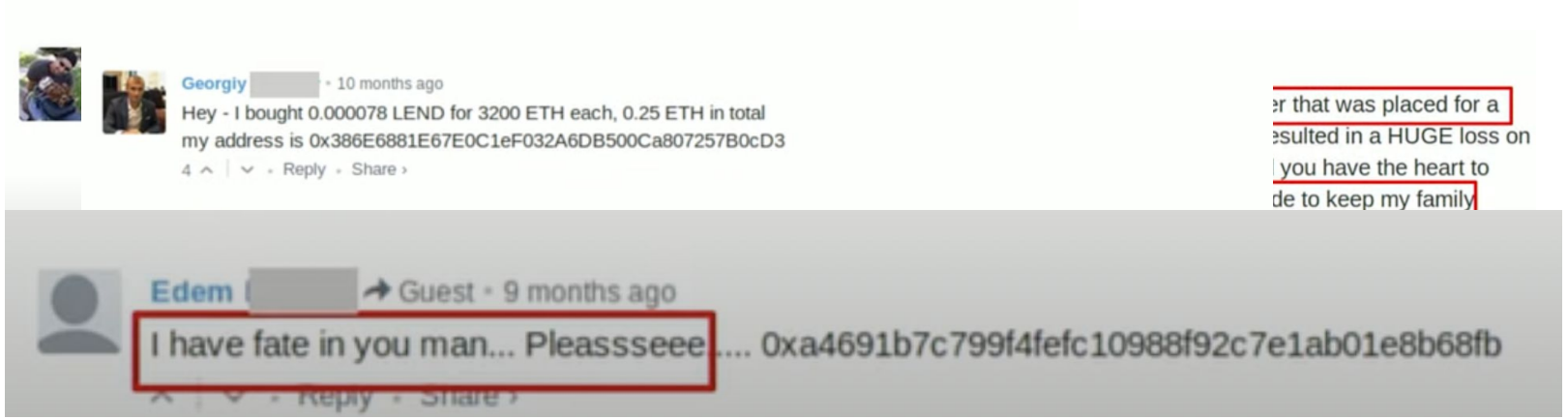
DEX Contract

Alice : 1 BBT
Bot : 10 ETH

# Recap of sequence of commands

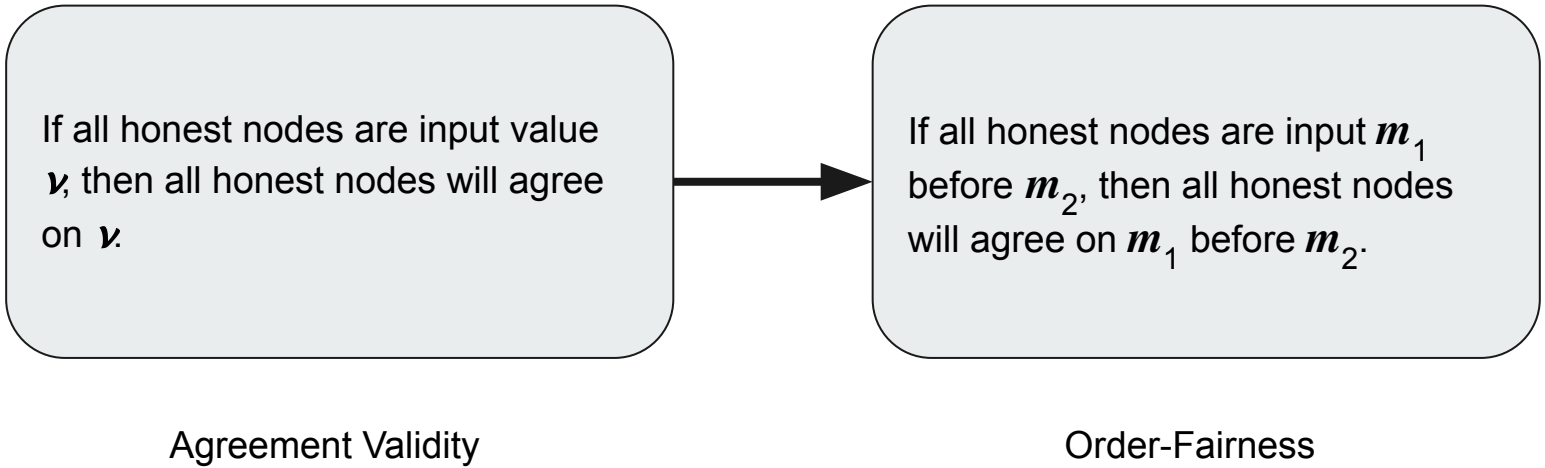


# Real life examples




Cite : Juels Ari , “Flash Boys 2.0”, *YouTube* uploaded by MIT Digital Currency Initiative,15 Oct,2019, https://www.youtube.com/watch?v=7yJa_6CtvHk

# Fair Ordering



# Motivation

- There is **no specification** to address **ordering concerns** in traditional systems because it attaches no importance to the sequence it produces.

- It is **incapable** of characterizing what makes the total order **“right” or “wrong”.**

# Goal Of The Paper

- **Byzantine ordered consensus** that increases the **correctness** of **BFT SMR**

- A **new architecture for BFT SMR** which **factors** out **ordering from consensus** which prevents Byzantine nodes from controlling ordering decisions(**Byzantine oligarchy**).

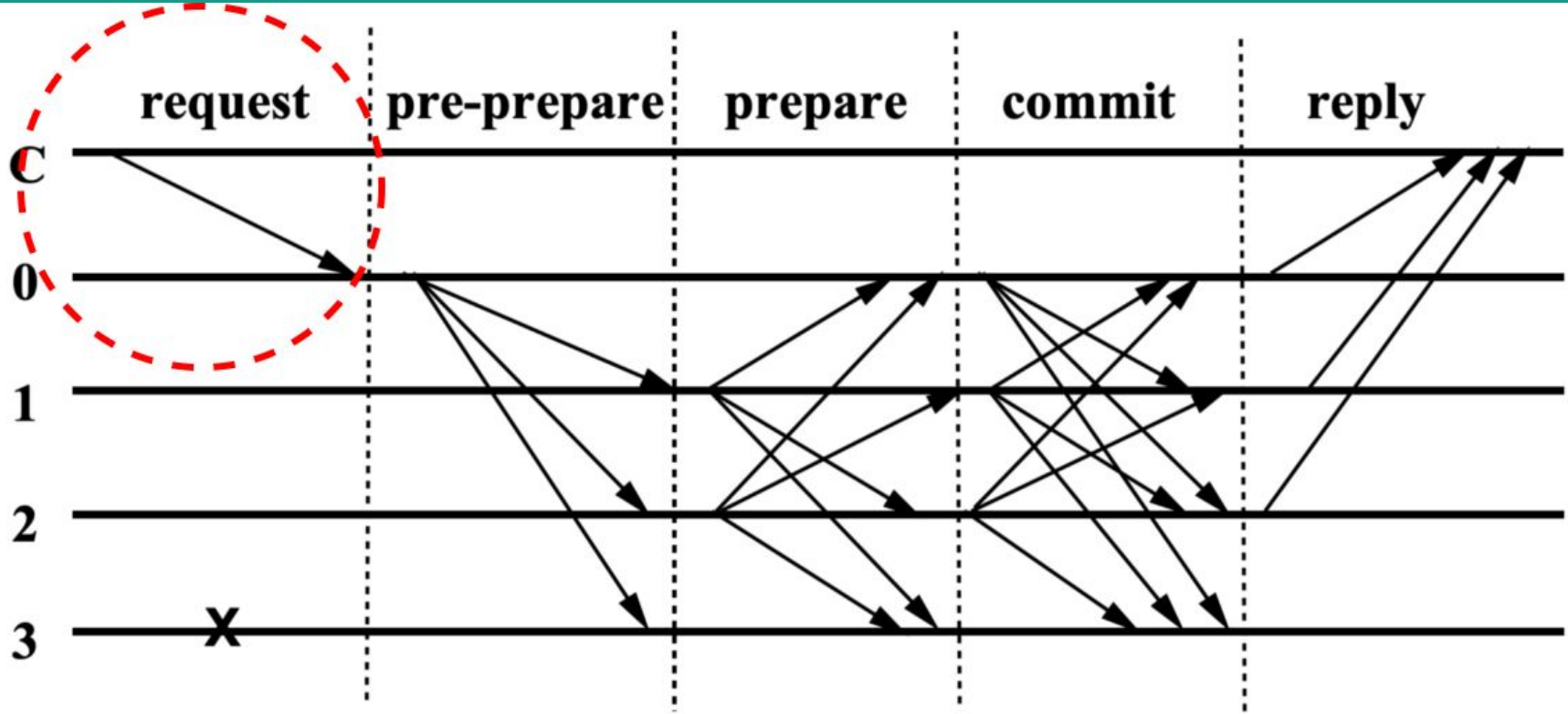
# Byzantine Ordered Consensus

Participating nodes can propose commands and also indicate how they prefer the commands to be ordered.

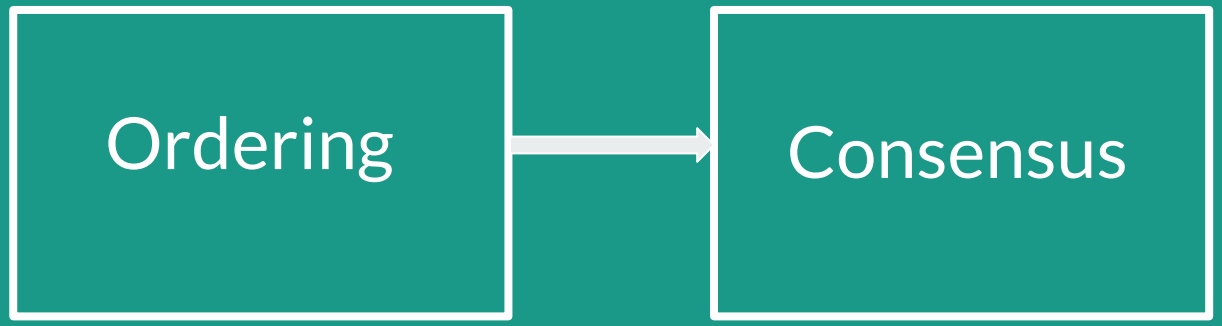
This makes it possible to specify which total orders a correct BFT SMR is allowed to produce, given the nodes ordering preferences.

# Pompe Protocol

# Our Focus



# How do we do that?

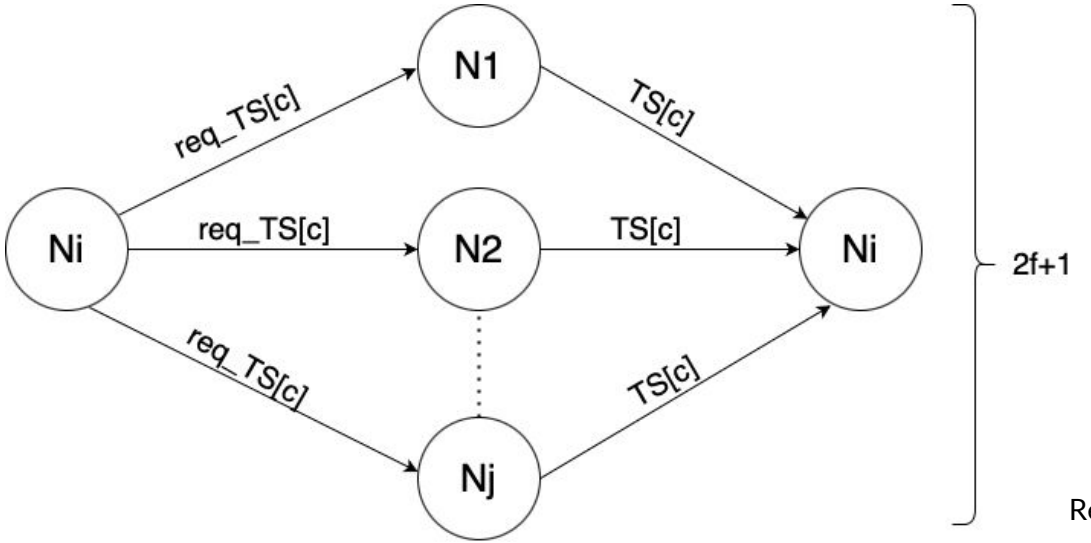



Agreement on ordering

# For example:

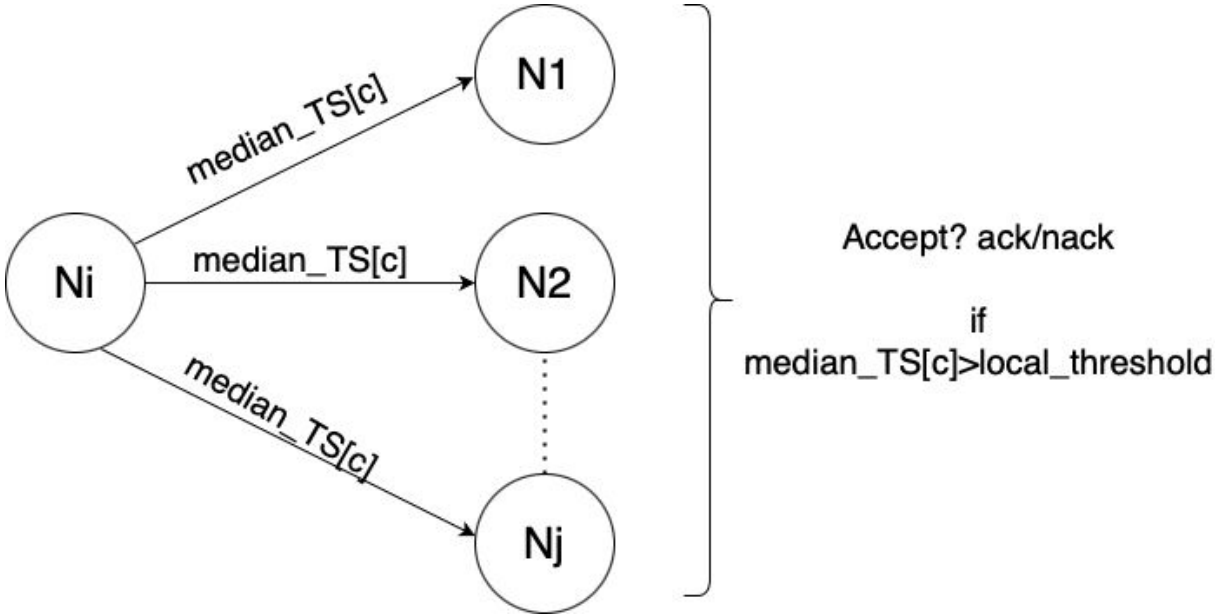
| TimeStamp ⟹<br>Nodes ⇩ | 1 | 2 | 3 |
|---|---|---|---|
| N1 | C1 | C2 | C3 |
| N2 | C1 | C3 | C2 |
| N3 | C2 | C4 | C5 |

# Pompe Protocol - Ordering Step 1




Request_ TimeStamp[command]

# Pompe Protocol - Ordering Step 2




Median_TimeStamp[command]

# LocalAcceptThresholdTS

- An integer, initialized to 0,
- tracks what $N_j$ believes to be, currently, the latest possible timestamp of any stable command in the ledger

# LocalSequencedSet

- a set, initially empty
- that tracks all commands that the node has accepted

# highTS

- an n-sized vector of integers where highTS[i]
- initialized to 0, stores the

highest timestamp received from node Ni

# highTSMsgs

- an n-sized vector of messages where highTSMsgs[i], initialized to null
- stores the message signed by node Ni that carried the value currently stored in highTS[i]

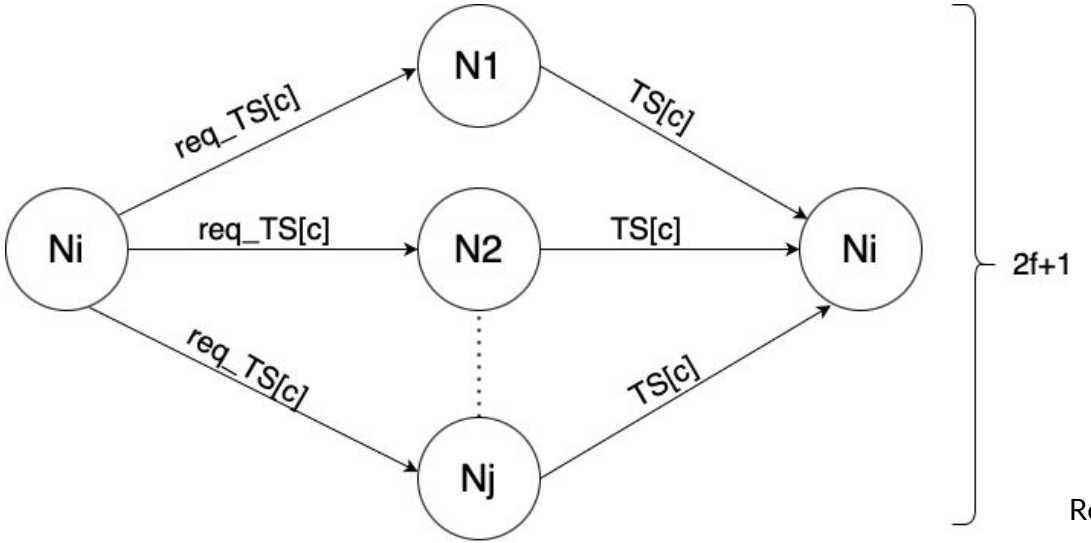
# globalSyncTS

- stores the highest T received so far in a Sync Message

# localSyncTS

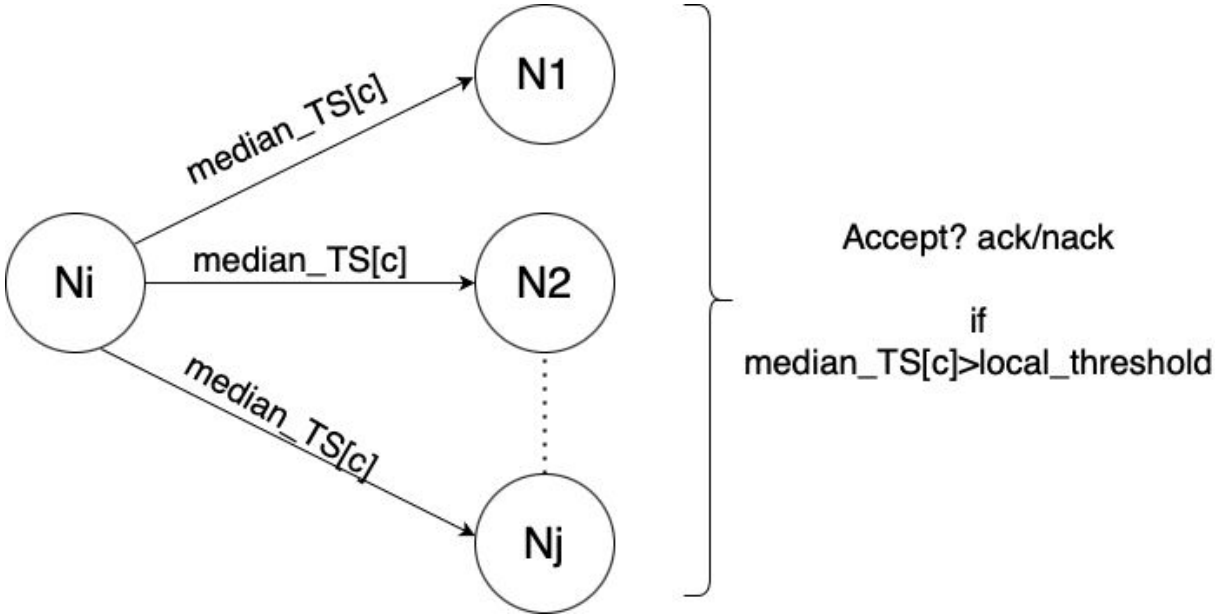
- stores instead the node's local timestamp at the time it received that Sync message

# Pompe Protocol - Ordering Step 1




Request_ TimeStamp[command]

# Pompe Protocol - Ordering Step 2



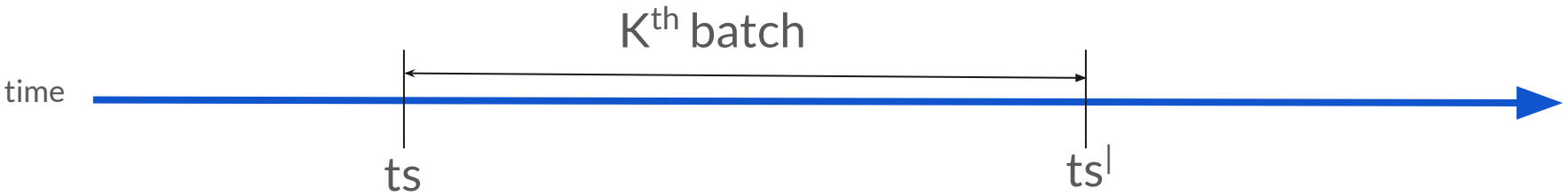

Median_TimeStamp[command]

# Pompe Protocol : Ordering based Consensus

**Only after certain conditions are met, we start Consensus.**

# Guarantee of Termination

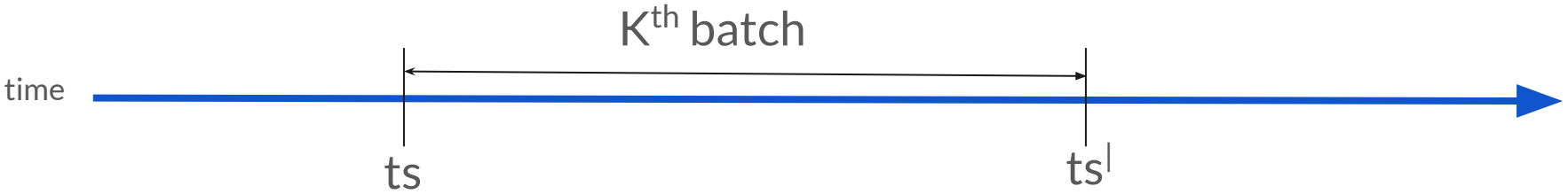
- Consensus will be done in command obtained between ts and ts'
- Consensus will be achieved in batches.



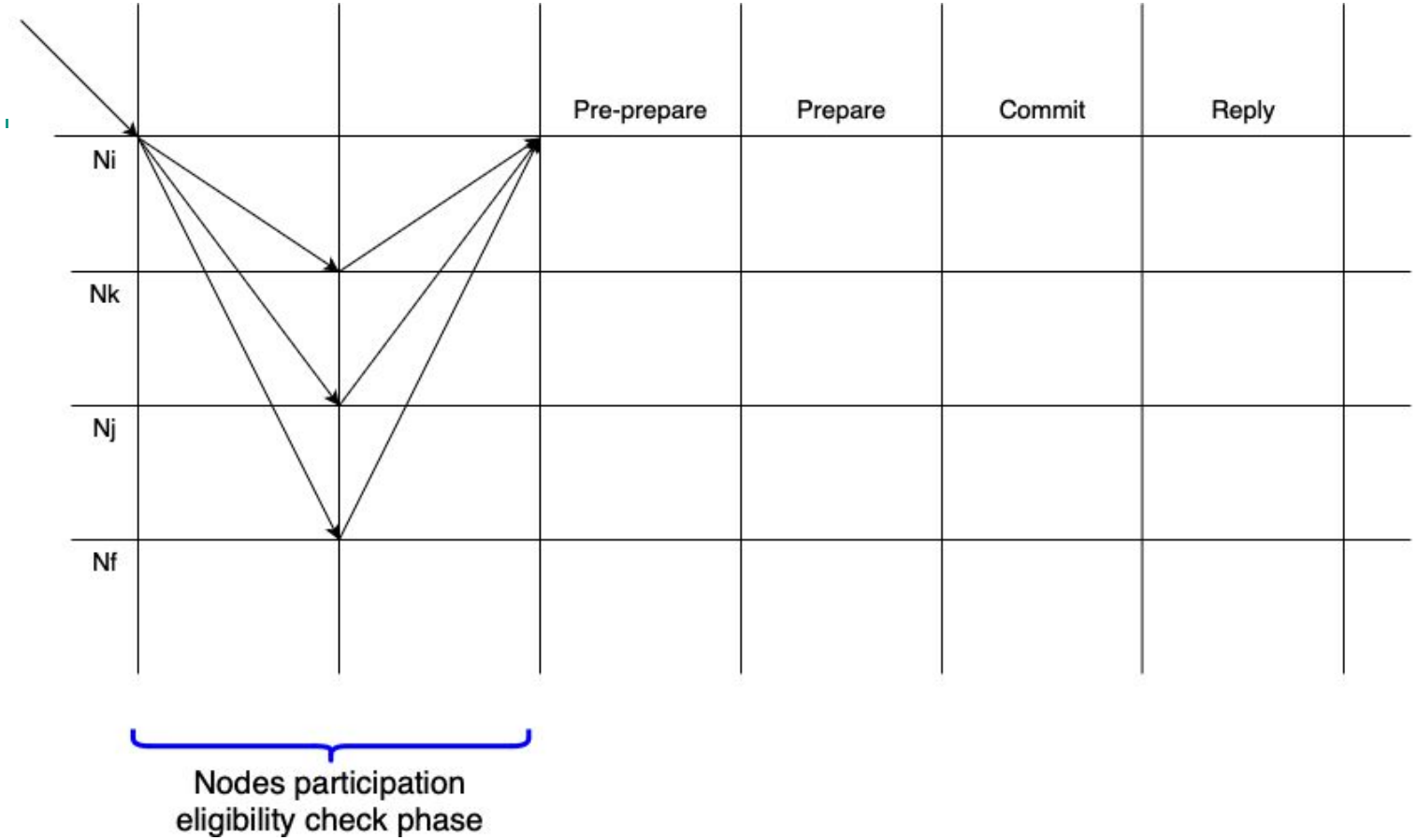
# Node's Participation Eligibility Checking Phase

# 1st Goal of Node's Participation Eligibility Checking Phase:

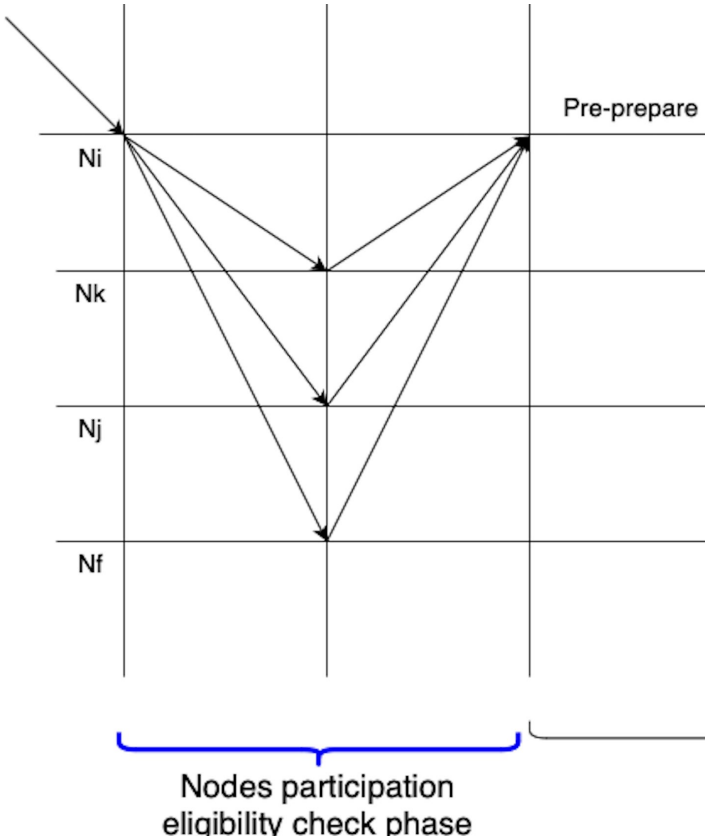
- Guarantee to have the participation of ATLEAST 2f+1 nodes with their sequences of commands which occurred between ts to ts$^{l}$.



# Node's Participation Eligibility Checking Phase



# Node's Participation Eligibility Checking Phase

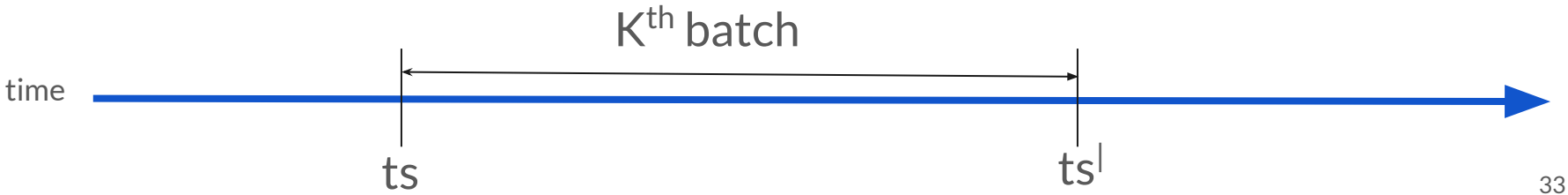


# Outcome of Node's Participation Eligibility Checking Phase:

After $ts^{l}$, if any command arrives and it says that it has a

$$ts[C_{new}] < ts^{l}$$

Then we guarantee to reject Cnew

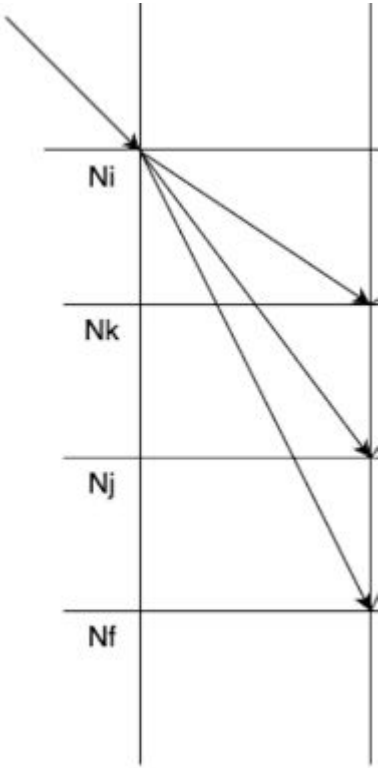


# $N_i$ becomes the primary

Communication type: Linear Communication (broadcasts to every node)

MSG sent: ⟨Collect, k⟩σNi

Now $N_i$ waits for responses from 2f + 1 nodes.



# $N_i$ becomes the primary

ANALOGY:

Imagine an invitation to a wedding.

Now you have to RSVP to primary.

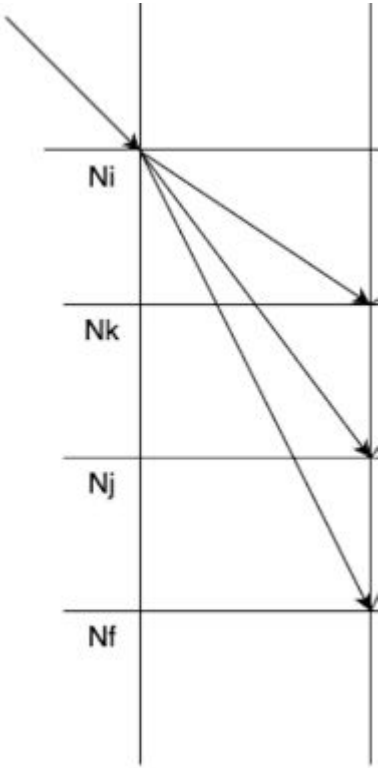
THE CATCH: Before you RSVP, you have to meet certain conditions.

$N_i$ sends invitations to every other node.

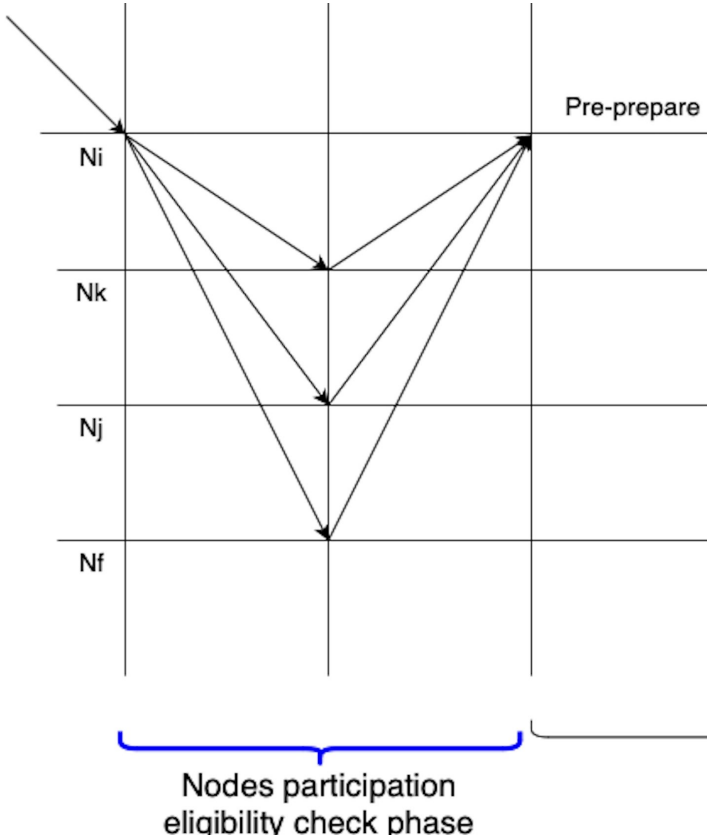
INVITATION

$[ts , ts']$

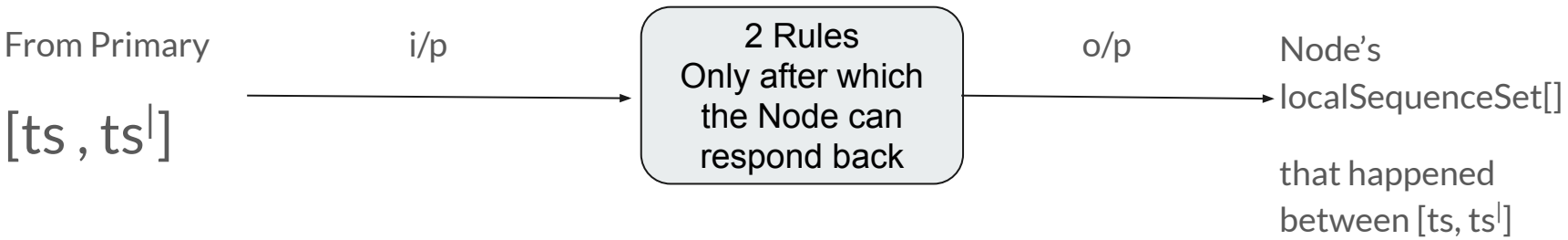
SEND TO ALL NODES



# The Participating Nodes



# Participating Node Overview:



# Node $N_j$ Participation: Rule 1

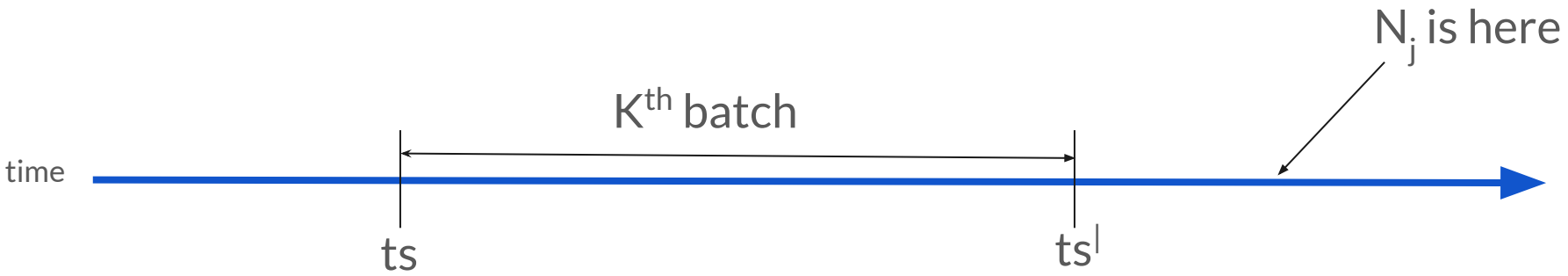
$N_j$'s GlobalSyncTs > ts'

# Node $N_j$ Participation: Rule 1

# $N_j$'s GlobalSyncTs > ts'

WHAT DOES THIS GUARANTEE ?

- It has reached the stage where it can fix that a set of commands it has in its local sequence set till ts'.
- Basically, it is ahead of ts' so now it can be a candidate to participate in consensus
- ATLEAST one more node, apart from $N_j$ which has experienced the time period [ts,ts']



# Node $N_j$ Participation: Rule 2

**After a Sync msg was received, a time period Δ ,**
**has elapsed on $N_j$'s timer.**

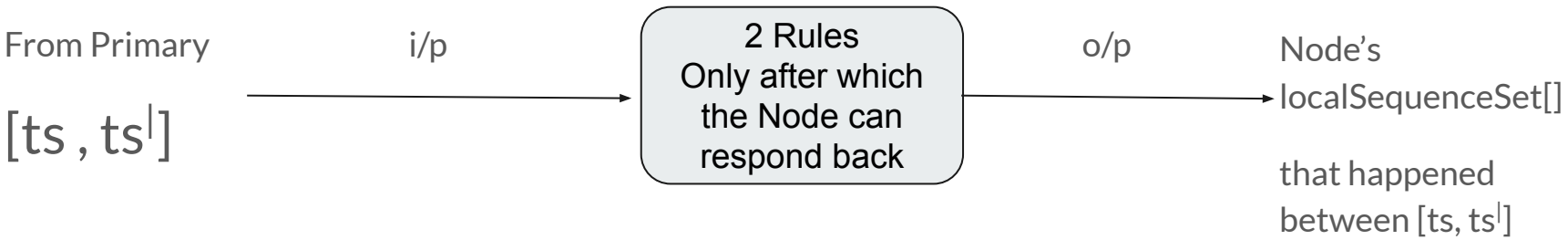
# Node $N_j$ Participation: Rule 2

**After a Sync msg was received, a time period Δ, has elapsed on $N_j$'s timer.**

WHAT DOES THIS GUARANTEE ?

- Guarantee for presence of 2f+1 Nodes.
- This delta is a buffer time which gives all correct nodes enough time to obtain assigned time stamp and create sequence for commands before ts' for localSequnceSet.
  So that $N_j$'s localTreshholdTs changes, the system would not accept anything after delta

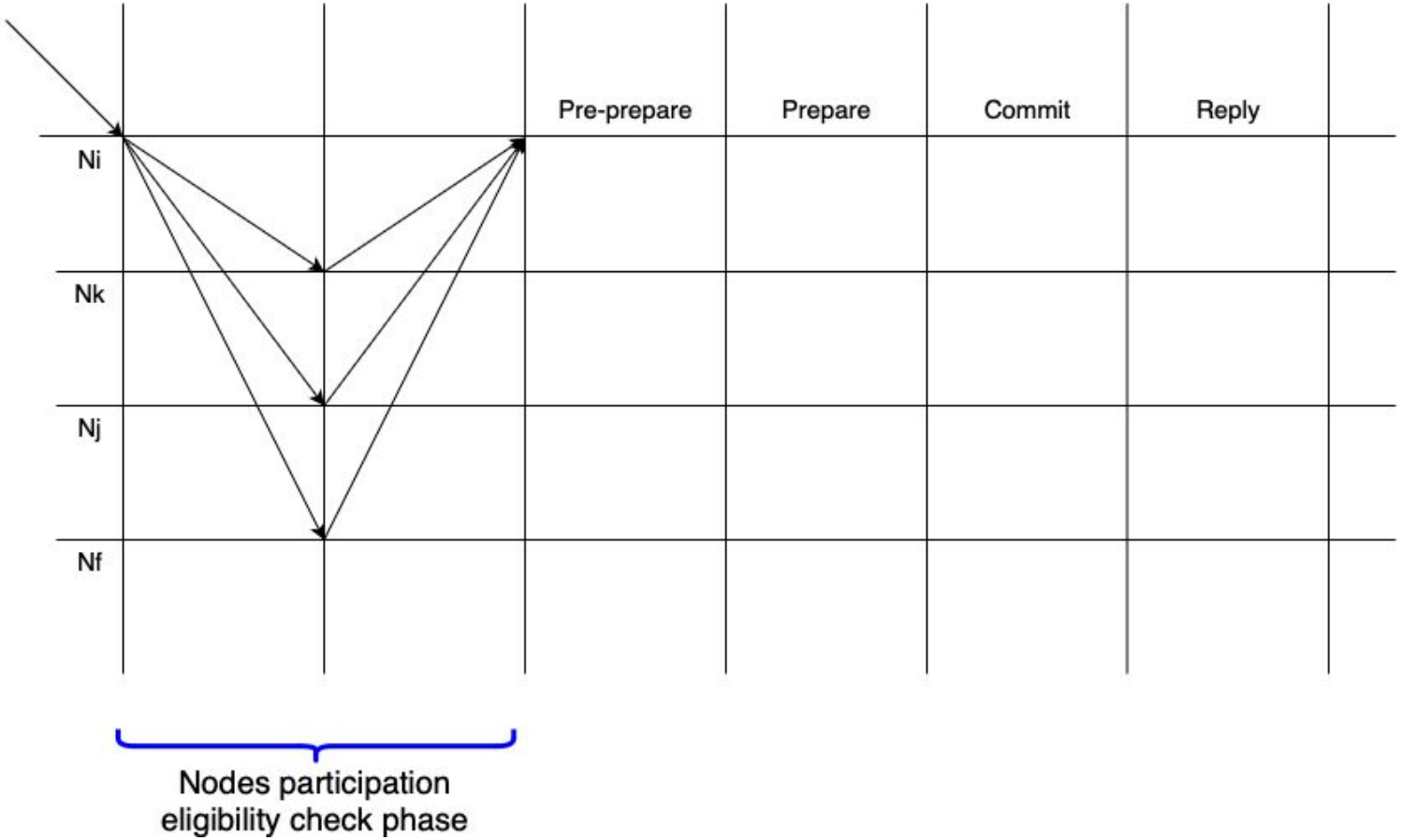
# Participating Node Overview:



# Consensus, but on what ?

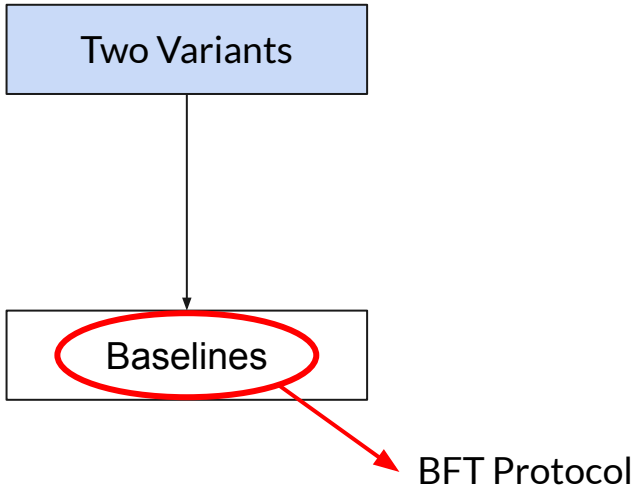
$N_i$ will now run consensus to agree upon U.

U is the union of  localSequenceSet of atleast 2f+1 nodes for consensus slot k.

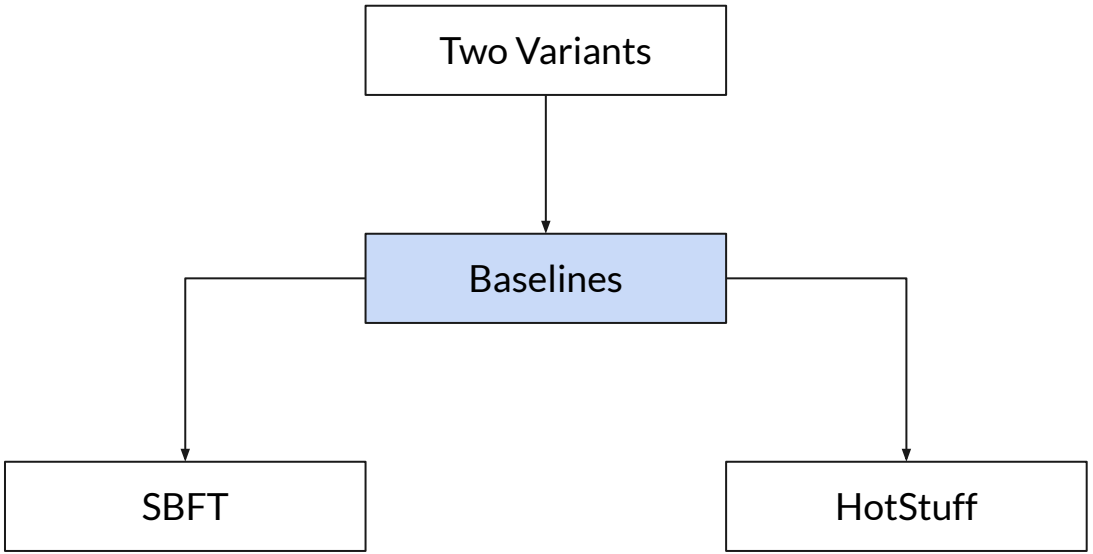

Pre-prepare
Prepare
Commit
Reply
Ni
Nk
Nj
Nf
Nodes participation
eligibility check phase

# Implementation & Experimental Evaluation

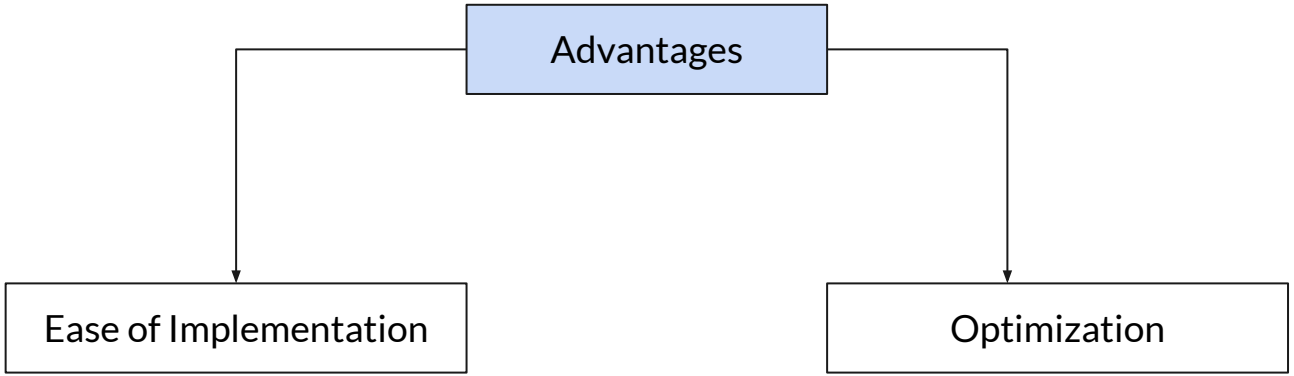
# Implementation



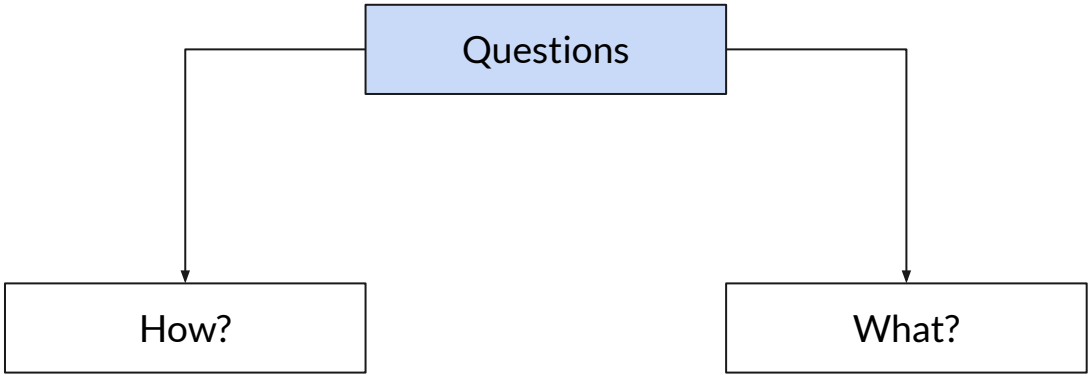
# Implementation



# Implementation



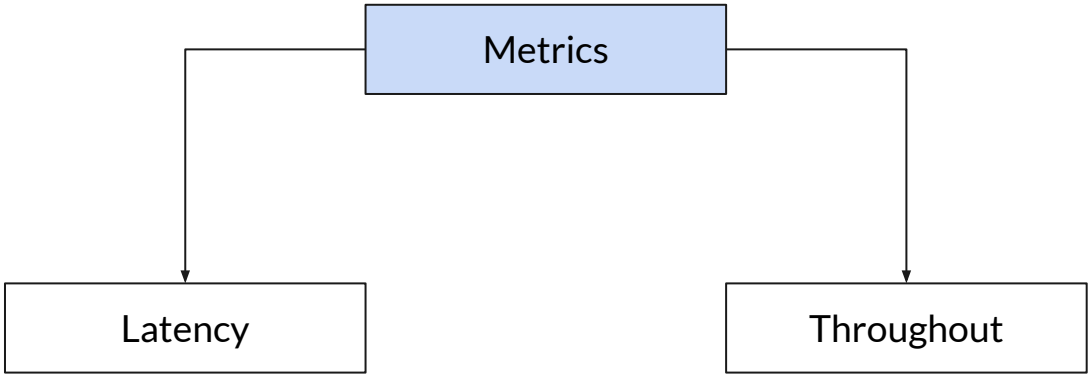
# Experimental Evaluation



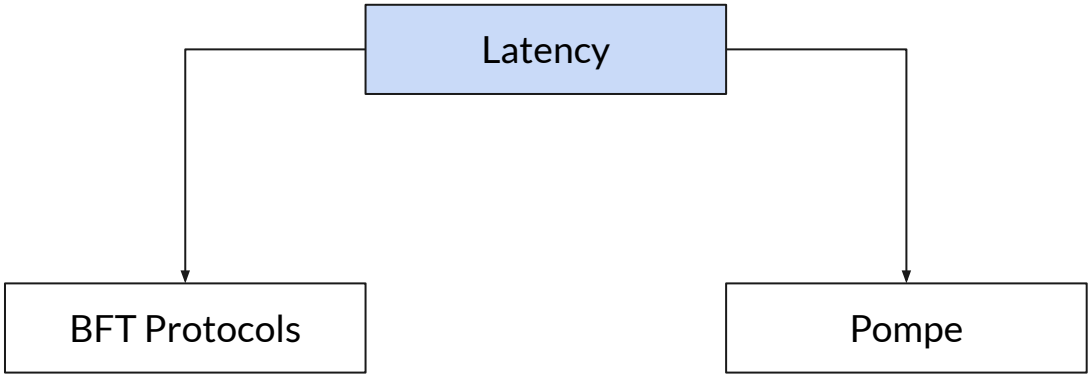

How to compare performance?

What is the impact on performance?

# Experimental Evaluation

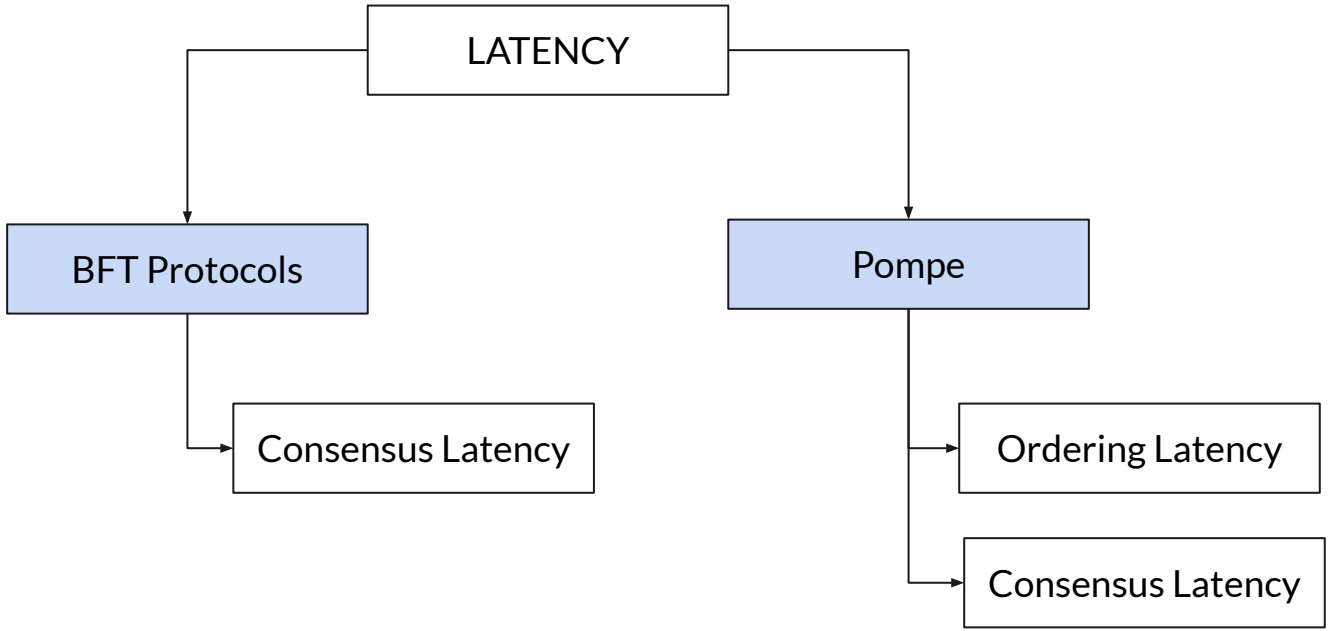


# Experimental Evaluation




How is it different?

# Experimental Evaluation



# Experimental Evaluation

- End-to-end Performance Test
  - Four Node Configuration
  - With and Without Batching

# Experimental Evaluation

| | throughput (cmds/s) | median latency (ms) |
|---|---|---|
| HotStuff ($\beta_c = 1$) | 474 | 8.2 |
| HotStuff ($\beta_c = 800$) | 253,360 | 49.9 |
| Pompē-HS ($\beta_o = 1$) | 1,642 | 2.3 (o), 47.7 (c) |
| Pompē-HS ($\beta_o = 200$) | 361,687 | 5.7 (o), 53.1 (c) |
| Concord ($\beta_c = 1$) | 40 | 53 |
| Concord ($\beta_c = 800$) | 6,633 | 67 |
| Pompē-C ($\beta_o = 1$) | 1,415 | 17 (o), 67 (c) |
| Pompē-C ($\beta_o = 200$) | 249,221 | 18 (o), 74 (c) |

- Baseline

  β = S (>1)

- Pompe

  β = S/n

REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

# Experimental Evaluation

| | throughput (cmds/s) | median latency (ms) |
|---|---|---|
| HotStuff ($\beta_c = 1$) | 474 | 8.2 |
| HotStuff ($\beta_c = 800$) | 253,360 | 49.9 |
| Pompē-HS ($\beta_o = 1$) | 1,642 | 2.3 (o), 47.7 (c) |
| Pompē-HS ($\beta_o = 200$) | 361,687 | 5.7 (o), 53.1 (c) |
| Concord ($\beta_c = 1$) | 40 | 53 |
| Concord ($\beta_c = 800$) | 6,633 | 67 |
| Pompē-C ($\beta_o = 1$) | 1,415 | 17 (o), 67 (c) |
| Pompē-C ($\beta_o = 200$) | 249,221 | 18 (o), 74 (c) |

O -> Ordering

C -> Consensus

REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

# Experimental Evaluation

- Performance Test on Geo-distributed Setup
  - Four Node Configuration
  - Three data centers

# Experimental Evaluation

| | throughput (cmds/s) | median latency (ms) |
|---|---|---|
| HotStuff ($\beta_c = 1$) | 474 | 8.2 |
| HotStuff ($\beta_c = 800$) | 253,360 | 49.9 |
| Pompē-HS ($\beta_o = 1$) | 1,642 | 2.3 (o), 47.7 (c) |
| Pompē-HS ($\beta_o = 200$) | 361,687 | 5.7 (o), 53.1 (c) |

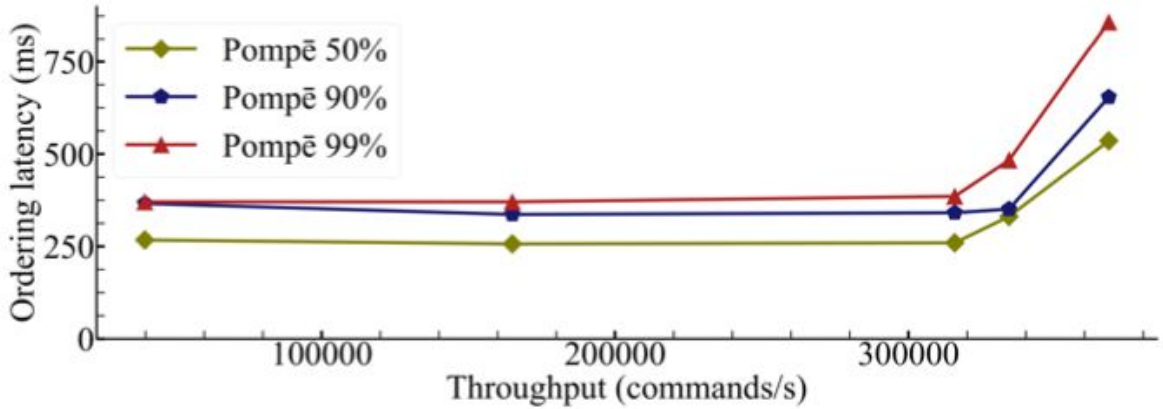
| | throughput (cmds/s) | median latency (ms) |
|---|---|---|
| HotStuff ($\beta_c = 800$) | 6,160 | 915.8 |
| Pompē-HS ($\beta_o = 200$) | 315,753 | 259.7 (o), 1518.1 (c) |
| Concord ($\beta_c = 800$) | 1,461 | 616 |
| Pompē-C ($\beta_o = 200$) | 172,774 | 325 (o), 1415 (c) |

Drastic drop

Use of rotating leaders

REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

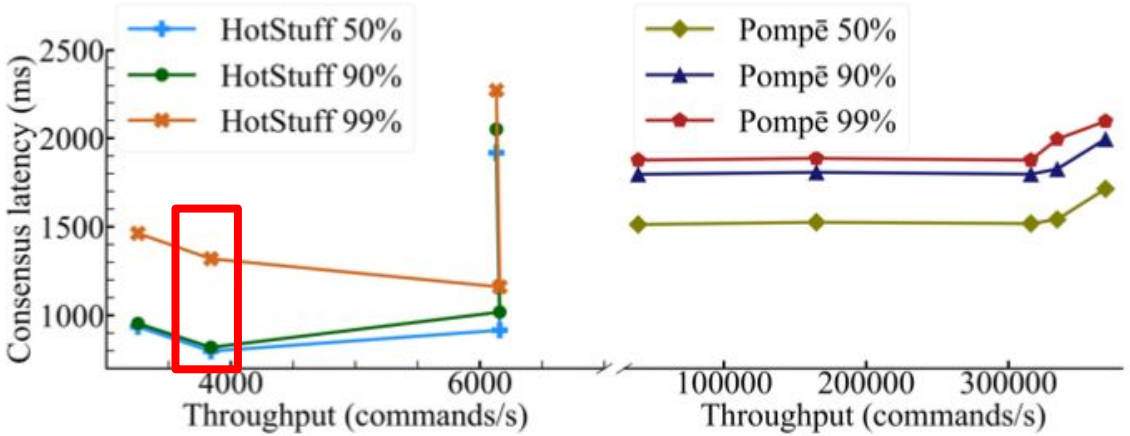
# Experimental Evaluation




Higher throughput at the cost of higher consensus latencies

REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

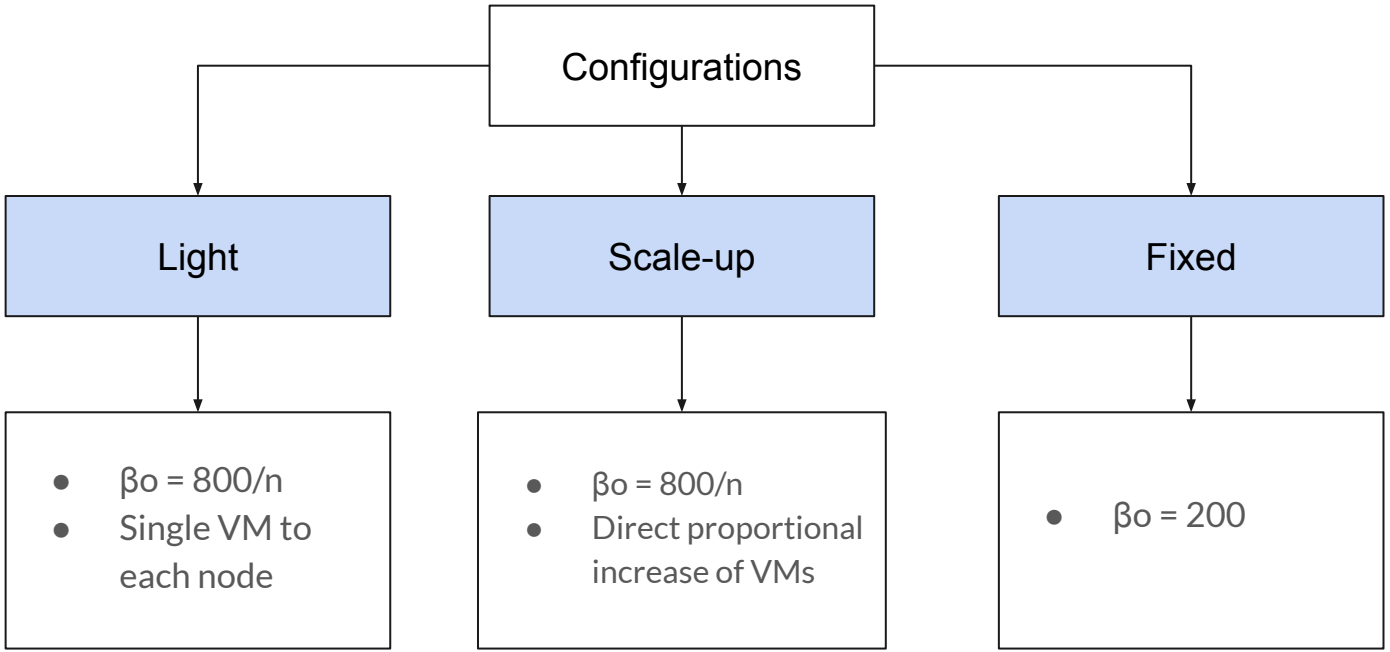
# Experimental Evaluation



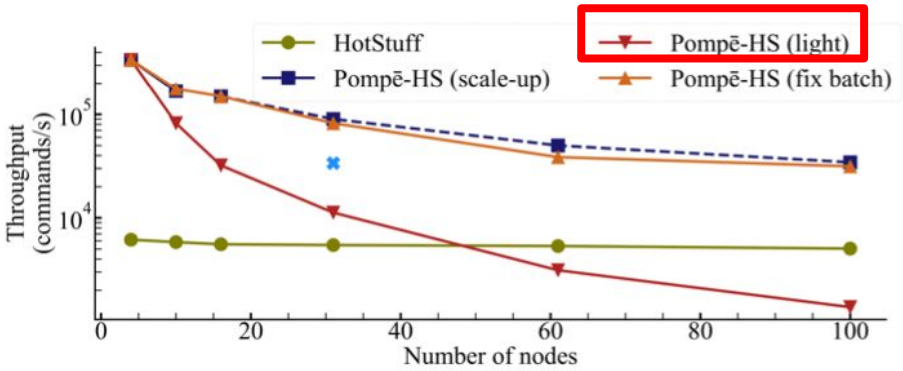

Initial drop due to more clients filling batches

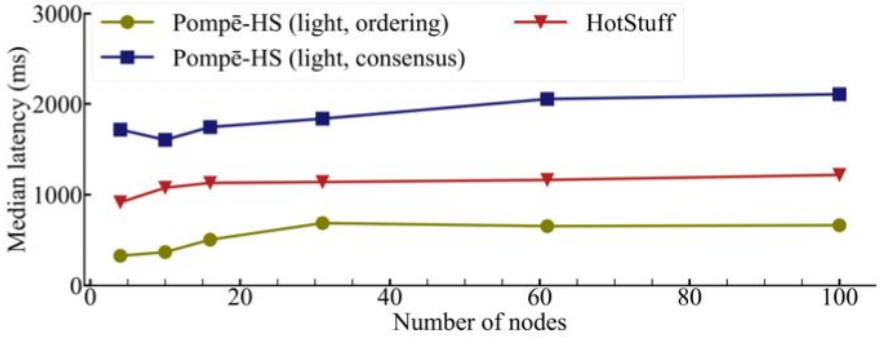
REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

# Scalability



# Scalability




Batch sizes inverse proportional to n, so throughput degrades as n increases

REF:Zhang, Y., Setty, S., Chen, Q., Zhou, L., & Alvisi, L. (n.d.). Byzantine Ordered Consensus without Byzantine Oligarchy | USENIX. Byzantine Ordered Consensus Without Byzantine Oligarchy | USENIX. Retrieved November 14, 2022, from

# Thank You!

*Any Questions?*