DCL

EPFL

Straggler Resilient Practical BFT via Blind Agreement

Mohamed Yassine Boukhari

Project Supervisor

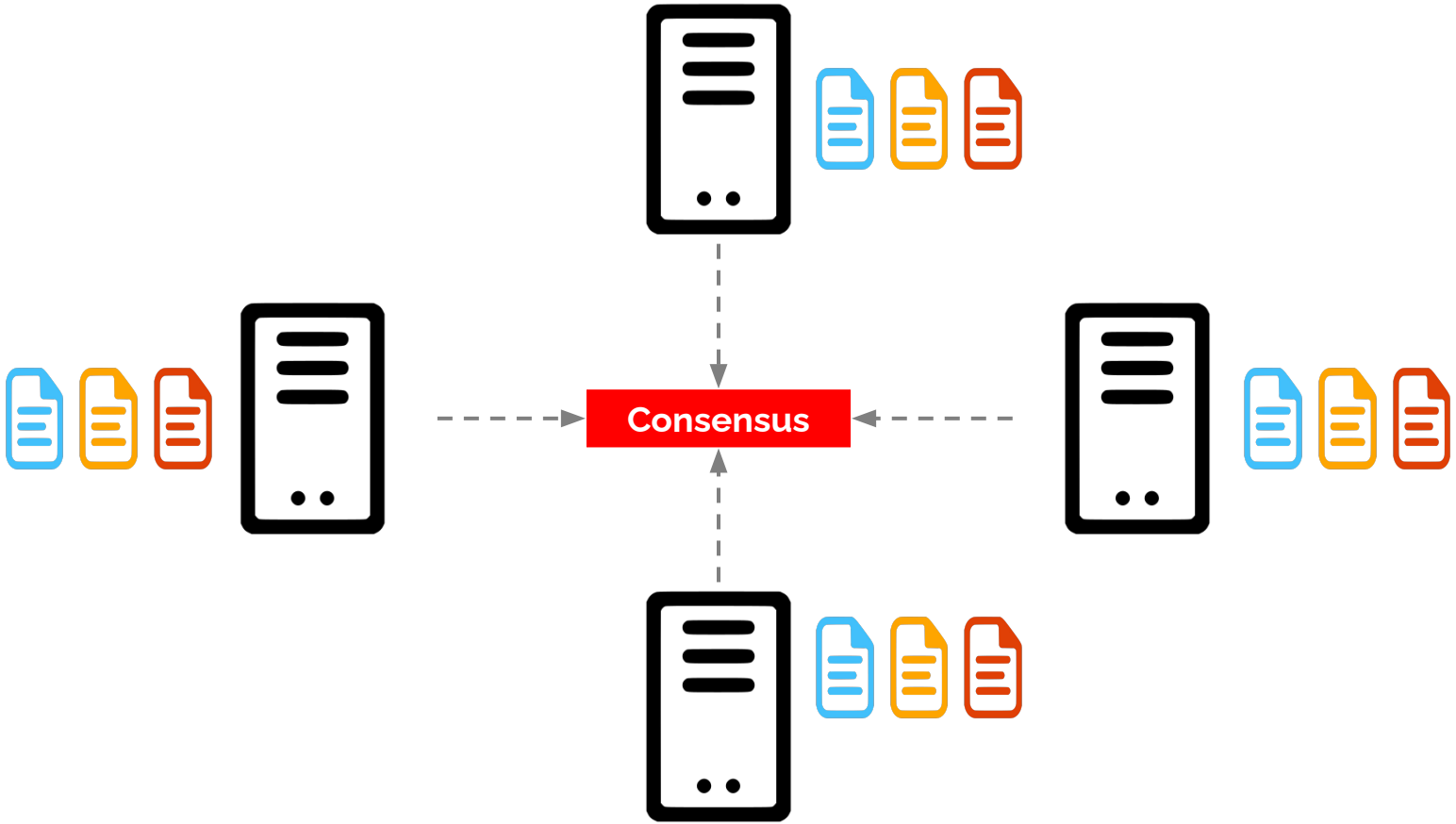
Dr. Gauthier Voron

Project Advisor

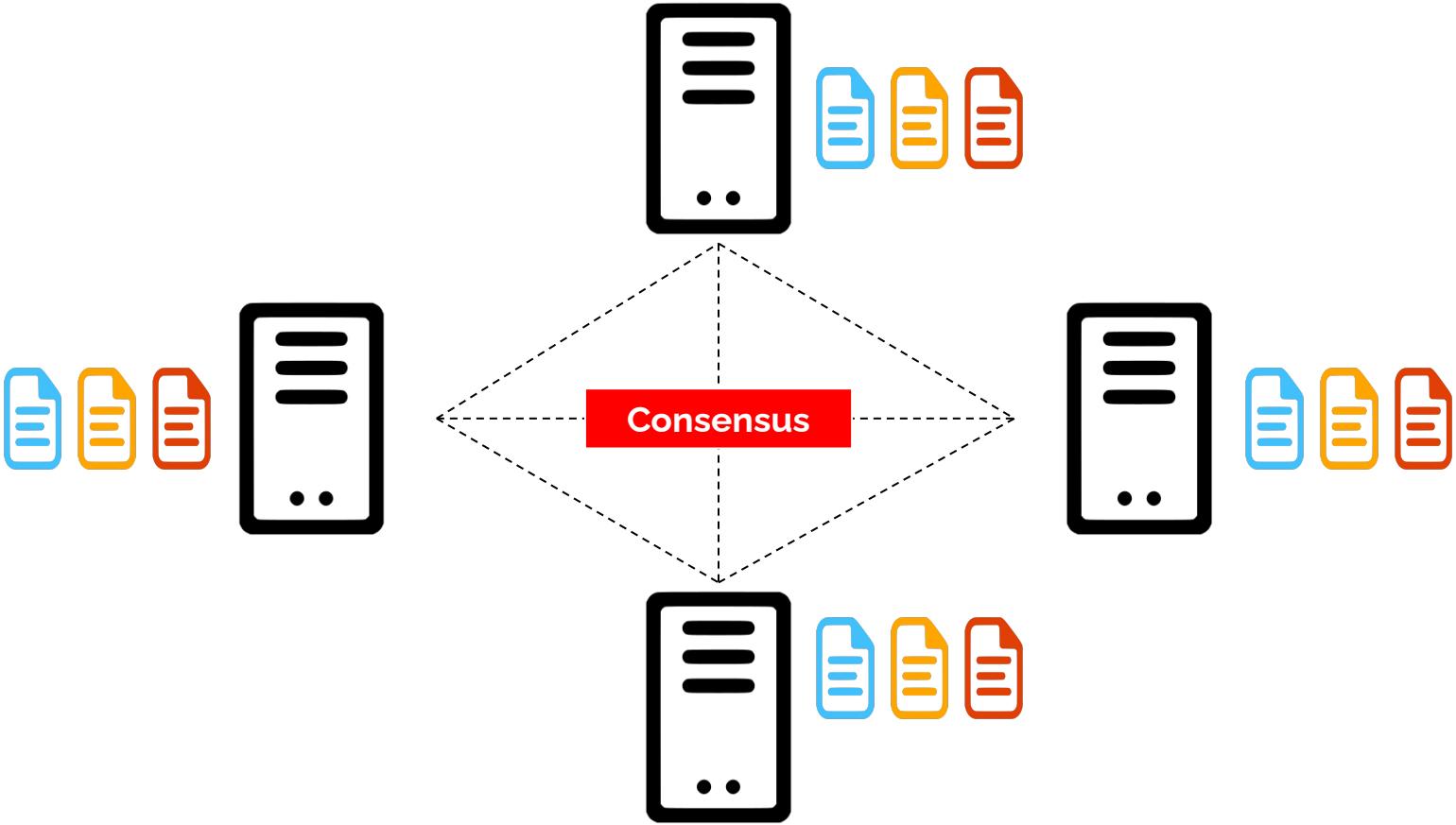
Prof. Rachid Guerraoui

BFT Consensus

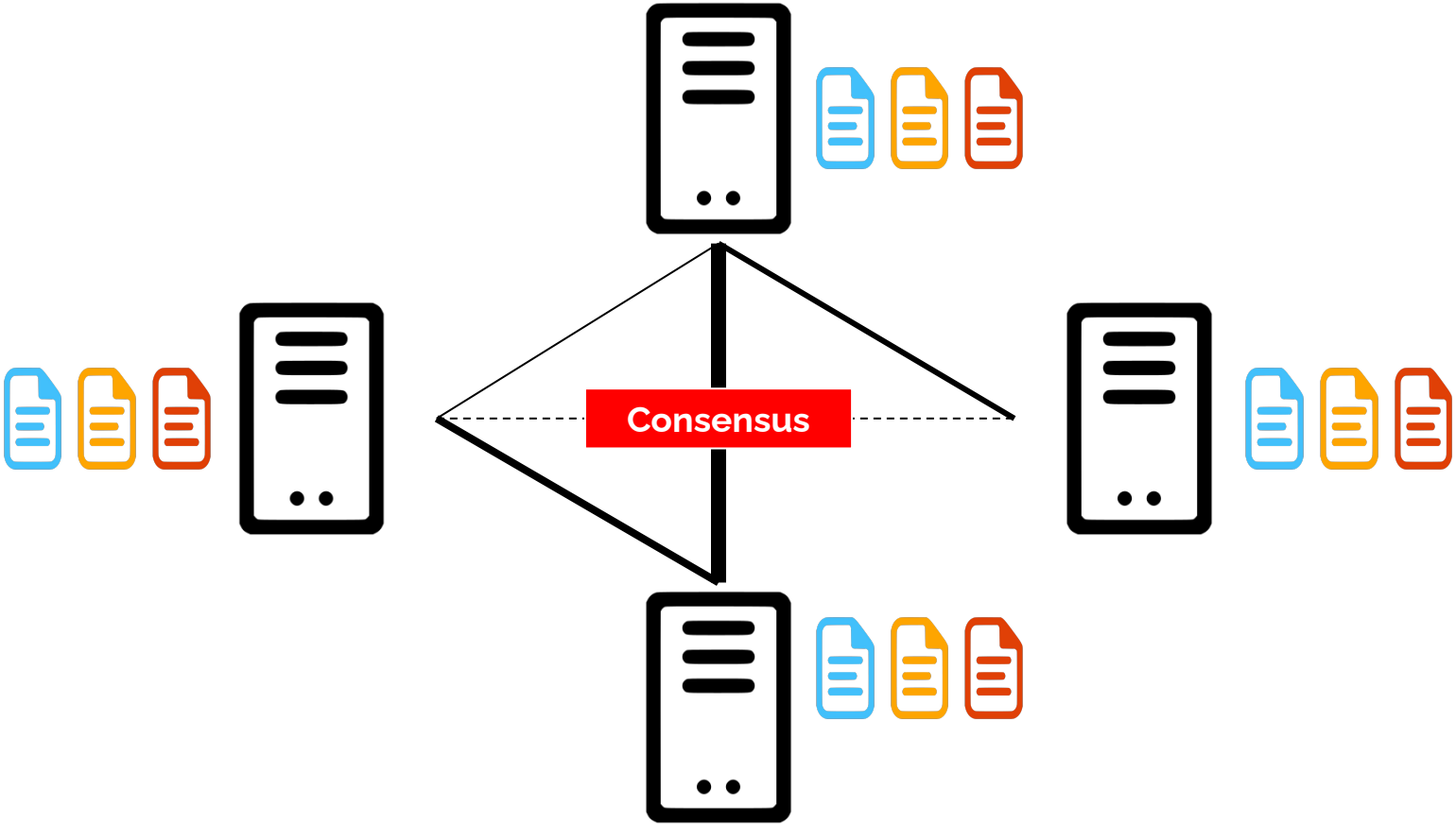
BFT Consensus



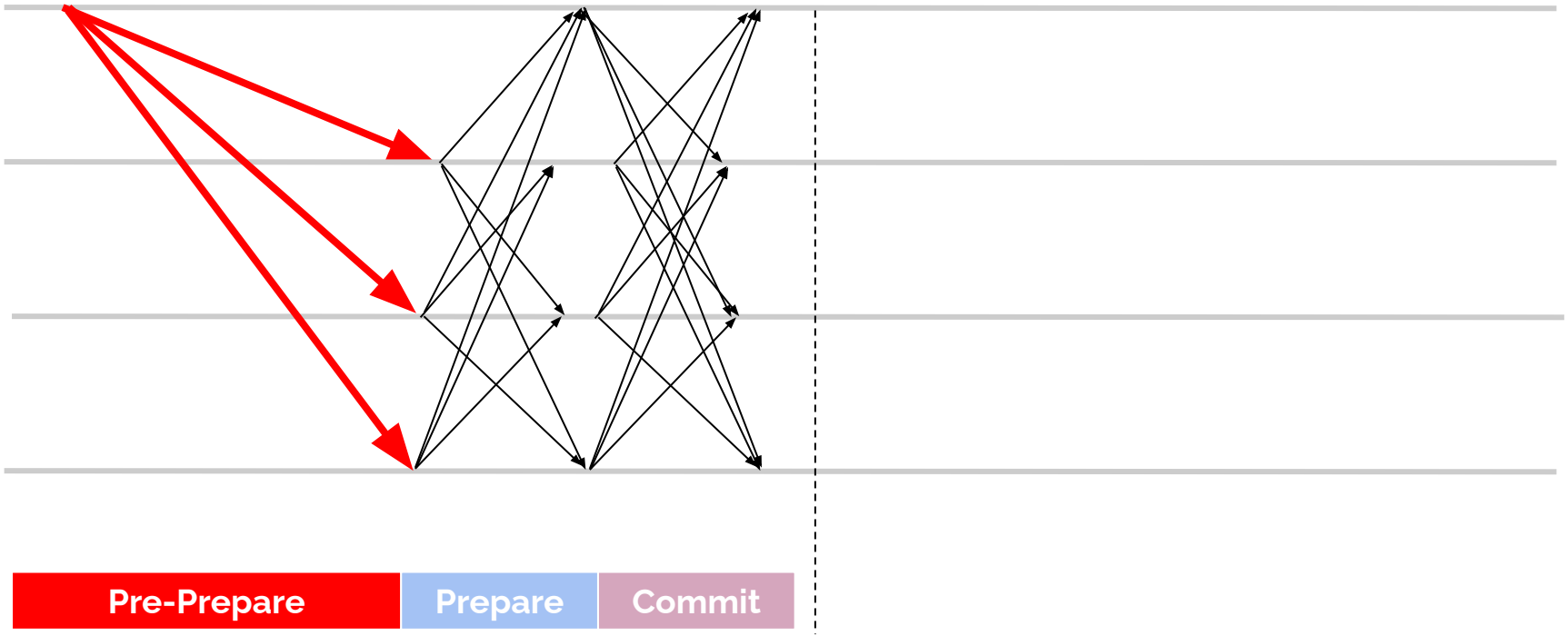
Expectations



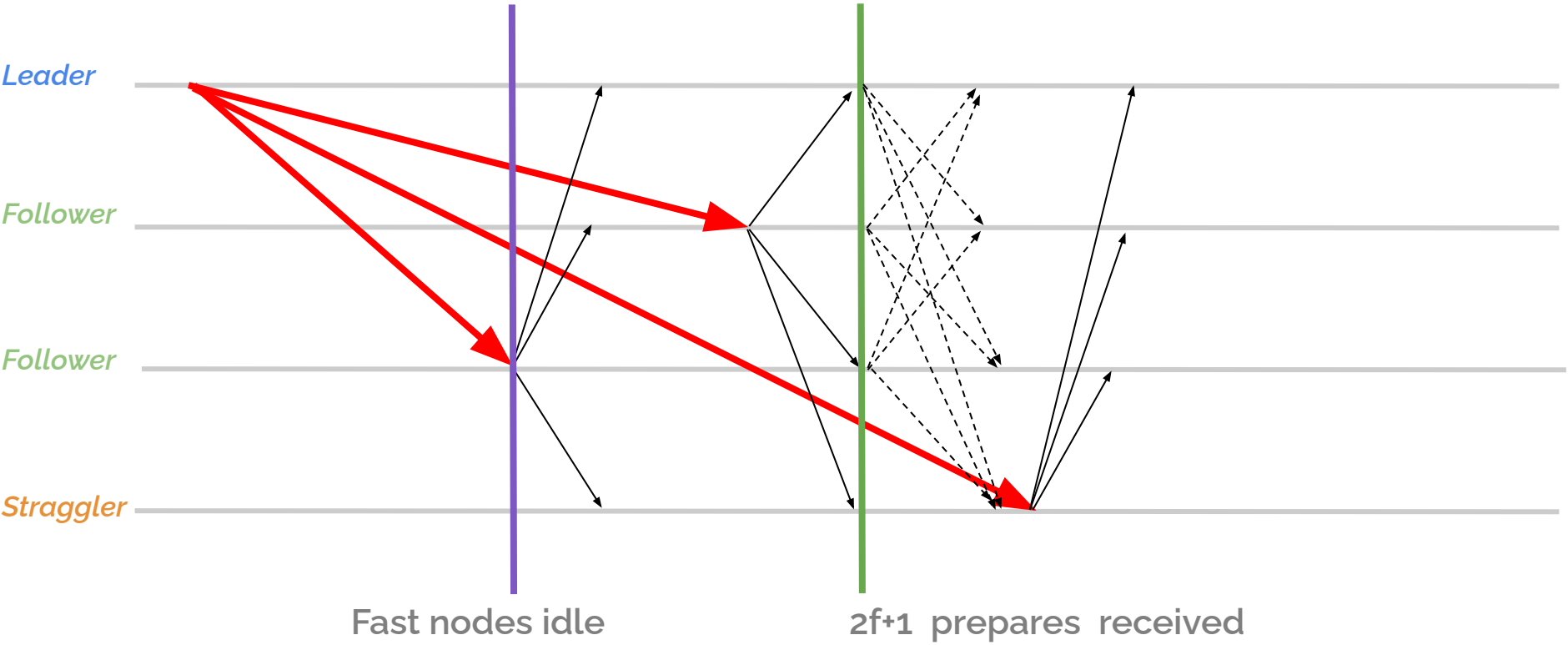
Reality

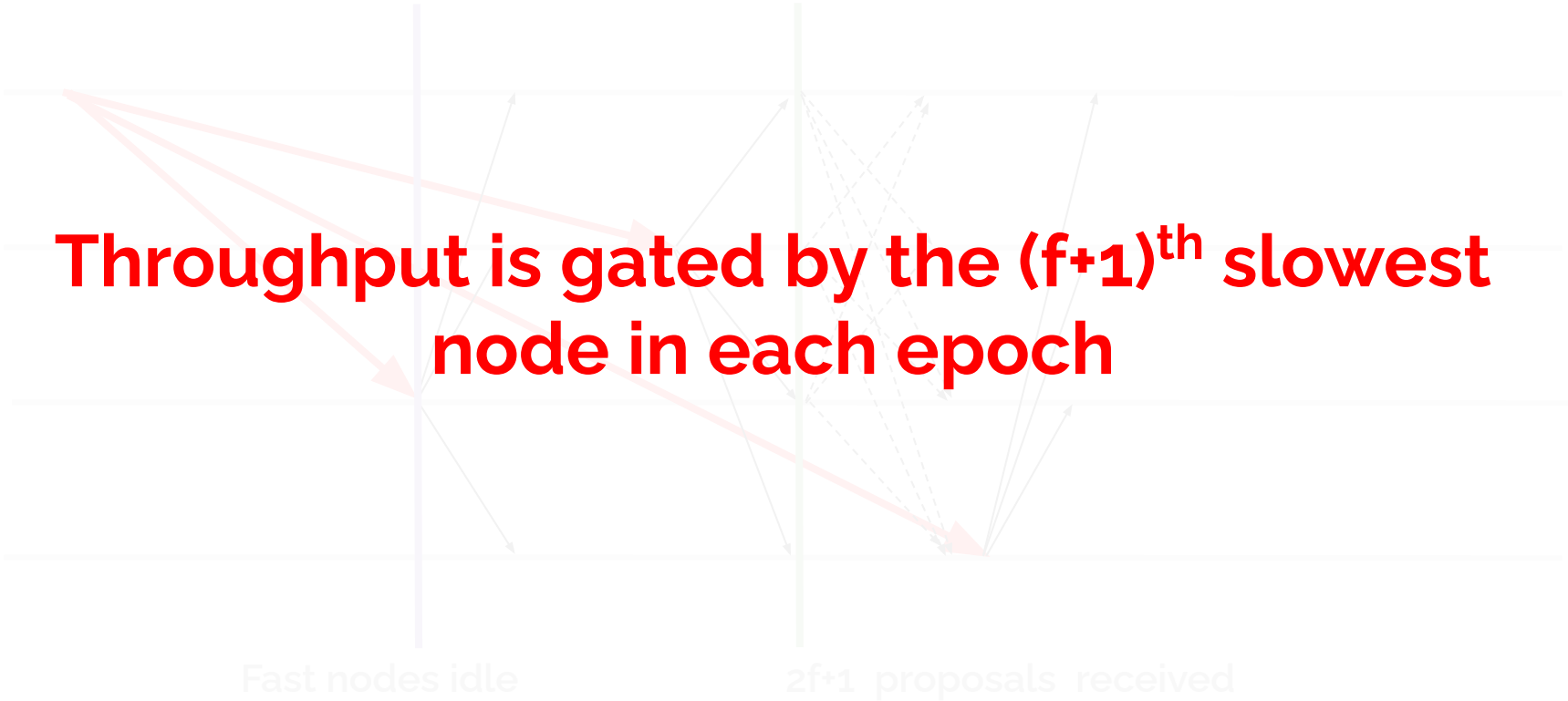


Existing BFT Protocols - PBFT



Heterogeneous Network Bandwidths





Throughput is gated by the $(f+1)^{th}$ slowest node in each epoch

Contributions

Contributions

- How to prototype and simulate BFT protocols on heterogeneous networks

Contributions

- How to prototype and simulate BFT protocols on heterogeneous networks ?

- How to make the slowest nodes vote in the consensus without slowing its progress

PAXI BFT

- Easy prototyping of BFT protocols
- Benchmarking
- Fill in three GO modules:
  - bft.go
  - replica.go
  - messages.go

PAXI BFT

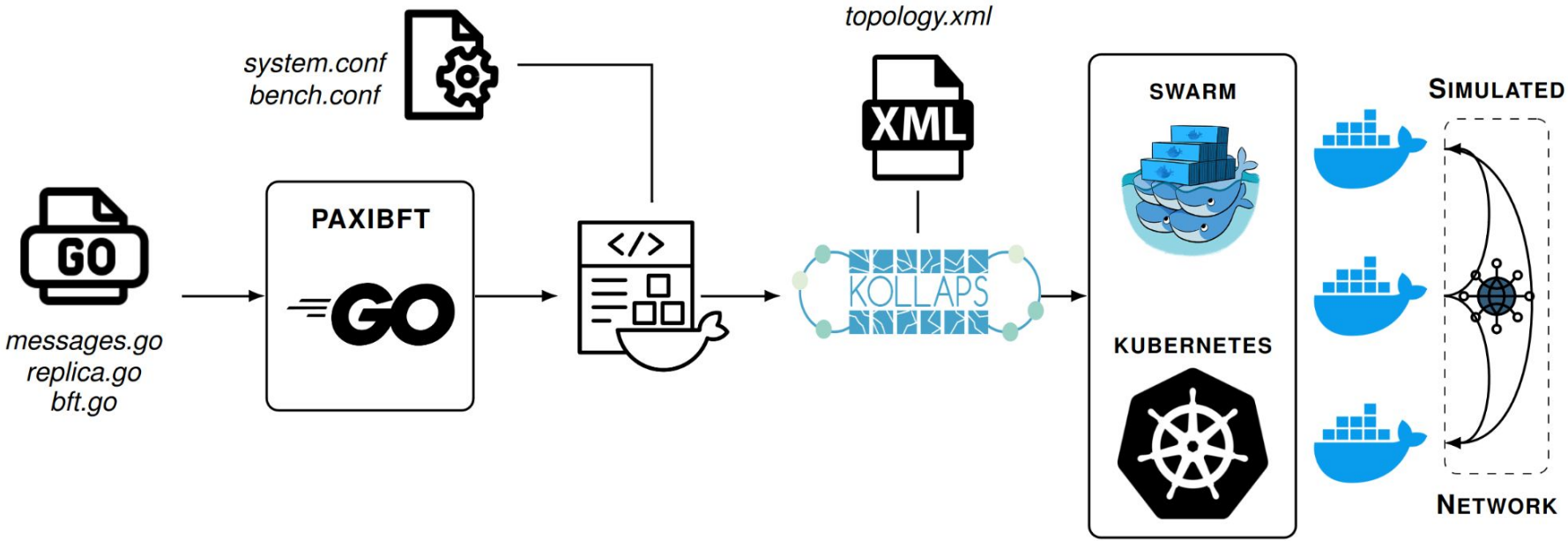
- Easy prototyping of BFT protocols
- Benchmarking
- Fill in three GO modules:
  - bft.go
  - replica.go
  - messages.go

PAXI BFT

- Easy prototyping of BFT protocols
- Benchmarking
- Fill in three GO modules:
  - bft.go
  - replica.go
  - messages.go

- Network simulation tool
- Agnostic of application language
- Easy customization of end-to-end properties (bandwidth, delay, jitter ...)
- Dynamic events such as node crashes or link removals

EASY BFT PRO SIM



Observations

Observations

- Nodes can blindly vote on a proposal if they have a proof that it is correct

Observations

- Nodes can blindly vote on a proposal if they have a proof that it is correct

- 2f+1 votes are required for Agreement but only f+1 needed for Correctness

~~Observations~~ *Idea*

- Nodes can blindly vote on a proposal if they have a proof that it is correct

- 2f+1 votes are required for Agreement but only f+1 needed for Correctness

→ Blind Agreement: Stragglers see only the hash of the request and wait for (f+1) votes to accept the proposal

Straggler Resilient BFT

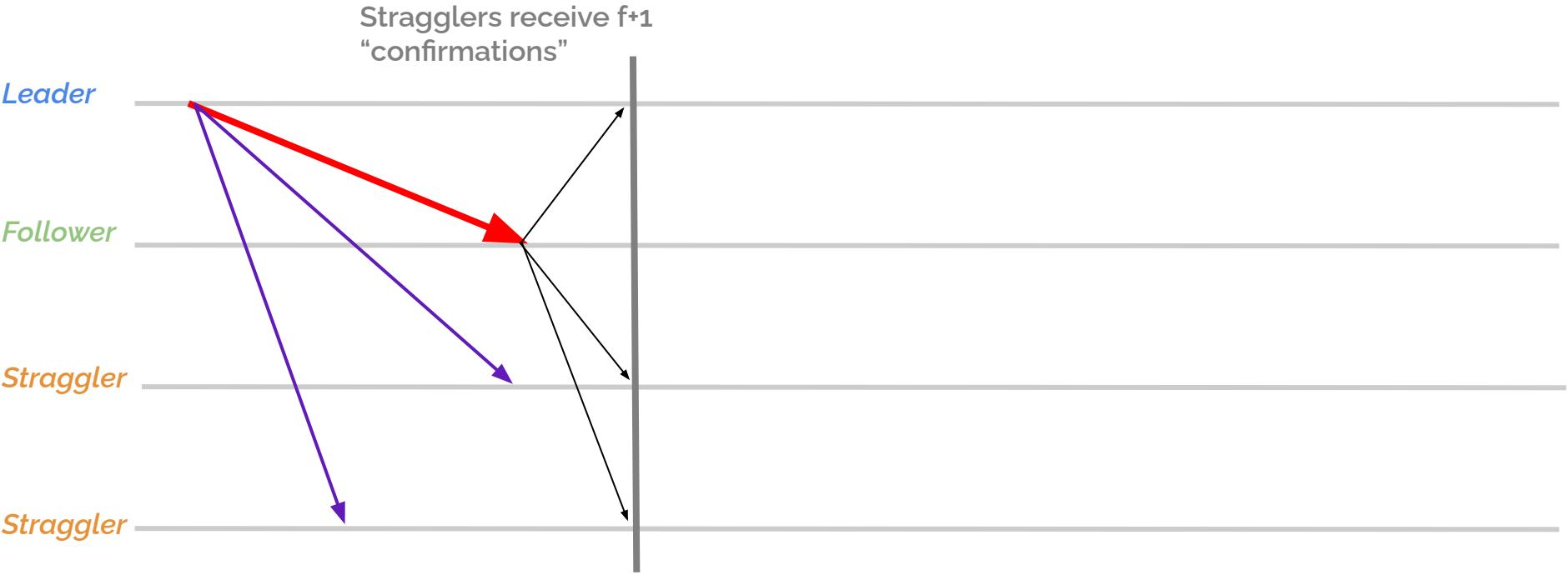
F-Pre-Prepare <V, SEQ, H, Request >

Straggler Resilient BFT

F-Pre-Prepare <V, SEQ, H, Request >

L-Pre-Prepare <V, SEQ, H, >

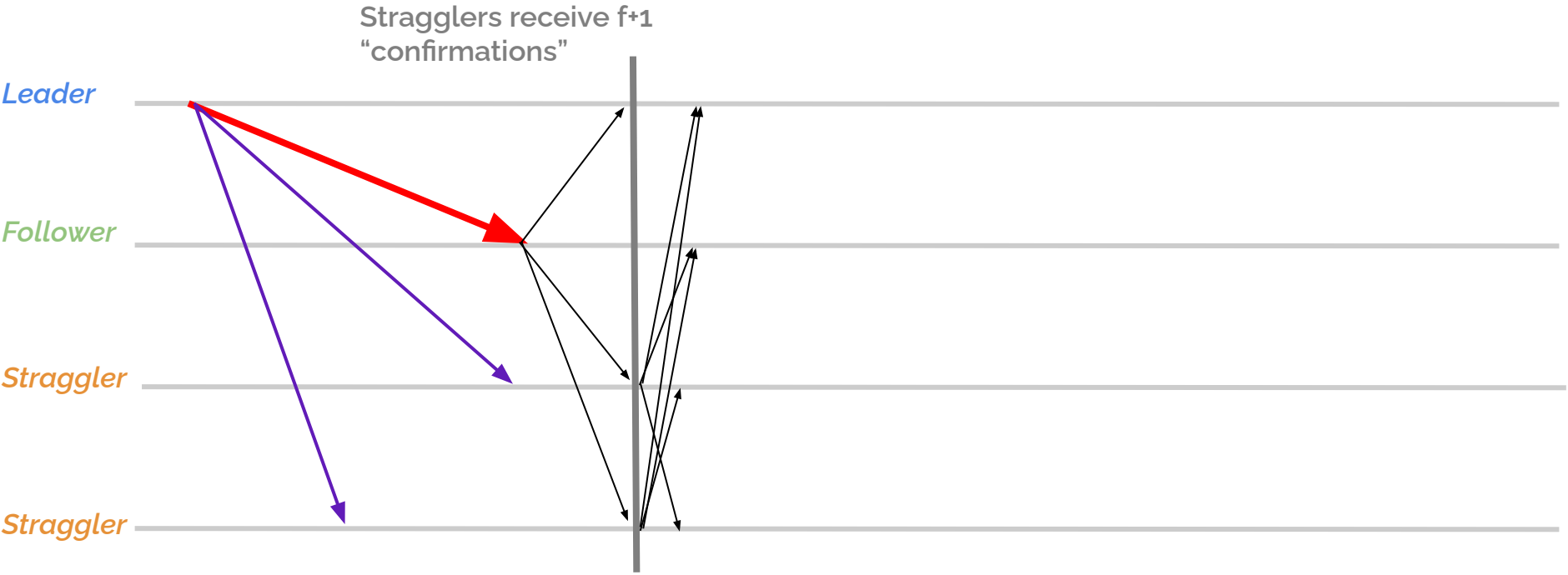
Straggler Resilient BFT



F-Pre-Prepare <V, SEQ, H, Request>

L-Pre-Prepare <V, SEQ, H>

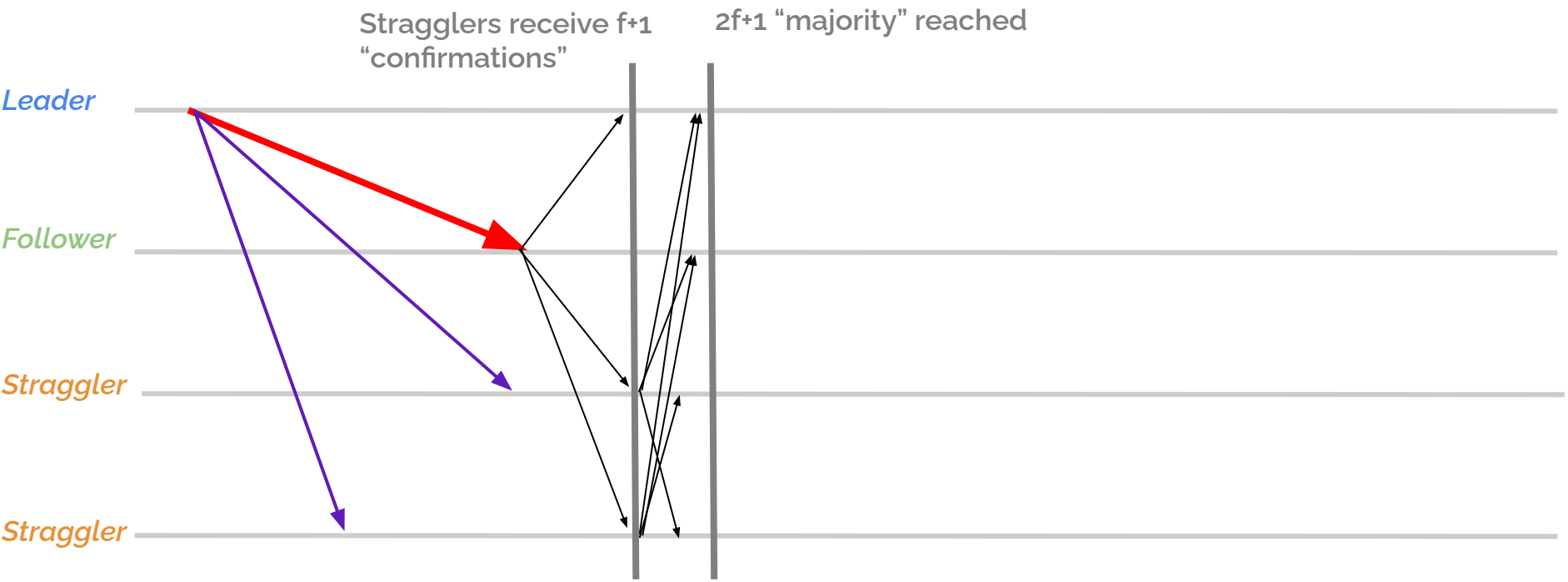
Straggler Resilient BFT



F-Pre-Prepare <V, SEQ, H, Request>

L-Pre-Prepare <V, SEQ, H>

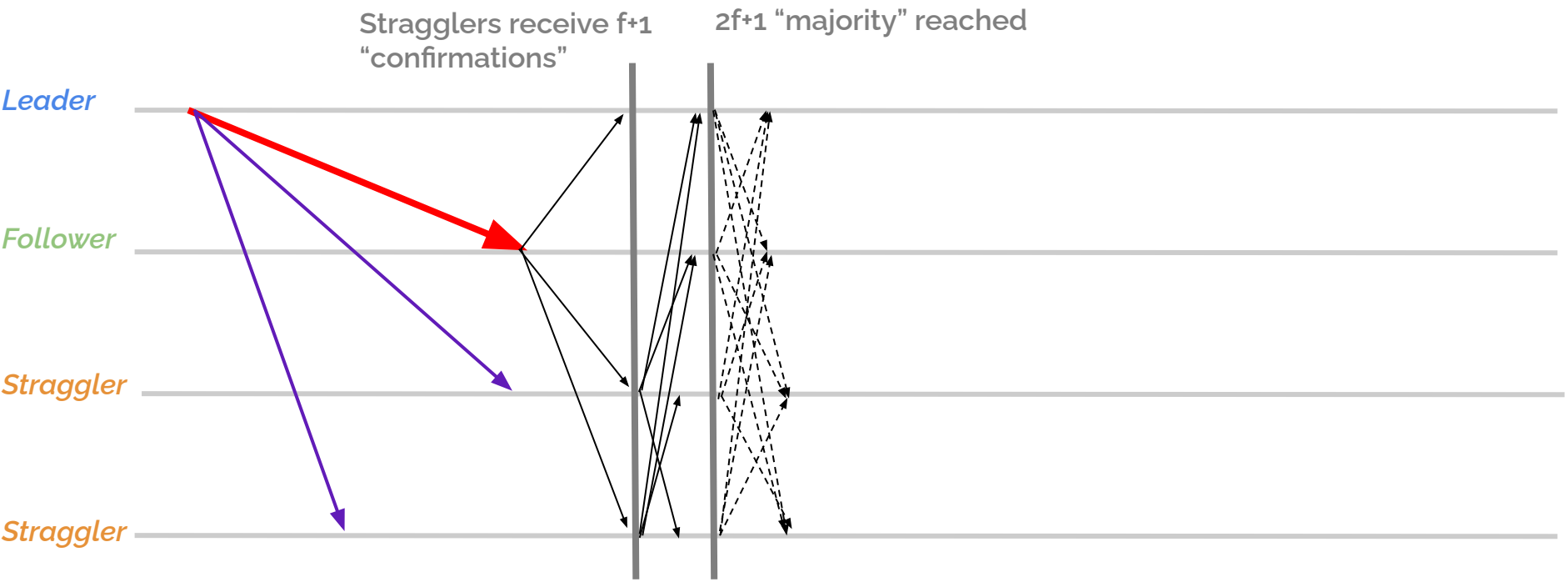
Straggler Resilient BFT



F-Pre-Prepare <V, SEQ, H, Request>

L-Pre-Prepare <V, SEQ, H>

Straggler Resilient BFT



F-Pre-Prepare <V, SEQ, H, Request>

L-Pre-Prepare <V, SEQ, H>

SR-BFT

In a System of $N = 3f + 1$ nodes

SR-BFT

In a System of $N = 3f + 1$ nodes

$H \geq f$ fast honest nodes

SR-BFT

In a System of $N = 3f + 1$ nodes

$H \geq f$ fast honest nodes

$S \leq f$ honest stragglers

SR-BFT

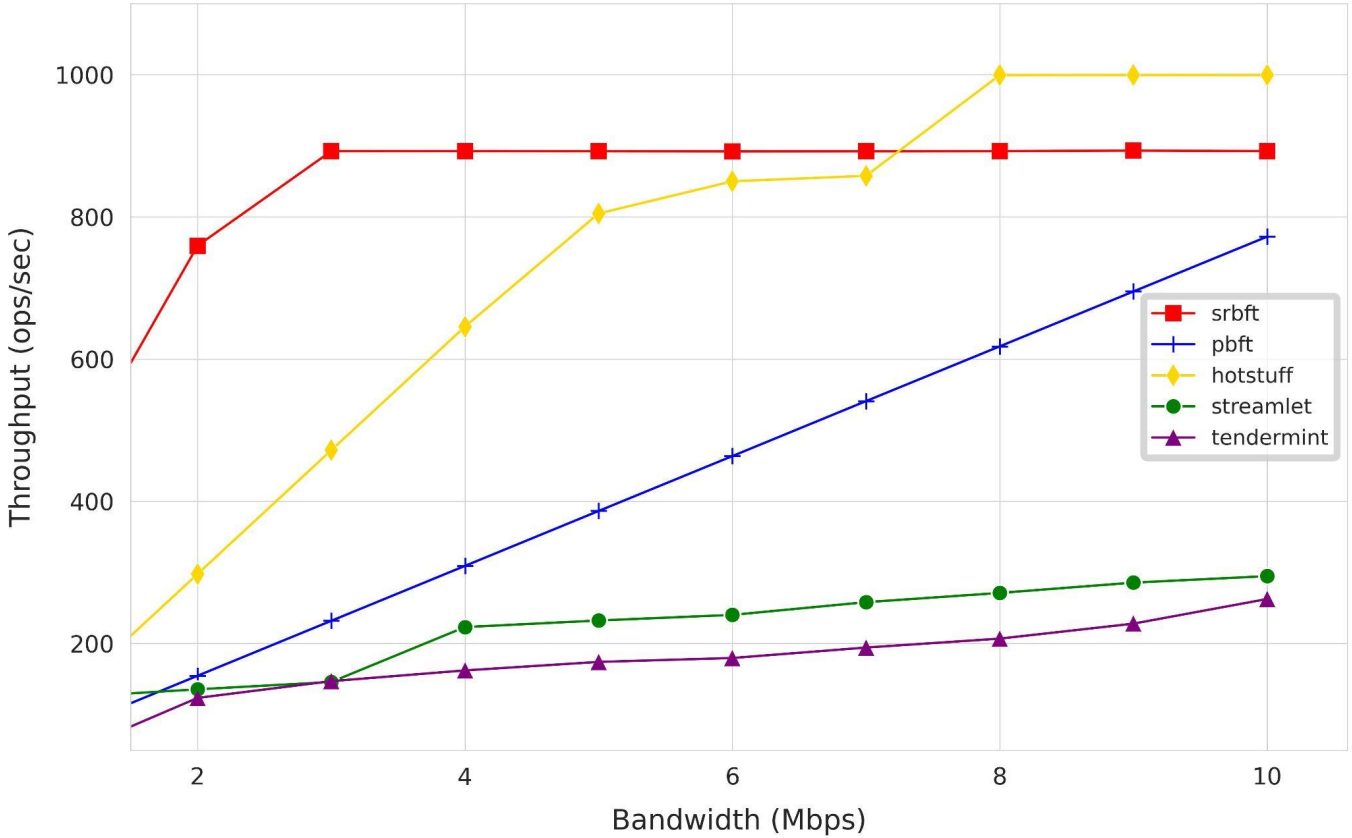
In a System of $N = 3f + 1$ nodes

$H \geq f$ fast honest nodes

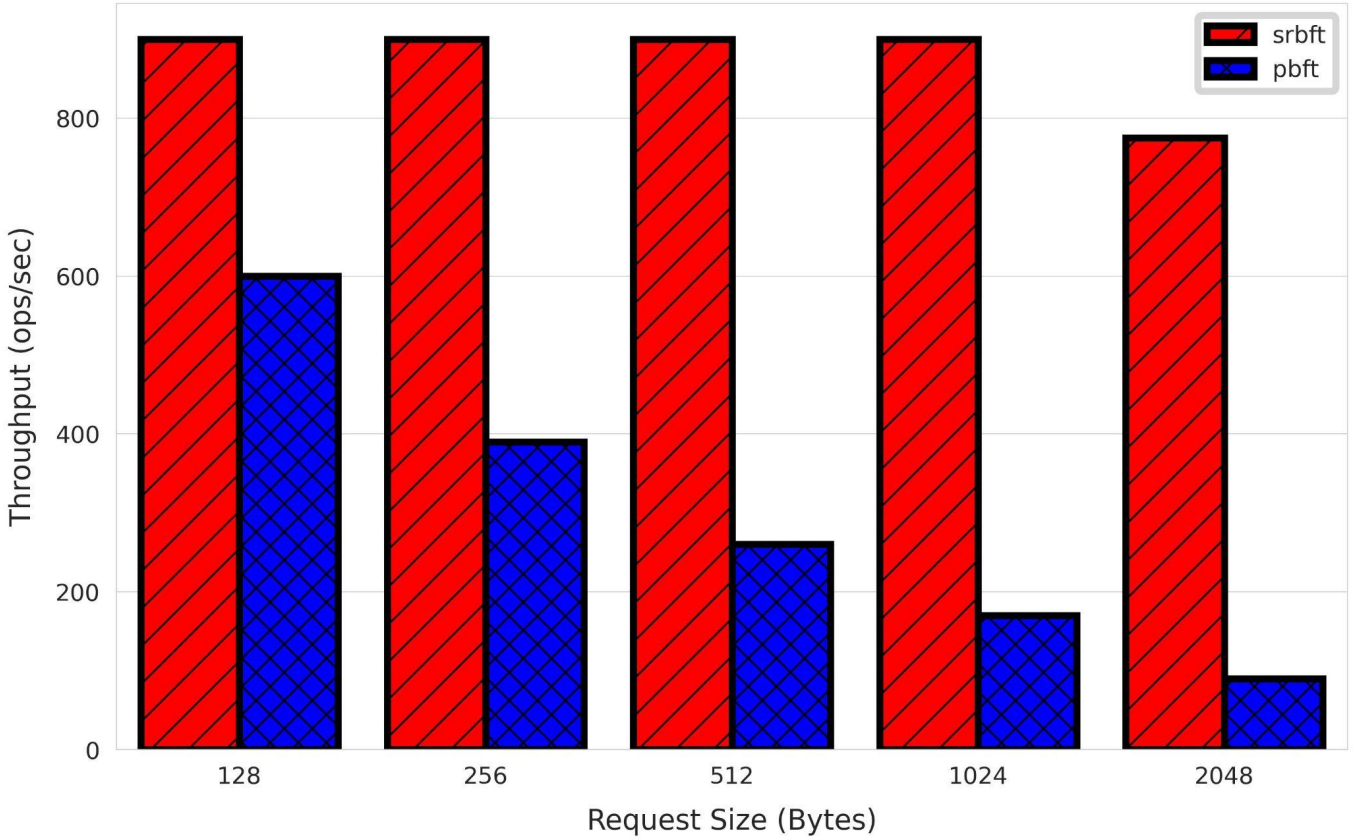
$B \leq f$ byzantines

$S \leq f$ honest stragglers

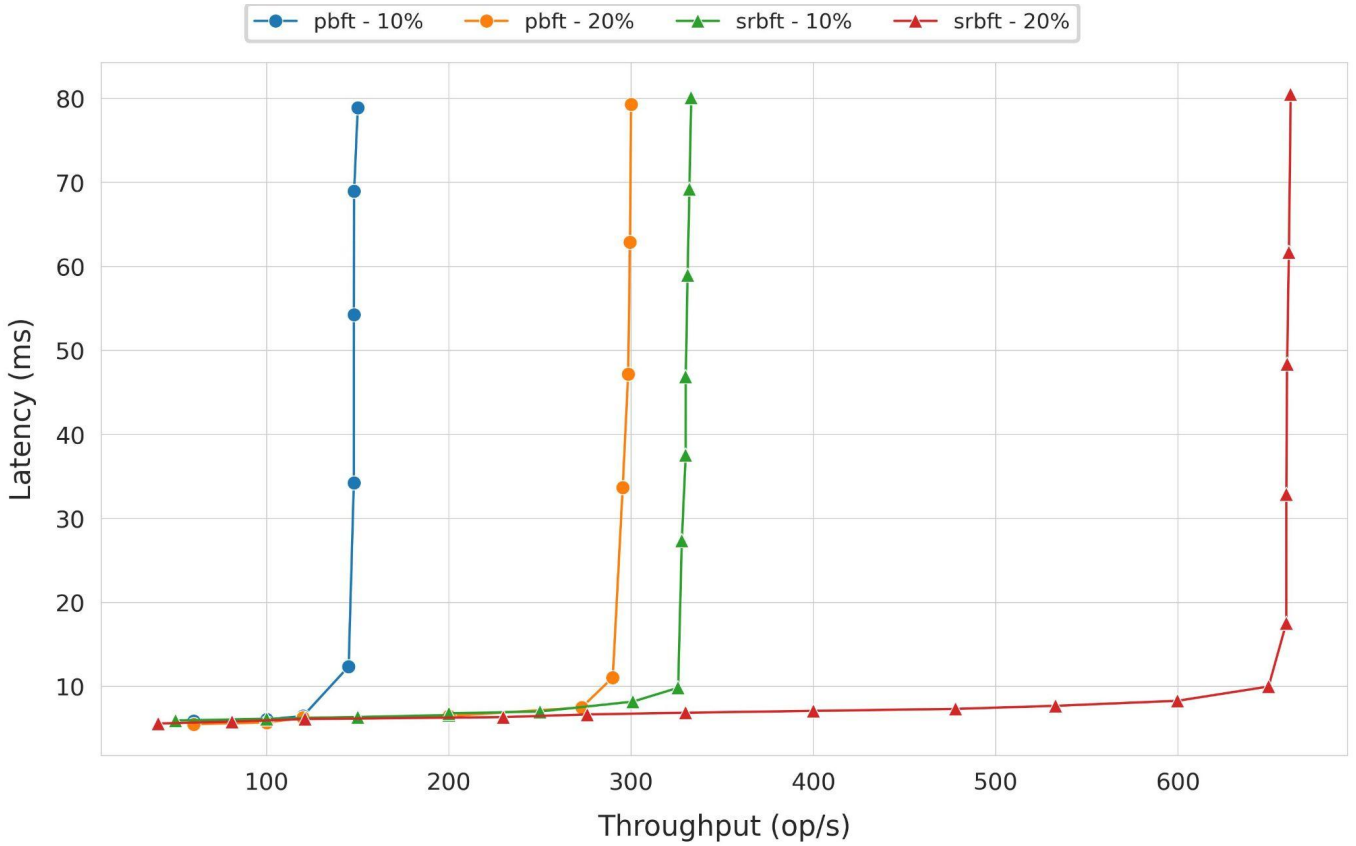
Throughput Comparison

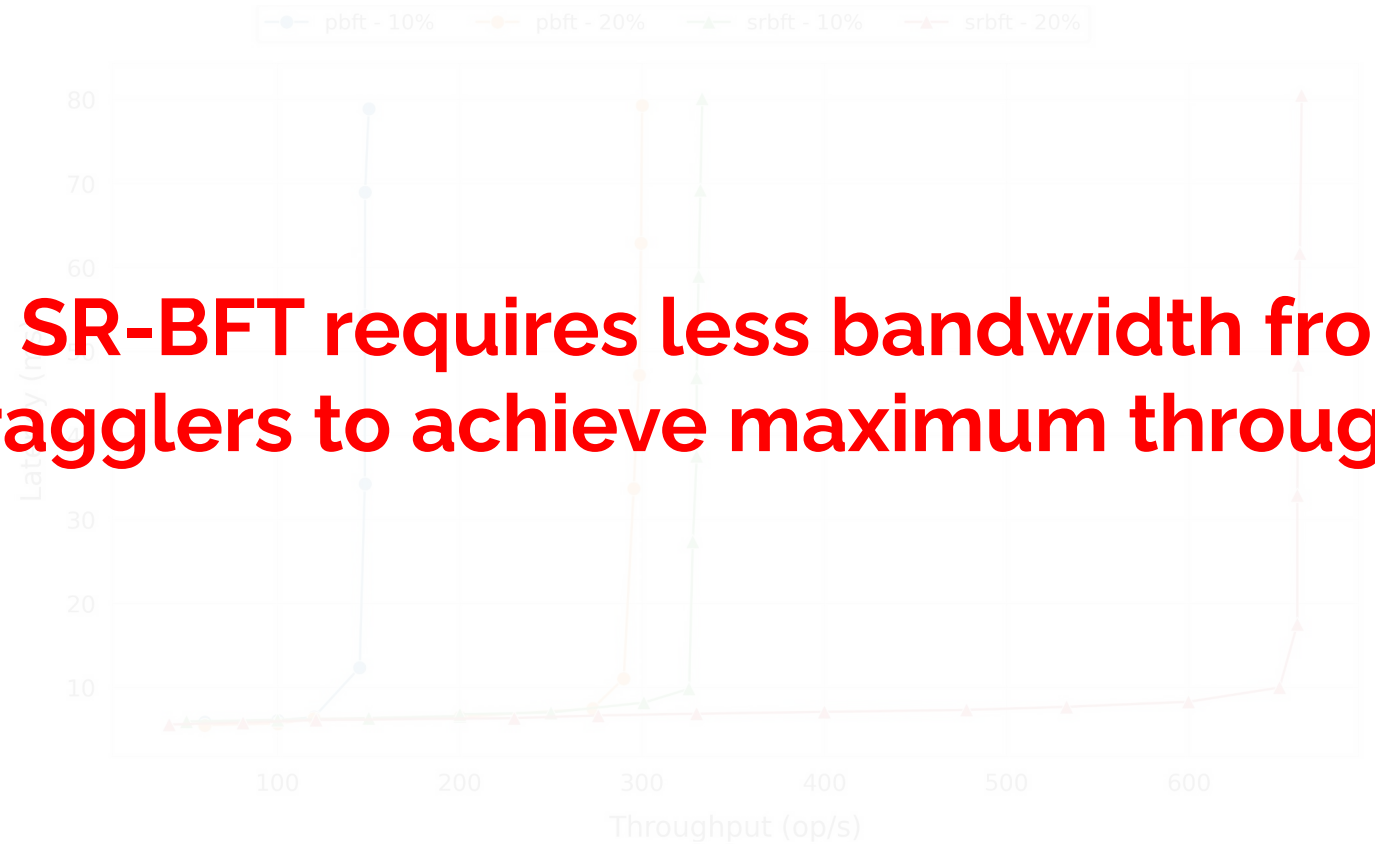


Throughput and Request Size



Throughput vs Latency





SR-BFT requires less bandwidth from stragglers to achieve maximum throughput

Lessons Learned

Lessons Learned

- Fight “heterogeneity” with “heterogeneity”

Lessons Learned

- Fight “heterogeneity” with “heterogeneity”
- Separate “data” and “control” planes

Future Works

Future Works

- Reputation-based leader rotation

Future Works

- Reputation-based leader rotation
- Dynamic straggler detection

Future Works

- Reputation-based leader rotation
- Dynamic straggler detection
- Bandwidth-friendly state transfer

Questions ?