Fault-Tolerant Distributed Transactions on Blockchain

Beyond the Design of PBFT



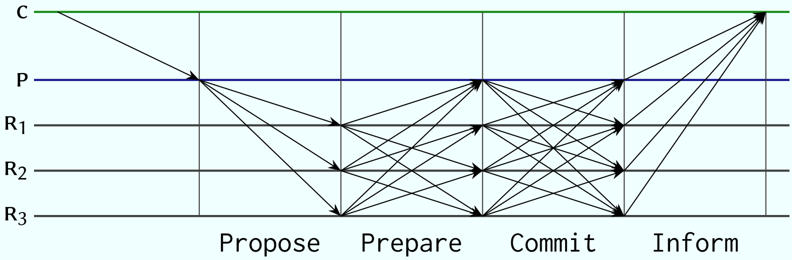
Suyash Gupta Jelle Hellings Mohammad Sadoghi

UCDAVIS
UNIVERSITY OF CALIFORNIA

ExpoLab
Creativity Unfolded

McMaster
University

Previously: PBFT



Central Question

What is the *expected performance* of PBFT? Motivate!

On the Performance of Consensus

Consensus throughput	Decisions per second made by consensus.
Consensus latency	Duration of a single round of consensus.
Resource utilization	The cost of consensus (e.g., computational, network bandwidth). *Imbalance* in resource utilized by replicas (e.g., primary).

On the Performance of Consensus

Consensus throughput	Decisions per second made by consensus.
Consensus latency	Duration of a single round of consensus.
Resource utilization	The cost of consensus (e.g., computational, network bandwidth). *Imbalance* in resource utilized by replicas (e.g., primary).
Complexity	Complexity of normal-case and of recovery (e.g., view-change).
Failure Model	The types of failures consensus can deal with.

On the Performance of Consensus

Consensus throughput	Decisions per second made by consensus.
Consensus latency	Duration of a single round of consensus.
Resource utilization	The cost of consensus (e.g., computational, network bandwidth). *Imbalance* in resource utilized by replicas (e.g., primary).
Complexity	Complexity of normal-case and of recovery (e.g., view-change).
Failure Model	The types of failures consensus can deal with.
Client latency	Duration between a client request and the outcome.

On the Performance of Consensus

Consensus throughput	Decisions per second made by consensus.
Consensus latency	Duration of a single round of consensus.
Resource utilization	The cost of consensus (e.g., computational, network bandwidth). *Imbalance* in resource utilized by replicas (e.g., primary).
Complexity	Complexity of normal-case and of recovery (e.g., view-change).
Failure Model	The types of failures consensus can deal with.
Client latency	Duration between a client request and the outcome. ▶ *Low loads*: Function of the consensus latency. ▶ *High loads*: Function of the consensus throughput.

Determining the Performance Variables

Number of replicas	determines the amount of messages exchanged.
Network bandwidth	determines how long it takes to exchange these messages.
Message delay	determines how long it takes for sent messages to arrive.
Computational speed	determines the speed by which messages are processed.

Determining the Performance Variables

Number of replicas determines the amount of messages exchanged.

Network bandwidth determines how long it takes to exchange these messages.

Message delay determines how long it takes for sent messages to arrive.

Computational speed determines the speed by which messages are processed.

System processing client transactions

Determining the Performance Variables

Number of replicas	determines the amount of messages exchanged.
Network bandwidth	determines how long it takes to exchange these messages.
Message delay	determines how long it takes for sent messages to arrive.
Computational speed	determines the speed by which messages are processed.

System processing client transactions

- Bottlenecks *outside consensus*: speed by which replicas *execute transactions*.

Determining the Performance Variables

Number of replicas	determines the amount of messages exchanged.
Network bandwidth	determines how long it takes to exchange these messages.
Message delay	determines how long it takes for sent messages to arrive.
Computational speed	determines the speed by which messages are processed.

System processing client transactions

- Bottlenecks *outside consensus*: speed by which replicas *execute transactions*.
- Computational speed typically sufficient when *parallelization* is used.

Determining the Performance Variables

Number of replicas	determines the amount of messages exchanged.
Network bandwidth	determines how long it takes to exchange these messages.
Message delay	determines how long it takes for sent messages to arrive.
Computational speed	determines the speed by which messages are processed.

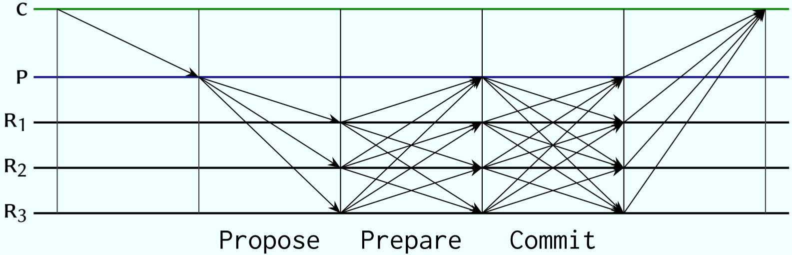
System processing client transactions

- Bottlenecks *outside consensus*: speed by which replicas *execute transactions*.
- Computational speed typically sufficient when *parallelization* is used.

Bottleneck in practice: consensus performance in terms of throughput and latency (as a function of *network bandwidth* and *message delay*).

The Single-Round Cost of PBFT (Sketch)

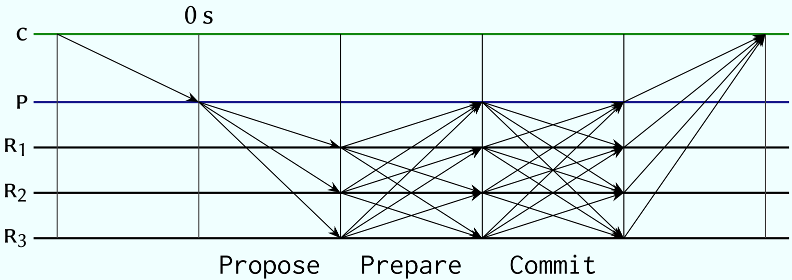
Assumption: Network bandwidth $B = 100\,\mathrm{MiB/s}$ and delay $\delta = 15\,\mathrm{ms}$



The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

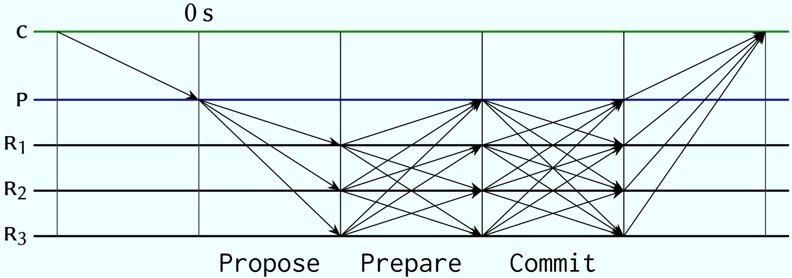
Propose: $s_t = 4048\,\text{B}$ each.



The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

Propose: $s_t = 4048\,\text{B}$ each.

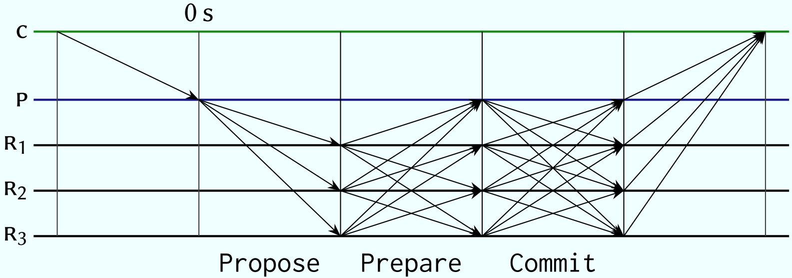


- **n** − 1 messages
- s_t B each

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

Propose: $s_t = 4048\,\text{B}$ each.



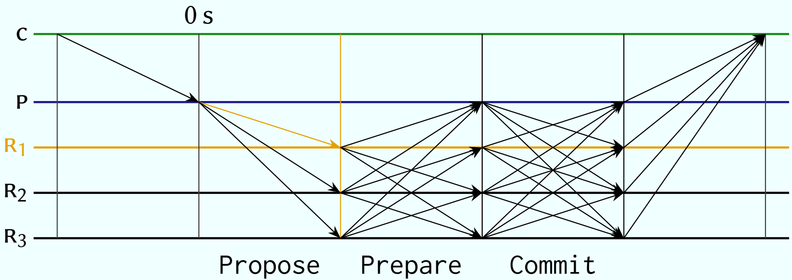
- $\mathbf{n} - 1$ messages
- s_t B each

$\left.\begin{array}{l} \\ \\ \end{array}\right\} (\mathbf{n} - 1)s_t = 3 \cdot 4048 = 12\,144\,\text{B}$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

Propose: $s_t = 4048\,\text{B}$ each.



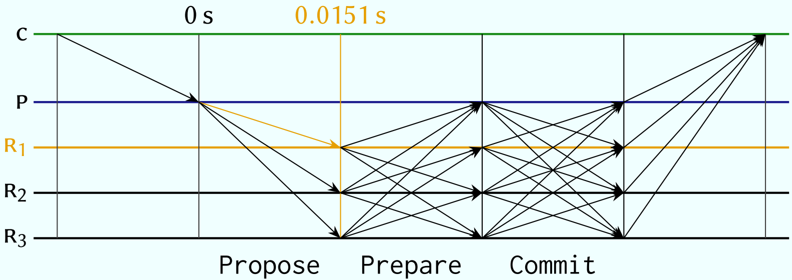
$$\left.\begin{array}{l} \blacktriangleright\ \mathbf{n} - 1 \text{ messages} \\ \blacktriangleright\ s_t \text{ B each} \end{array}\right\} (\mathbf{n} - 1)s_t = 3 \cdot 4048 = 12\,144\,\text{B}$$

- B MiB/s
- Last byte arrives after δ

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each.



- $\mathbf{n} - 1$ messages
- s_t B each
- B MiB/s
- Last byte arrives after δ

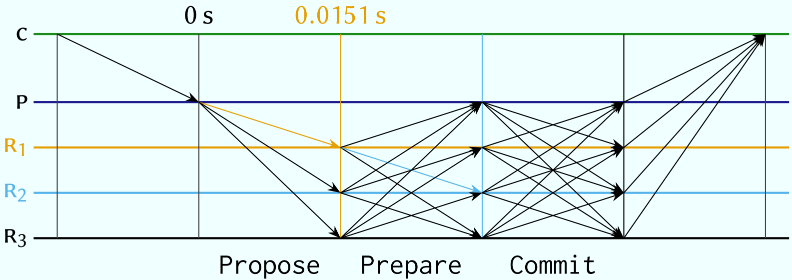
$$\left.\begin{array}{l}\mathbf{n}-1 \text{ messages}\\ s_t \text{ B each}\end{array}\right\} (\mathbf{n} - 1)s_t = 3 \cdot 4048 = 12\,144\,\text{B}$$

$$\left.\begin{array}{l}B \text{ MiB/s}\\ \text{Last byte arrives after } \delta\end{array}\right\} \frac{(\mathbf{n}-1)s_t}{B} + \delta = \frac{12144}{100 \cdot 2^{20}} + 0.015 \approx 0.0151\,\text{s}.$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

`Propose`: $s_t = 4048\,\text{B}$ each. `Prepare` and `Commit`: $s_m = 256\,\text{B}$ each.

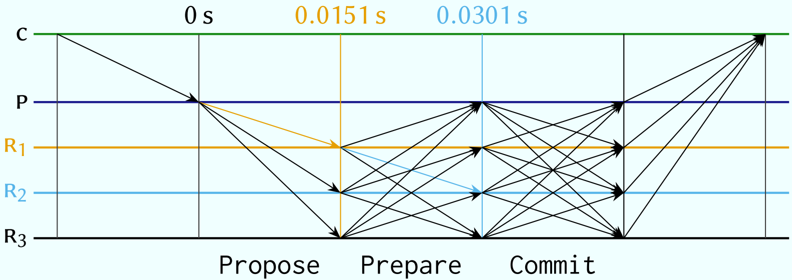


- **n** − 1 messages
- s_m B each
- B MiB/s
- Last byte arrives after δ

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\text{MiB/s}$ and delay $\delta = 15\,\text{ms}$

Propose: $s_t = 4048\,\text{B}$ each. Prepare and Commit: $s_m = 256\,\text{B}$ each.



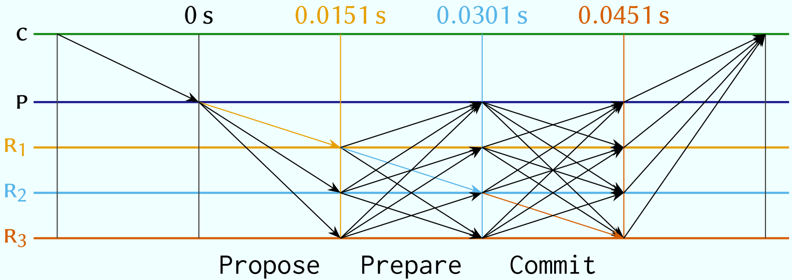
- $\mathbf{n} - 1$ messages
- s_m B each
- B MiB/s
- Last byte arrives after δ

$$\frac{(\mathbf{n}-1)s_m}{B} + \delta = \frac{768}{100 \cdot 2^{20}} + 0.015 \approx 0.0150\,\text{s}.$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\mathrm{MiB/s}$ and delay $\delta = 15\,\mathrm{ms}$

Propose: $s_t = 4048\,\mathrm{B}$ each. Prepare and Commit: $s_m = 256\,\mathrm{B}$ each.



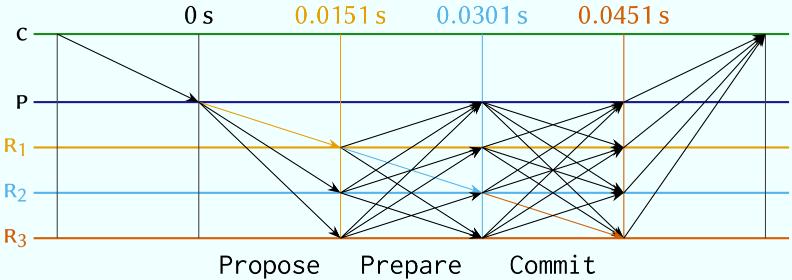
- **n** − 1 messages
- s_m B each
- B MiB/s
- Last byte arrives after δ

$$\left.\begin{array}{l} \mathbf{n}-1 \text{ messages} \\ s_m \text{ B each} \\ B \text{ MiB/s} \\ \text{Last byte arrives after } \delta \end{array}\right\} \approx 0.0150\,\mathrm{s}.$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.

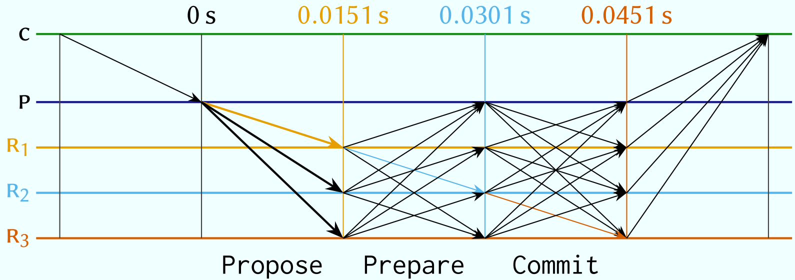


$\Delta_{\text{PBFT}} =$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\mathrm{MiB/s}$ and delay $\delta = 15\,\mathrm{ms}$

Propose: $s_t = 4048\,\mathrm{B}$ each. Prepare and Commit: $s_m = 256\,\mathrm{B}$ each.

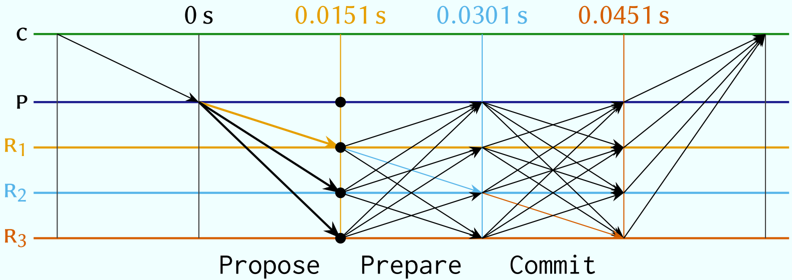


$$\Delta_{\mathrm{PBFT}} = \frac{(\mathbf{n} - 1)s_t}{B}$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.

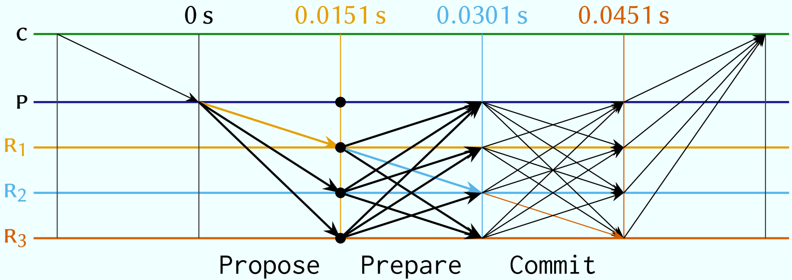


$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n} - 1)s_t}{B} + \delta$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\mathrm{MiB/s}$ and delay $\delta = 15\,\mathrm{ms}$

Propose: $s_t = 4048\,\mathrm{B}$ each. Prepare and Commit: $s_m = 256\,\mathrm{B}$ each.

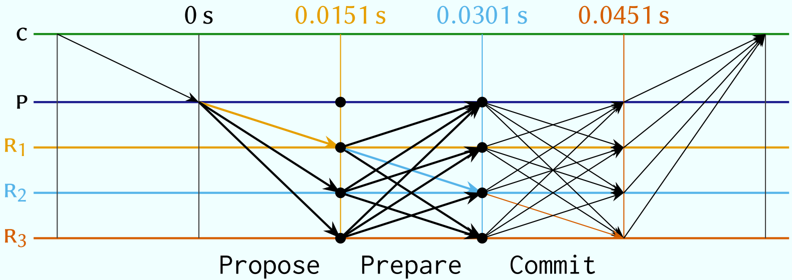


$$\Delta_{\mathrm{PBFT}} = \frac{(\mathbf{n} - 1)s_t}{B} + \delta + \frac{(\mathbf{n} - 1)s_m}{B}$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.

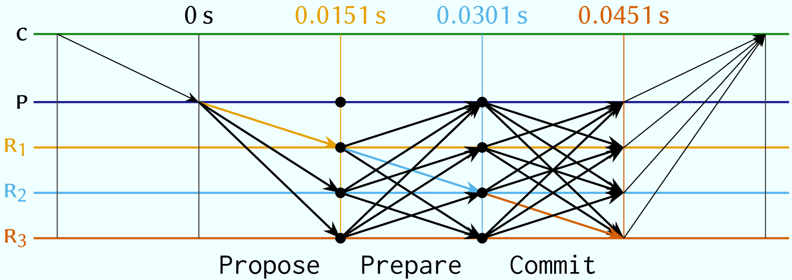


$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_t}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.

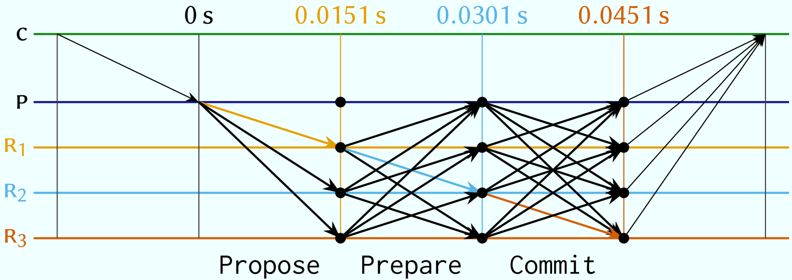


$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n} - 1)s_t}{B} + \delta + \frac{(\mathbf{n} - 1)s_m}{B} + \delta + \frac{(\mathbf{n} - 1)s_m}{B}$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.

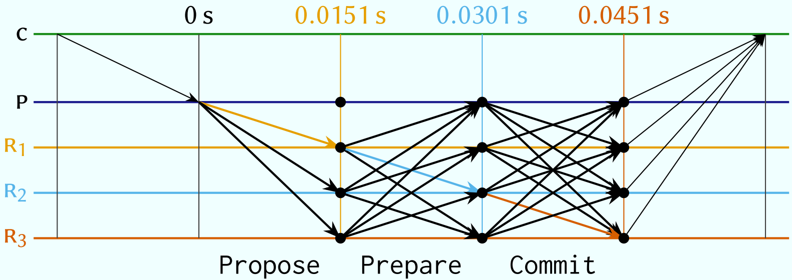


$$\Delta_{\mathrm{PBFT}} = \frac{(\mathbf{n}-1)s_t}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100\,\mathrm{MiB/s}$ and delay $\delta = 15\,\mathrm{ms}$

Propose: $s_t = 4048\,\mathrm{B}$ each. Prepare and Commit: $s_m = 256\,\mathrm{B}$ each.

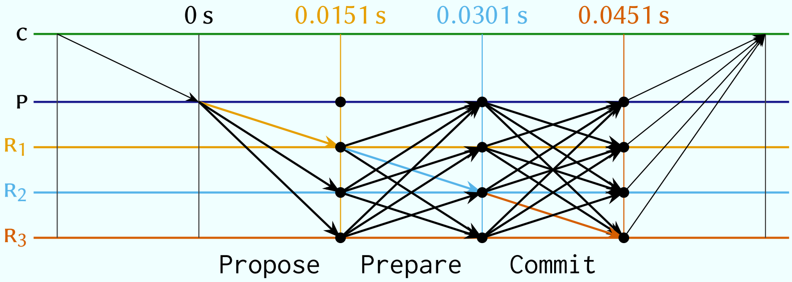


$$\Delta_{\mathrm{PBFT}} = \frac{(\mathbf{n}-1)s_t}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta$$
$$= \frac{(\mathbf{n}-1)s_t + 2(\mathbf{n}-1)s_m}{B} + 3\delta$$

The Single-Round Cost of PBFT (Sketch)

Assumption: Network bandwidth $B = 100$ MiB/s and delay $\delta = 15$ ms

Propose: $s_t = 4048$ B each. Prepare and Commit: $s_m = 256$ B each.



$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_t}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta + \frac{(\mathbf{n}-1)s_m}{B} + \delta$$

$$= \frac{(\mathbf{n}-1)s_t + 2(\mathbf{n}-1)s_m}{B} + 3\delta$$

$\approx 3\delta$ (assuming *high delay relative to bandwidth*).

The Throughput of PBFT

Sequential: Next consensus round starts after finishing the current round

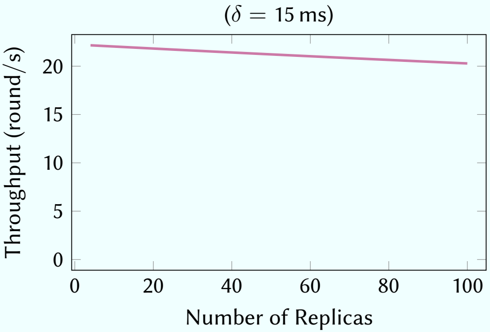
$$T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}} = \frac{B}{(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}} + 3B\delta}.$$

The Throughput of PBFT

Sequential: Next consensus round starts after finishing the current round

$$T_{\mathrm{PBFT}} = \frac{1}{\Delta_{\mathrm{PBFT}}} = \frac{B}{(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}} + 3B\delta}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$

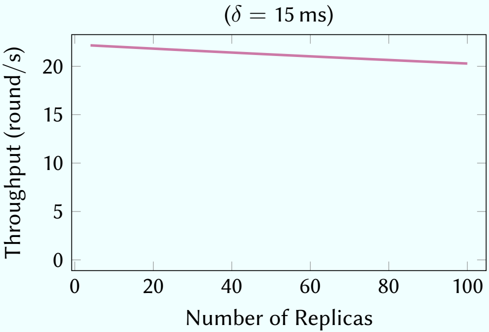


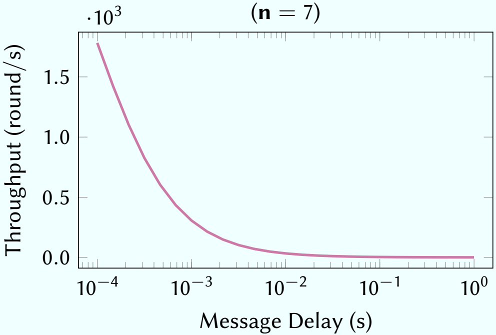
The Throughput of PBFT

Sequential: Next consensus round starts after finishing the current round

$$T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}} = \frac{B}{(\mathbf{n}-1)s_{\text{t}} + 2(\mathbf{n}-1)s_{\text{m}} + 3B\delta}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_{\text{t}} = 4048\,\text{B}$, $s_{\text{m}} = 256\,\text{B}$





Implementation techniques for PBFT

Realistic wide-area message delays: 10 ms–300 ms

The throughput T_{PBFT} of sequential PBFT is *impractically low*.

Implementation techniques for PBFT

Realistic wide-area message delays: 10 ms–300 ms

The throughput T_{PBFT} of sequential PBFT is *impractically low*.

Fine-tuning PBFT implementations

Batching	many transactions per consensus decision.
Out-of-order processing	many consensus decisions at the same time.
Overlapping	phases of consecutive rounds.

Batching Client Requests

The cost of a single round of PBFT

Message	Sent by	Size
`Propose`	Primary	s_{t}
`Prepare`	Backups	s_{m}
`Commit`	All	s_{m}

Batching Client Requests

The cost of a single round of PBFT

Batching: each decision is on **m** transactions.

Message	Sent by	Size	*(batch)*
`Propose`	Primary	s_t	ms_t
`Prepare`	Backups	s_m	s_m
`Commit`	All	s_m	s_m

Batching Client Requests

The cost of a single round of PBFT

Batching: each decision is on **m** transactions.

Message	Sent by	Size	*(batch)*
`Propose`	Primary	s_t	ms_t
`Prepare`	Backups	s_m	s_m
`Commit`	All	s_m	s_m
Total:	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n}^2)$	$\mathcal{O}(ms_t\mathbf{n} + s_m\mathbf{n}^2)$

The Throughput of PBFT with Batching

Sequential, batching *m* transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta;$$

The Throughput of PBFT with Batching

Sequential, batching *m* transactions per consensus round

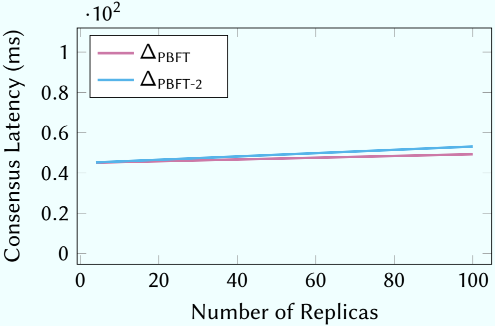
$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

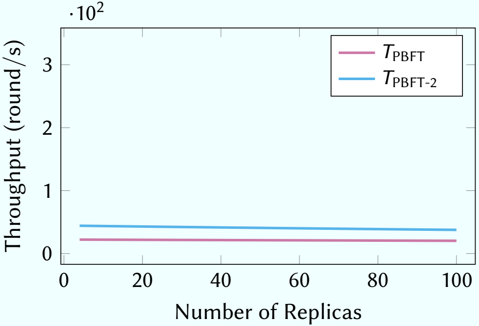
The Throughput of PBFT with Batching

Sequential, batching *m* transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_t + 2(\mathbf{n}-1)s_m}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

Assumption: $B = 100\ \text{MiB/s}$, $\delta = 15\ \text{ms}$, $s_t = 4048\ \text{B}$, $s_m = 256\ \text{B}$



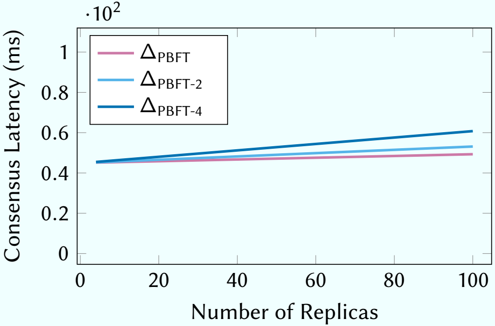


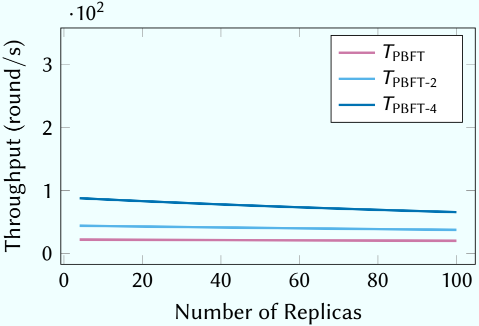
The Throughput of PBFT with Batching

Sequential, batching m transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\text{t}} + 2(\mathbf{n}-1)s_{\text{m}}}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

Assumption: $B = 100\ \text{MiB/s}$, $\delta = 15\ \text{ms}$, $s_{\text{t}} = 4048\ \text{B}$, $s_{\text{m}} = 256\ \text{B}$



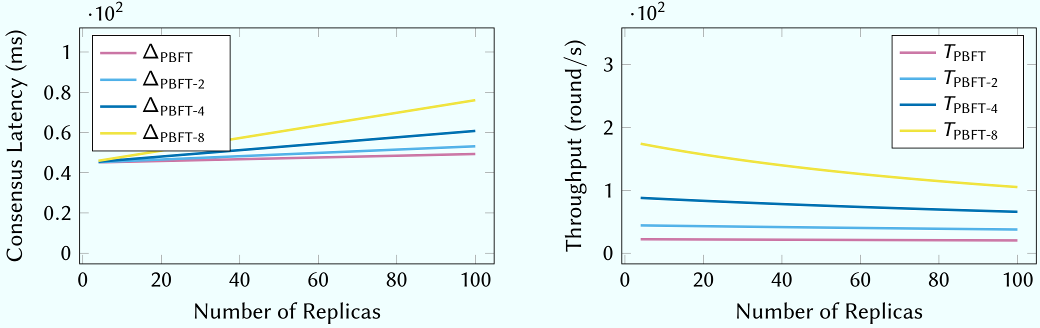


The Throughput of PBFT with Batching

Sequential, batching m transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

Assumption: $B = 100\,\text{MiB/s}$, $\delta = 15\,\text{ms}$, $s_{\mathrm{t}} = 4048\,\text{B}$, $s_{\mathrm{m}} = 256\,\text{B}$

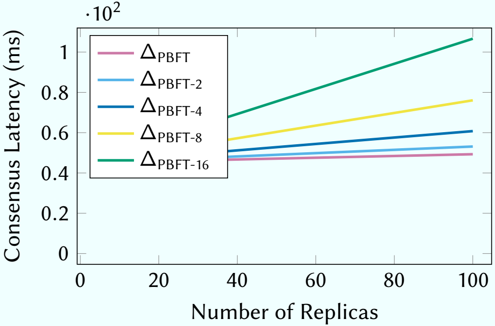


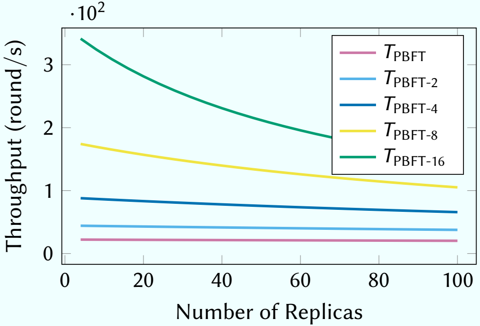
The Throughput of PBFT with Batching

Sequential, batching m transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $\delta = 15\,\mathrm{ms}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$



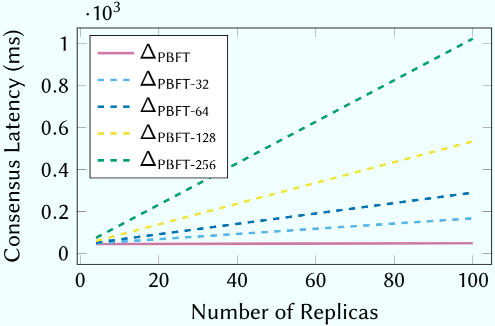


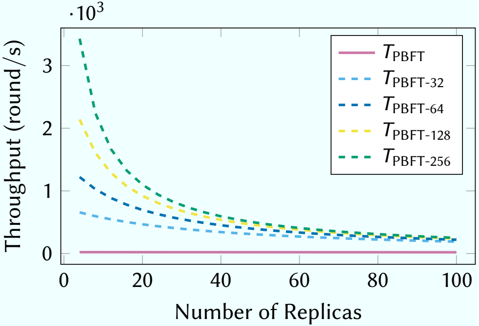
The Throughput of PBFT with Batching

Sequential, batching *m* transactions per consensus round

$$\Delta_{\text{PBFT-}m} = \frac{m(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta; \qquad T_{\text{PBFT-}m} = m\frac{1}{\Delta_{\text{PBFT-}m}}.$$

Assumption: $B = 100\ \text{MiB/s}$, $\delta = 15\ \text{ms}$, $s_{\mathrm{t}} = 4048\ \text{B}$, $s_{\mathrm{m}} = 256\ \text{B}$





Using Batching to Improve Throughput Scalability

	Messages	*(per trans.)*	Size	*(per trans.)*
PBFT	$2\mathbf{n}(\mathbf{n}-1)$	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$

Using Batching to Improve Throughput Scalability

	Messages	*(per trans.)*	Size	*(per trans.)*	
PBFT	$2\mathbf{n}(\mathbf{n}-1)$	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	*Quadratic*

Using Batching to Improve Throughput Scalability

	Messages	*(per trans.)*	Size	*(per trans.)*	
PBFT	$2\mathbf{n}(\mathbf{n}-1)$	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	*Quadratic*
PBFT-**n**	$2\mathbf{n}(\mathbf{n}-1)$	$2(\mathbf{n}-1)$	$\mathcal{O}(\mathbf{n}s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n}^2)$	$\mathcal{O}(s_{\mathrm{t}}\mathbf{n} + s_{\mathrm{m}}\mathbf{n})$	

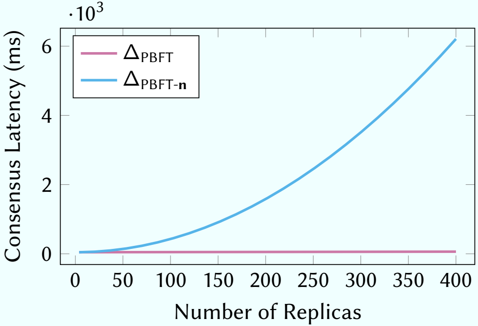
Using Batching to Improve Throughput Scalability

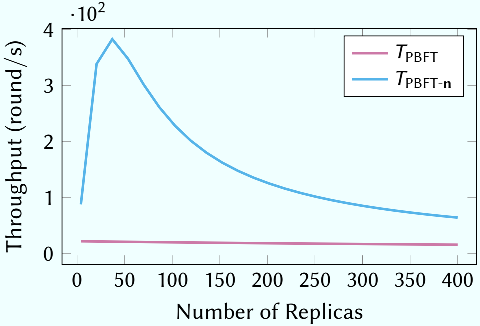
	Messages	*(per trans.)*	Size	*(per trans.)*	
PBFT	$2\mathbf{n}(\mathbf{n}-1)$	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n}^2)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n}^2)$	*Quadratic*
PBFT-$\mathbf{n}$	$2\mathbf{n}(\mathbf{n}-1)$	$2(\mathbf{n}-1)$	$\mathcal{O}(\mathbf{n}s_t\mathbf{n} + s_m\mathbf{n}^2)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n})$	*Linear*

Using Batching to Improve Throughput Scalability

	Messages	*(per trans.)*	Size	*(per trans.)*	
PBFT	$2\mathbf{n}(\mathbf{n}-1)$	$2\mathbf{n}(\mathbf{n}-1)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n}^2)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n}^2)$	*Quadratic*
PBFT-**n**	$2\mathbf{n}(\mathbf{n}-1)$	$2(\mathbf{n}-1)$	$\mathcal{O}(\mathbf{n}s_t\mathbf{n} + s_m\mathbf{n}^2)$	$\mathcal{O}(s_t\mathbf{n} + s_m\mathbf{n})$	*Linear*

Assumption: $B = 100\ \text{MiB/s}$, $\delta = 15\ \text{ms}$, $s_t = 4048\ \text{B}$, $s_m = 256\ \text{B}$





Resource Utilization of Sequential PBFT

Assumption: $\mathbf{n} = 4$, $B = 100\,\text{MiB/s}$, $\delta = 15\,\text{ms}$, $s_{\mathrm{t}} = 4048\,\text{B}$, $s_{\mathrm{m}} = 256\,\text{B}$

$$\Delta_{\mathrm{PBFT}} = \frac{(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B} + 3\delta \approx 45.1\,\text{ms}.$$

Resource Utilization of Sequential PBFT

Assumption: $\mathbf{n} = 4$, $B = 100\,\text{MiB/s}$, $\delta = 15\,\text{ms}$, $s_\mathrm{t} = 4048\,\text{B}$, $s_\mathrm{m} = 256\,\text{B}$

$$\Delta_{\text{PBFT}} = \underbrace{\frac{(\mathbf{n}-1)s_\mathrm{t} + 2(\mathbf{n}-1)s_\mathrm{m}}{B}}_{\substack{\textit{message transfer} \\ \approx 0.1\,\text{ms}}} + 3\delta \approx 45.1\,\text{ms}.$$

Resource Utilization of Sequential PBFT

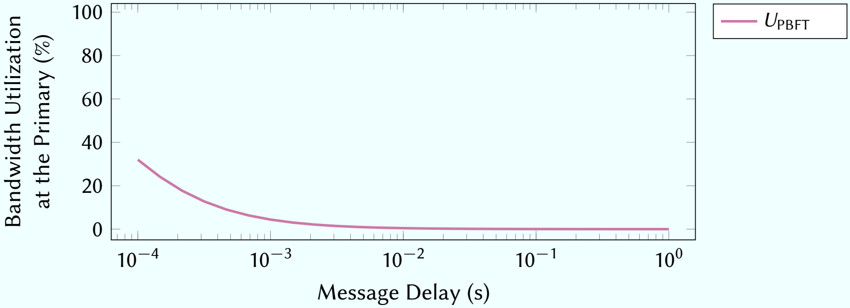
Assumption: $\mathbf{n} = 4$, $B = 100\,\mathrm{MiB/s}$, $\delta = 15\,\mathrm{ms}$, $s_\mathrm{t} = 4048\,\mathrm{B}$, $s_\mathrm{m} = 256\,\mathrm{B}$

$$\Delta_{\mathrm{PBFT}} = \underbrace{\frac{(\mathbf{n}-1)s_\mathrm{t} + 2(\mathbf{n}-1)s_\mathrm{m}}{B}}_{\substack{\textit{message transfer} \\ \approx 0.1\,\mathrm{ms}}} + \underbrace{3\delta}_{\substack{\textit{waiting} \\ 45.0\,\mathrm{ms}}} \approx 45.1\,\mathrm{ms}.$$

Resource Utilization of Sequential PBFT

Assumption: $\mathbf{n} = 4$, $B = 100\,\mathrm{MiB/s}$, $\delta = 15\,\mathrm{ms}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$

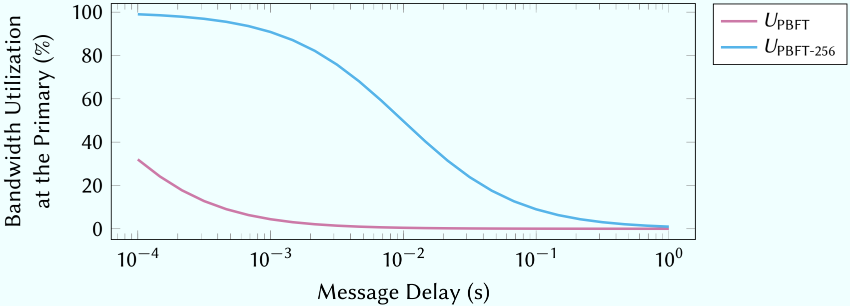
$$\Delta_{\mathrm{PBFT}} = \underbrace{\frac{(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B}}_{\substack{\textit{message transfer}\\ \approx 0.1\,\mathrm{ms}}} + \underbrace{3\delta}_{\substack{\textit{waiting}\\ 45.0\,\mathrm{ms}}} \approx 45.1\,\mathrm{ms}.$$



Resource Utilization of Sequential PBFT

Assumption: $\mathbf{n} = 4$, $B = 100\,\mathrm{MiB/s}$, $\delta = 15\,\mathrm{ms}$, $s_\mathrm{t} = 4048\,\mathrm{B}$, $s_\mathrm{m} = 256\,\mathrm{B}$

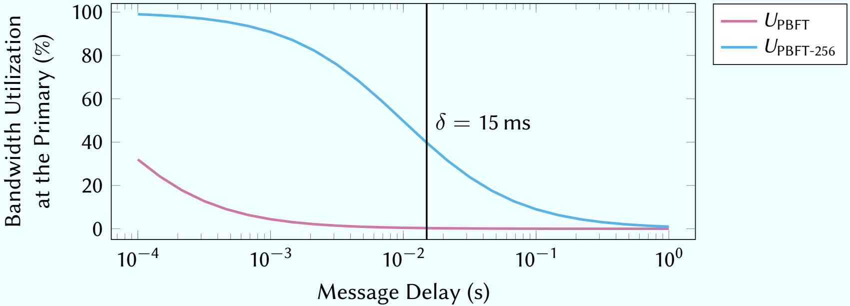
$$\Delta_{\mathrm{PBFT}} = \underbrace{\frac{(\mathbf{n}-1)s_\mathrm{t} + 2(\mathbf{n}-1)s_\mathrm{m}}{B}}_{\substack{\textit{message transfer}\\ \approx 0.1\,\mathrm{ms}}} + \underbrace{3\delta}_{\substack{\textit{waiting}\\ 45.0\,\mathrm{ms}}} \approx 45.1\,\mathrm{ms}.$$



Resource Utilization of Sequential PBFT

Assumption: $\mathbf{n} = 4$, $B = 100\,\mathrm{MiB/s}$, $\delta = 15\,\mathrm{ms}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$

$$\Delta_{\mathrm{PBFT}} = \underbrace{\frac{(\mathbf{n}-1)s_{\mathrm{t}} + 2(\mathbf{n}-1)s_{\mathrm{m}}}{B}}_{\substack{\textit{message transfer} \\ \approx 0.1\,\mathrm{ms}}} + \underbrace{3\delta}_{\substack{\textit{waiting} \\ 45.0\,\mathrm{ms}}} \approx 45.1\,\mathrm{ms}.$$



Problem: Resource Utilization of PBFT

- Typically *less than 5%* bandwidth utilization at primary.
- With huge batches still *less than 40%* bandwidth utilization at primary.

Problem: Resource Utilization of PBFT

- Typically *less than 5%* bandwidth utilization at primary.
- With huge batches still *less than 40%* bandwidth utilization at primary.

To maximize throughput: use *all* bandwidth at the primary.

Problem: Resource Utilization of PBFT

- Typically *less than 5%* bandwidth utilization at primary.
- With huge batches still *less than 40%* bandwidth utilization at primary.

To maximize throughput: use *all* bandwidth at the primary.

Out-of-order processing

Primary can proposes *future rounds* before current rounds are finished.

Problem: Resource Utilization of PBFT

- Typically *less than 5%* bandwidth utilization at primary.
- With huge batches still *less than 40%* bandwidth utilization at primary.

To maximize throughput: use *all* bandwidth at the primary.

Out-of-order processing

Primary can proposes *future rounds* before current rounds are finished.

Practical challenges

- Memory usage: replicas maintain meta-data for each *active* round.
- Byzantine behavior: exhaust the set of round numbers.

Problem: Resource Utilization of PBFT

- Typically *less than 5%* bandwidth utilization at primary.
- With huge batches still *less than 40%* bandwidth utilization at primary.

To maximize throughput: use *all* bandwidth at the primary.

Out-of-order processing

Primary can proposes *future rounds* before current rounds are finished.

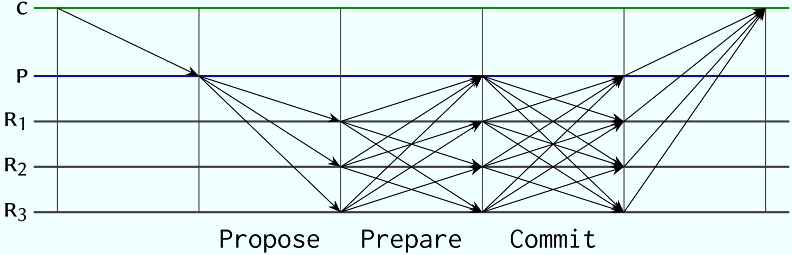
Practical challenges

- Memory usage: replicas maintain meta-data for each *active* round.
- Byzantine behavior: exhaust the set of round numbers.

Limit proposals to an *active window* of valid rounds.
E.g., only proposals in 1000 rounds after the last finished round.

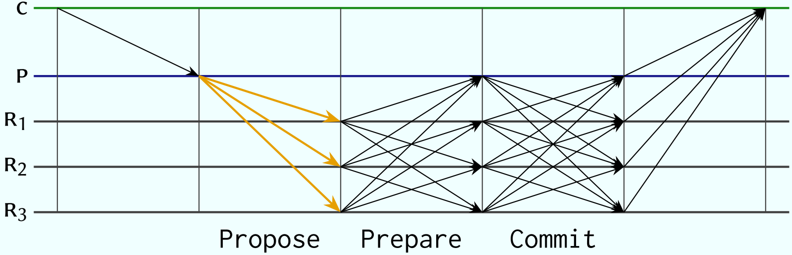
The Single-Round Cost of PBFT (revised)

Assumption: Primary does most work ($s_t > s_m$)



The Single-Round Cost of PBFT (revised)

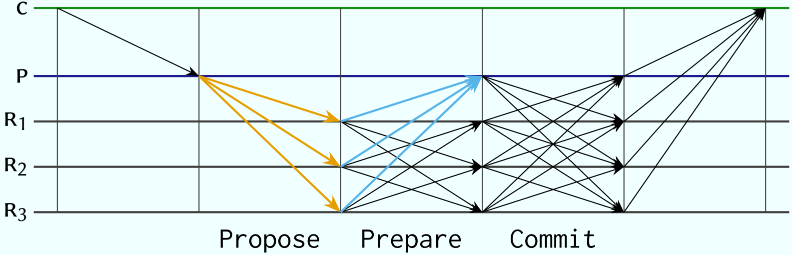
Assumption: Primary does most work ($s_t > s_m$)



- Send $\mathbf{n} - 1$ messages
- s_t B each

The Single-Round Cost of PBFT (revised)

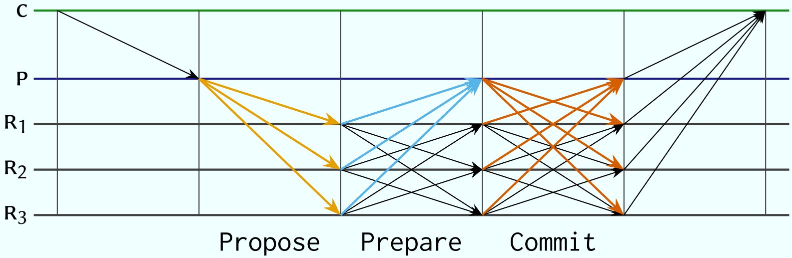
Assumption: Primary does most work ($s_t > s_m$)



- Receive $\mathbf{n} - 1$ messages
- s_m B each

The Single-Round Cost of PBFT (revised)

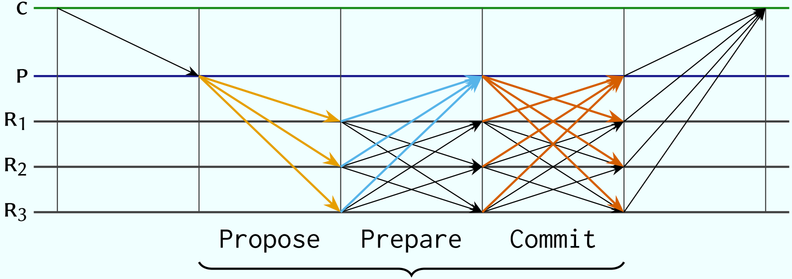
Assumption: Primary does most work ($s_t > s_m$)



- Send and receive $\mathbf{n} - 1$ messages
- s_m B each

The Single-Round Cost of PBFT (revised)

Assumption: Primary does most work ($s_t > s_m$)



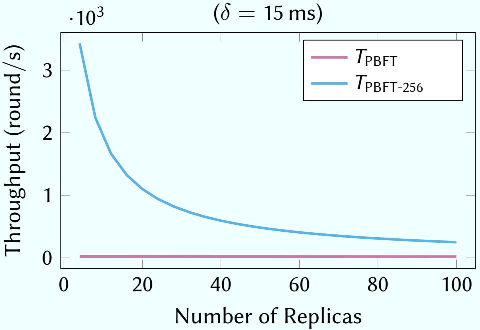
$$\underbrace{(\mathbf{n}-1)s_t + (\mathbf{n}-1)s_m + 2(\mathbf{n}-1)s_m}_{} = (\mathbf{n}-1)(s_t + 3s_m)\ \mathrm{B/round.}$$

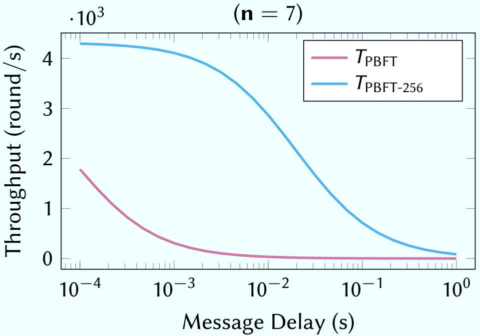
The Out-of-Order Throughput of PBFT

Assumption: Primary does most work ($s_t > s_m$)

$$T_{\text{ooo-PBFT}} = \frac{B}{(\mathbf{n} - 1)(s_t + 3s_m)}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_t = 4048\,\text{B}$, $s_m = 256\,\text{B}$



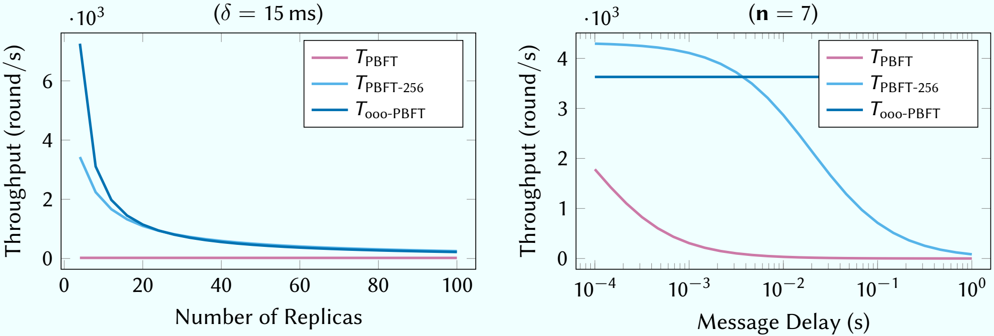


The Out-of-Order Throughput of PBFT

Assumption: Primary does most work ($s_t > s_m$)

$$T_{\text{ooo-PBFT}} = \frac{B}{(\mathbf{n} - 1)(s_t + 3s_m)}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_t = 4048\,\text{B}$, $s_m = 256\,\text{B}$

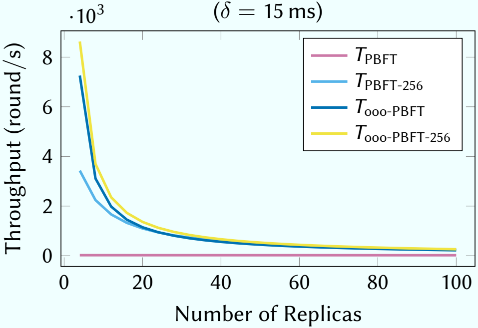


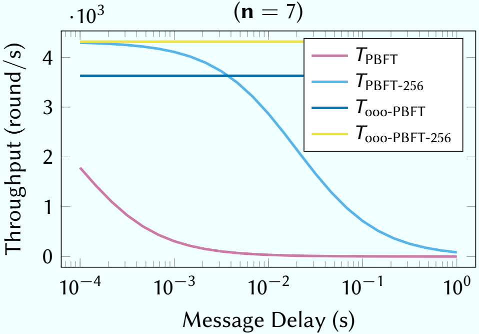
The Out-of-Order Throughput of PBFT

Assumption: Primary does most work ($s_t > s_m$)

$$T_{\text{ooo-PBFT}} = \frac{B}{(\mathbf{n}-1)(s_t + 3s_m)}; \qquad T_{\text{ooo-PBFT-}m} = \frac{mB}{(\mathbf{n}-1)(ms_t + 3s_m)}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_t = 4048\,\text{B}$, $s_m = 256\,\text{B}$





Overlapping Communication Phases

Out-of-order processing is *complex* to implement.

Overlapping Communication Phases

Out-of-order processing is *complex* to implement.

Consider a backup replica R.

- Last step of round ρ: `Commit` messages.
- First step of round $\rho + 1$: `Prepare` messages.

Overlapping Communication Phases

Out-of-order processing is *complex* to implement.

Consider a backup replica R.

- Last step of round ρ: `Commit` messages.
- First step of round $\rho + 1$: `Prepare` messages.

Idea: Overlapping communication phases

Merge `Commit` message of ρ with the `Prepare` message of $\rho + 1$.

Overlapping Communication Phases

Out-of-order processing is *complex* to implement.

Consider a backup replica R.

- Last step of round ρ: `Commit` messages.
- First step of round $\rho + 1$: `Prepare` messages.

Idea: Overlapping communication phases

Merge `Commit` message of ρ with the `Prepare` message of $\rho + 1$.

- Make proposal of round $\rho + 1$ *refer* to round ρ.
- `Prepare` for round $\rho + 1$ *implies* `Commit` for round ρ.
- Primary proposes round $\rho + 1$ *after* it finished the prepare-phase for round ρ.

Overlapping Communication Phases

Out-of-order processing is *complex* to implement.

Consider a backup replica R.

- Last step of round ρ: `Commit` messages.
- First step of round $\rho + 1$: `Prepare` messages.

Idea: Overlapping communication phases

Merge `Commit` message of ρ with the `Prepare` message of $\rho + 1$.

- Make proposal of round $\rho + 1$ *refer* to round ρ.
- `Prepare` for round $\rho + 1$ *implies* `Commit` for round ρ.
- Primary proposes round $\rho + 1$ *after* it finished the prepare-phase for round ρ.

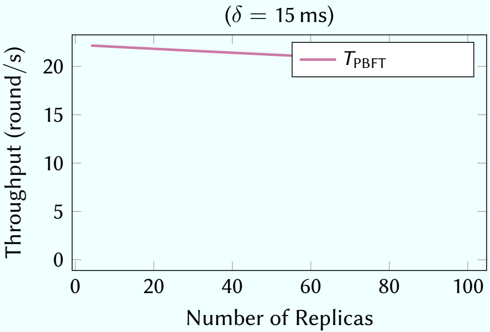
Implies strict consecutive processing of rounds

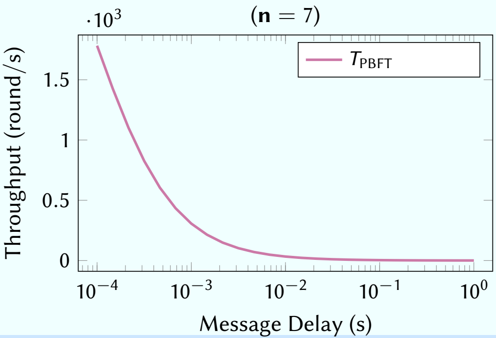
Overlapping *cannot* be combined with out-of-order processing!

The Single-Round Cost of PBFT with Overlapping

$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_{\text{t}} + 2(\mathbf{n}-1)s_{\text{m}}}{B} + 3\delta; \qquad T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}}$$

Assumption: $B = 100\,\text{MiB/s}$, $s_{\text{t}} = 4048\,\text{B}$, $s_{\text{m}} = 256\,\text{B}$



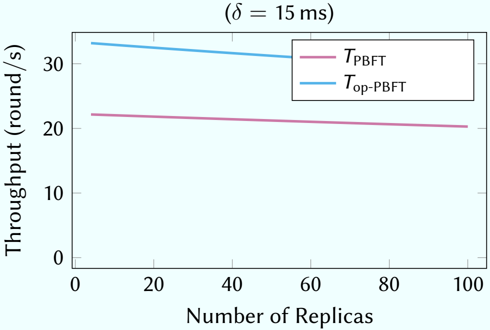


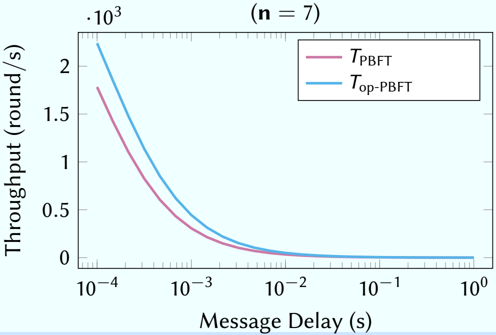
The Single-Round Cost of PBFT with Overlapping

$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_{\text{t}} + 2(\mathbf{n}-1)s_{\text{m}}}{B} + 3\delta; \qquad T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}}$$

$$\Delta_{\text{op-PBFT}} = \frac{(\mathbf{n}-1)s_{\text{t}} + (\mathbf{n}-1)s_{\text{m}}}{B} + 2\delta; \qquad T_{\text{op-PBFT}} = \frac{1}{\Delta_{\text{op-PBFT}}}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_{\text{t}} = 4048\,\text{B}$, $s_{\text{m}} = 256\,\text{B}$



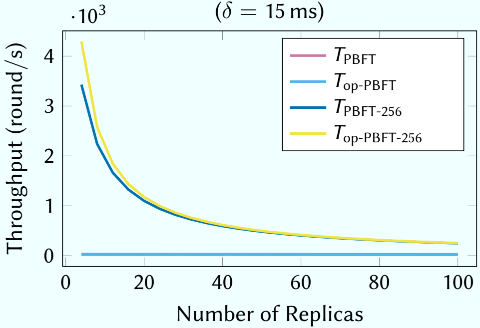


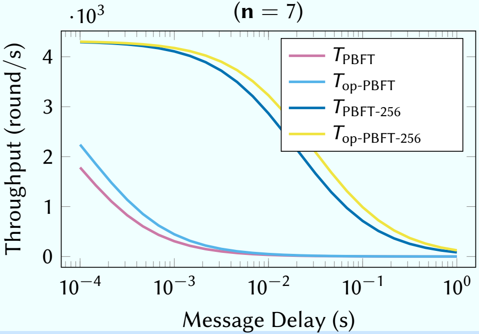
The Single-Round Cost of PBFT with Overlapping

$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_{\text{t}} + 2(\mathbf{n}-1)s_{\text{m}}}{B} + 3\delta; \qquad T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}}$$

$$\Delta_{\text{op-PBFT}} = \frac{(\mathbf{n}-1)s_{\text{t}} + (\mathbf{n}-1)s_{\text{m}}}{B} + 2\delta; \qquad T_{\text{op-PBFT}} = \frac{1}{\Delta_{\text{op-PBFT}}}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_{\text{t}} = 4048\,\text{B}$, $s_{\text{m}} = 256\,\text{B}$



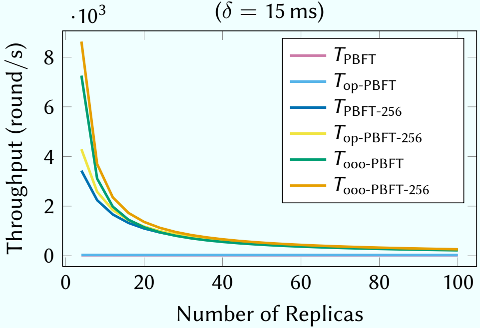


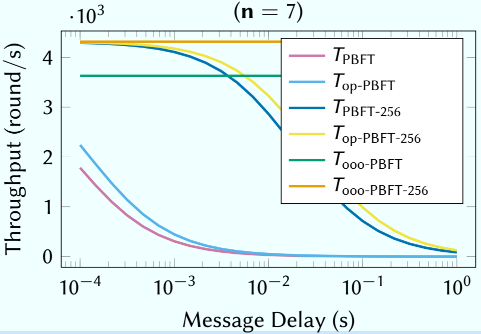
The Single-Round Cost of PBFT with Overlapping

$$\Delta_{\text{PBFT}} = \frac{(\mathbf{n}-1)s_t + 2(\mathbf{n}-1)s_m}{B} + 3\delta; \qquad T_{\text{PBFT}} = \frac{1}{\Delta_{\text{PBFT}}}$$

$$\Delta_{\text{op-PBFT}} = \frac{(\mathbf{n}-1)s_t + (\mathbf{n}-1)s_m}{B} + 2\delta; \qquad T_{\text{op-PBFT}} = \frac{1}{\Delta_{\text{op-PBFT}}}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_t = 4048\,\text{B}$, $s_m = 256\,\text{B}$

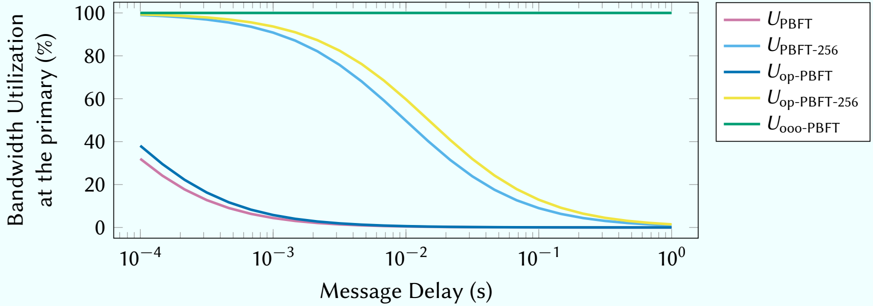




Implementation techniques for PBFT: Summary

Batching	introduces very high round latencies.
Out-of-order processing	has high implementation complexity.
Overlapping	only provides limited gains.

Assumption: $\mathbf{n} = 4$, $B = 100\,\mathrm{MiB}/\mathrm{s}$, $\delta = 15\,\mathrm{ms}$, $s_\mathrm{t} = 4048\,\mathrm{B}$, $s_\mathrm{m} = 256\,\mathrm{B}$



Primary-backup Consensus Beyond PBFT

A PBFT-like design is at the basis of *many* consensus protocols.

Primary-backup Consensus Beyond PBFT

A PBFT-like design is at the basis of *many* consensus protocols.

Technologies employed by PBFT-like consensus

Threshold signatures eliminate quadratic all-to-all communication.

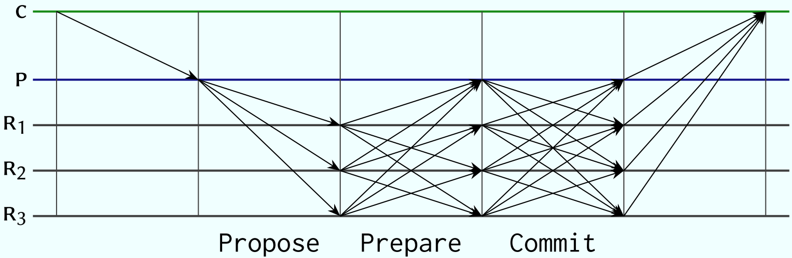
Speculative execution execute before strong recovery guarantees are met.

Optimistic execution fully optimize for when the primary is correct.

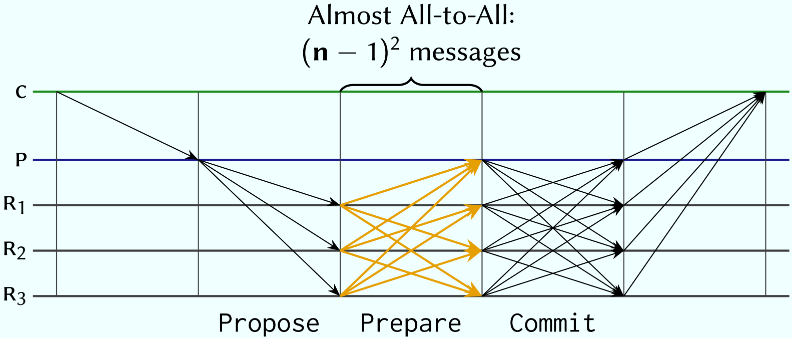
Trusted components use hardware components that cannot behave Byzantine.

Here, we will only cover *threshold signatures*.

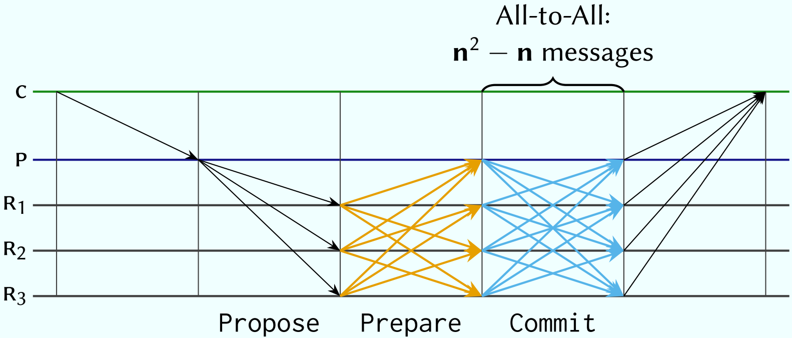
All-to-All Communication in PBFT



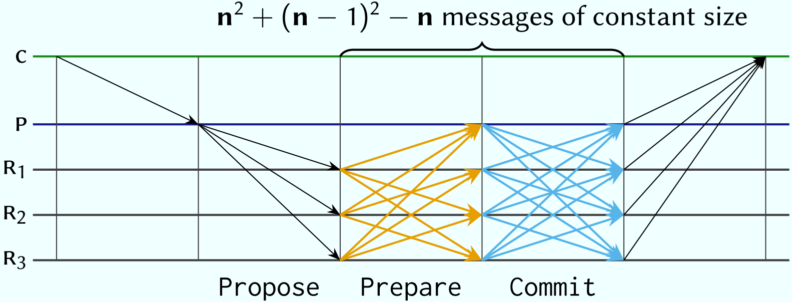
All-to-All Communication in PBFT



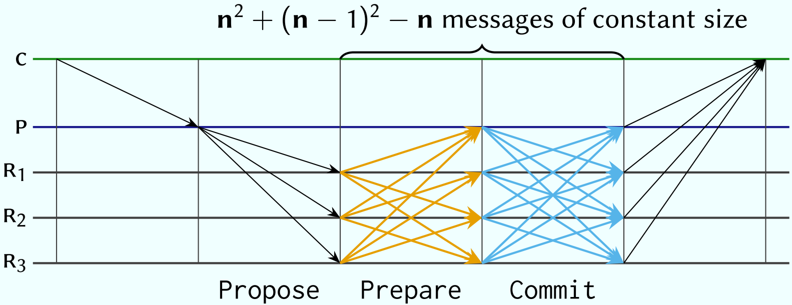
All-to-All Communication in PBFT



All-to-All Communication in PBFT



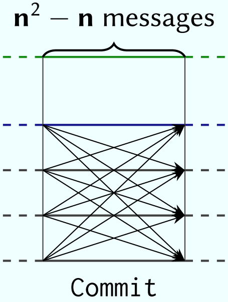
All-to-All Communication in PBFT



Challenge: Reduce communication from $\mathcal{O}(\mathbf{n}^2)$ to $\mathcal{O}(\mathbf{n})$ *messages of constant size*.

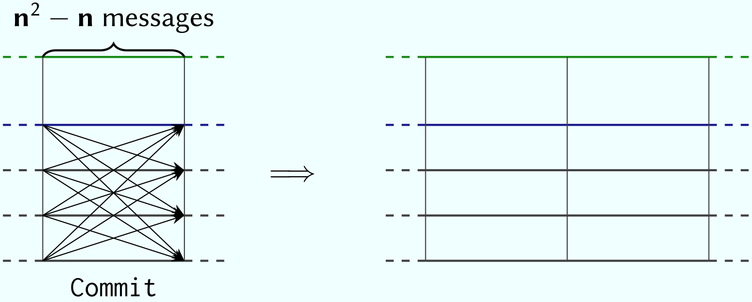
Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase



Tackling All-to-All via All-to-one-to-All Aggregation

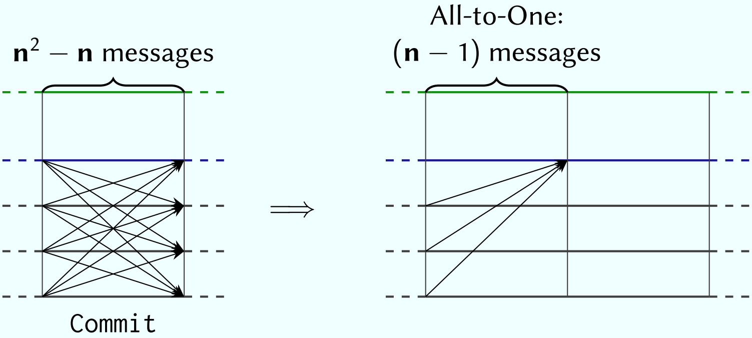
Consider the commit phase



Idea: All replicas send to *one aggregator* that then sends to all replicas.

Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase

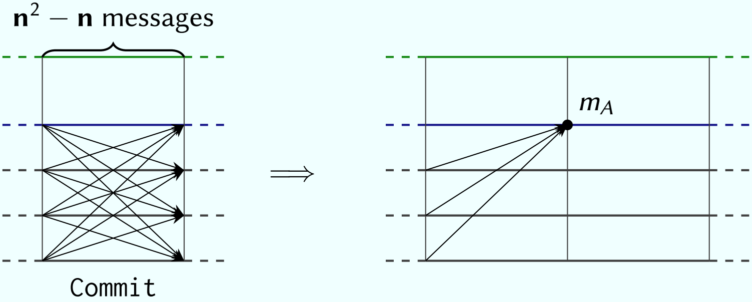


Idea: All replicas send to *one aggregator* that then sends to all replicas.

1. All replicas send their `Commit` messages to the aggregator.

Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase

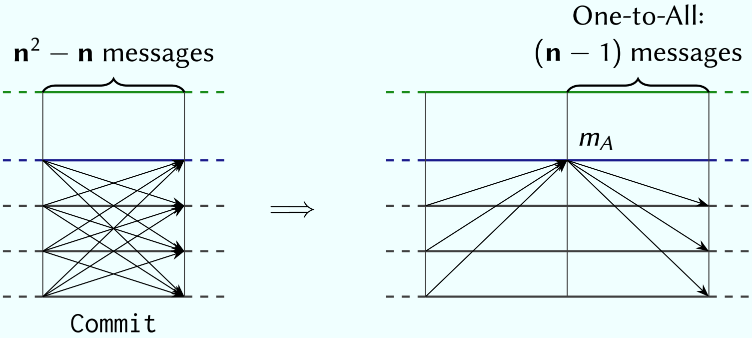


Idea: All replicas send to *one aggregator* that then sends to all replicas.

2. The aggregator combines $\mathbf{n} - \mathbf{f}$ `Commit` messages into an aggregated message m_A.

Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase

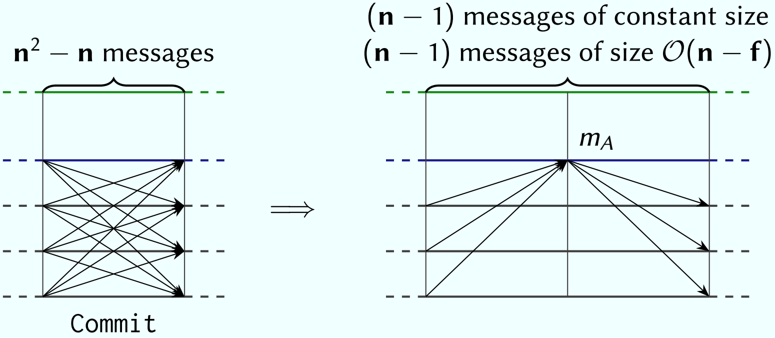


Idea: All replicas send to *one aggregator* that then sends to all replicas.

3. The aggregator sends m_A to all replicas.

Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase

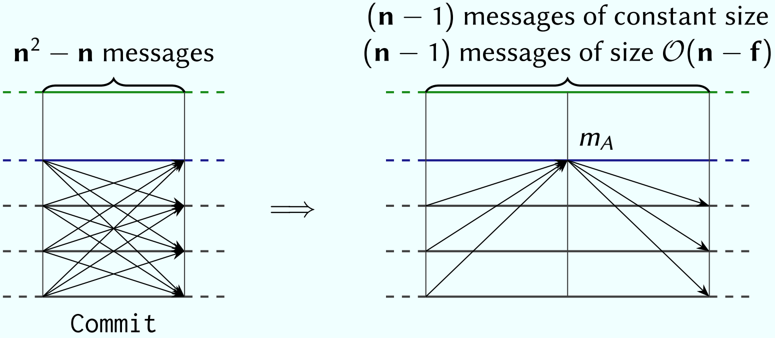


Idea: All replicas send to *one aggregator* that then sends to all replicas.

3. The aggregator sends m_A to all replicas *of size* $\mathcal{O}(\mathbf{n}-\mathbf{f})$ *each*.

Tackling All-to-All via All-to-one-to-All Aggregation

Consider the commit phase



Idea: All replicas send to *one aggregator* that then sends to all replicas.

Effectively reduced communication from $\mathcal{O}(\mathbf{n}^2)$ to $\mathcal{O}(\mathbf{n}(\mathbf{n} - \mathbf{f}))$.

Improving Aggregation with Threshold Signatures

Problem: An aggregated message of size c will have size $\mathcal{O}(c(\mathbf{n}-\mathbf{f}))$.

- We have identical `Commit` messages from at-least $\mathbf{n}-\mathbf{f}$ replicas.
- Goal: aggregate these into a single message of size $\mathcal{O}(c)$ instead of $\mathcal{O}(c(\mathbf{n}-\mathbf{f}))$.

Improving Aggregation with Threshold Signatures

Problem: An aggregated message of size c will have size $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.

- We have identical `Commit` messages from at-least $\mathbf{n} - \mathbf{f}$ replicas.
- Goal: aggregate these into a single message of size $\mathcal{O}(c)$ instead of $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.
- Crucially: we want to aggregate the digital signatures!

Improving Aggregation with Threshold Signatures

Problem: An aggregated message of size c will have size $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.

- We have identical `Commit` messages from at-least $\mathbf{n} - \mathbf{f}$ replicas.
- Goal: aggregate these into a single message of size $\mathcal{O}(c)$ instead of $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.
- Crucially: we want to aggregate the digital signatures!

Solution: Using a $\mathbf{n} : \mathbf{f}$-threshold-signature scheme with public key K

- Each replica has a unique private key.
- Replicas can produce partial signatures for value v using their private key.
- Using $\mathbf{n} - \mathbf{f}$ partial signatures for v, one can produce a *threshold signature*.

Improving Aggregation with Threshold Signatures

Problem: An aggregated message of size c will have size $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.

- We have identical `Commit` messages from at-least $\mathbf{n} - \mathbf{f}$ replicas.
- Goal: aggregate these into a single message of size $\mathcal{O}(c)$ instead of $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.
- Crucially: we want to aggregate the digital signatures!

Solution: Using a $\mathbf{n} : \mathbf{f}$-threshold-signature scheme with public key K

- Each replica has a unique private key.
- Replicas can produce partial signatures for value v using their private key.
- Using $\mathbf{n} - \mathbf{f}$ partial signatures for v, one can produce a *threshold signature*.

Improving Aggregation with Threshold Signatures

Problem: An aggregated message of size c will have size $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.

- We have identical `Commit` messages from at-least $\mathbf{n} - \mathbf{f}$ replicas.
- Goal: aggregate these into a single message of size $\mathcal{O}(c)$ instead of $\mathcal{O}(c(\mathbf{n} - \mathbf{f}))$.
- Crucially: we want to aggregate the digital signatures!

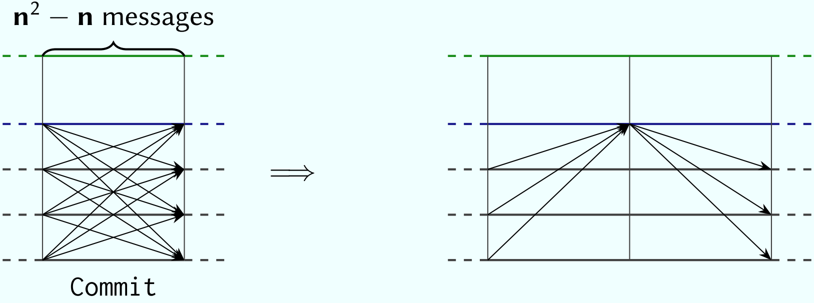
Solution: Using a $\mathbf{n} : \mathbf{f}$-threshold-signature scheme with public key K

- Each replica has a unique private key.
- Replicas can produce partial signatures for value v using their private key.
- Using $\mathbf{n} - \mathbf{f}$ partial signatures for v, one can produce a *threshold signature*.

Threshold signatures aggregate $\mathbf{n} - \mathbf{f}$ distinct signatures into a *single constant-sized* value.

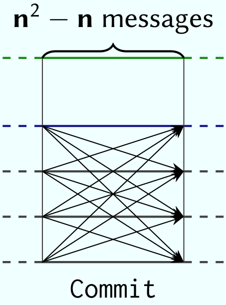
All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase

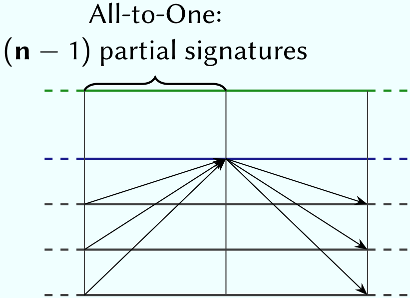


All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase

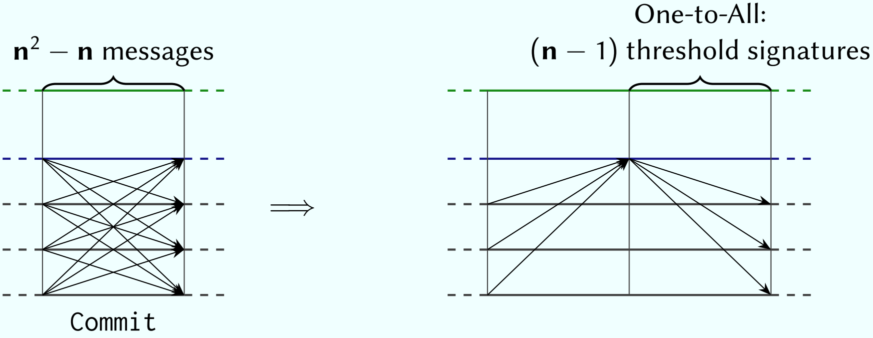


$\Longrightarrow$



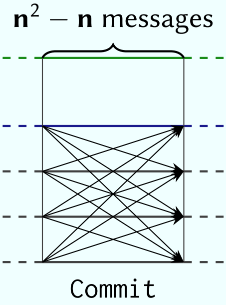
All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase

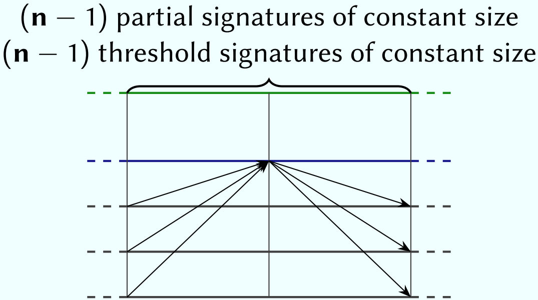


All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase

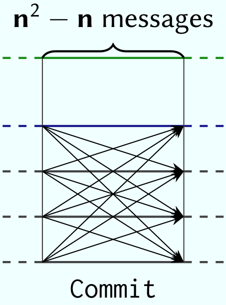


$\Longrightarrow$

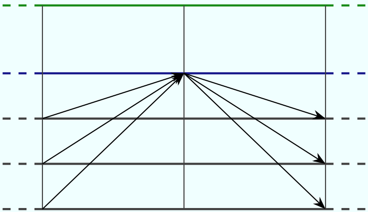


All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase



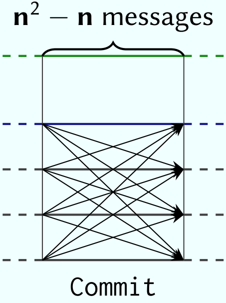
$\Longrightarrow$



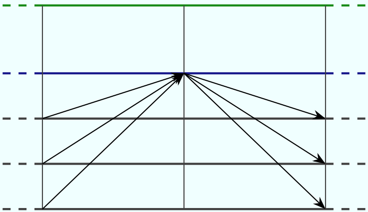
Effectively reduced communication from $\mathcal{O}(\mathbf{n}^2)$ to $\mathcal{O}(\mathbf{n})$.

All-to-one-to-All Aggregation with Threshold Signatures

Consider the commit phase



$\Longrightarrow$



Effectively reduced communication from $\mathcal{O}(\mathbf{n}^2)$ to $\mathcal{O}(\mathbf{n})$.
Similar change can be made to the prepare phase.

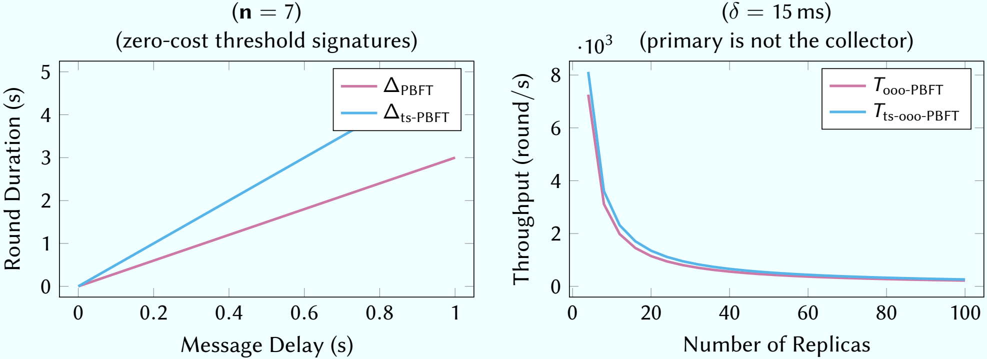
Using Threshold Signatures in PBFT

- Both prepare and commit phase: from $2(\mathbf{n}-1)^2$ to $4(\mathbf{n}-1)$ messages.
- Consensus from *three* to *five* rounds: higher consensus and client latencies.
- High *computational cost* for the aggregrator.
- Need recovery methods to deal with *faulty aggregators*.

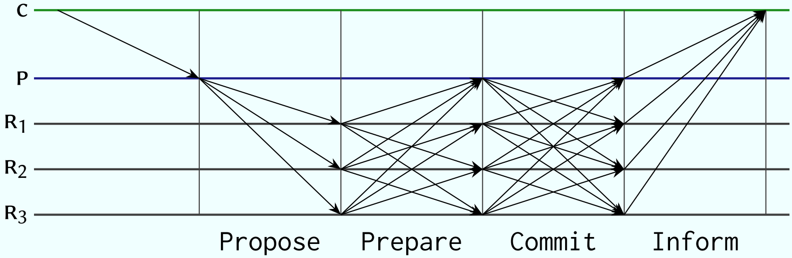
Using Threshold Signatures in PBFT

- Both prepare and commit phase: from $2(\mathbf{n}-1)^2$ to $4(\mathbf{n}-1)$ messages.
- Consensus from *three* to *five* rounds: higher consensus and client latencies.
- High *computational cost* for the aggregrator.
- Need recovery methods to deal with *faulty aggregators*.

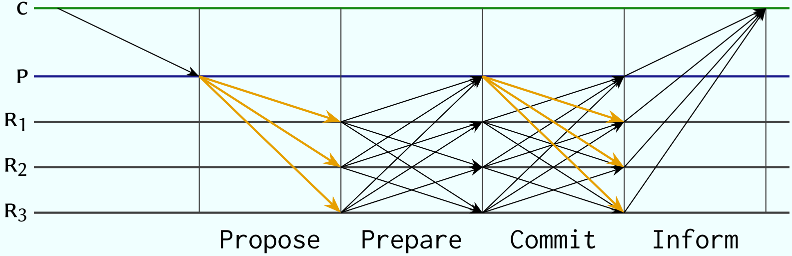
Assumption: $B = 100\,\mathrm{MiB/s}$, $s_\mathrm{t} = 4048\,\mathrm{B}$, $s_\mathrm{m} = 256\,\mathrm{B}$



Limitations of Primary-Backup Consensus

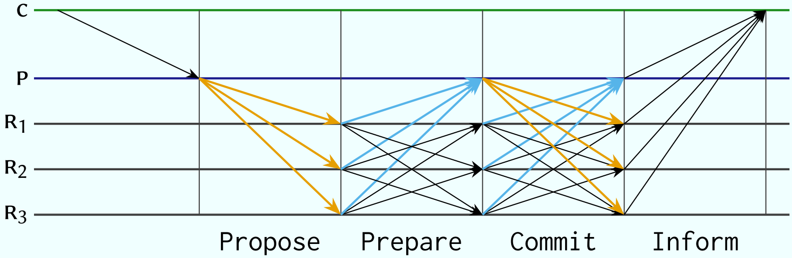


Limitations of Primary-Backup Consensus



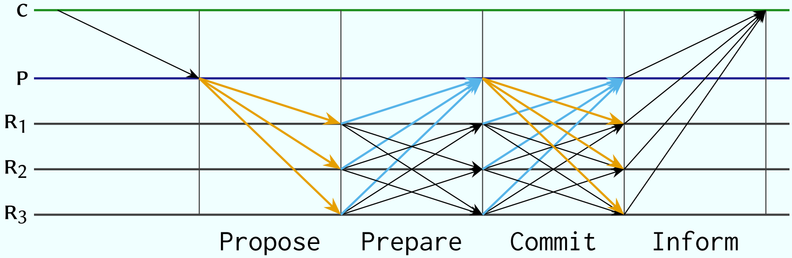
Primary Send $(\mathbf{n}-1)$ `Propose`, send $(\mathbf{n}-1)$ `Commit`.

Limitations of Primary-Backup Consensus



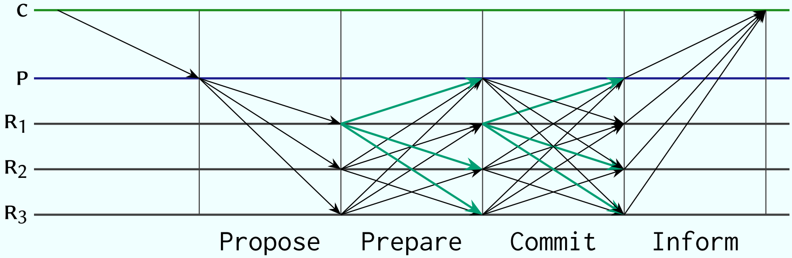
Primary Send $(\mathbf{n}-1)$ `Propose`, send $(\mathbf{n}-1)$ `Commit`.
Receive $(\mathbf{n}-1)$ `Prepare`, receive $(\mathbf{n}-1)$ `Commit`.

Limitations of Primary-Backup Consensus



Primary Send $(\mathbf{n}-1)$ Propose, send $(\mathbf{n}-1)$ Commit.
Receive $(\mathbf{n}-1)$ Prepare, receive $(\mathbf{n}-1)$ Commit.
Total: $\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}$.

Limitations of Primary-Backup Consensus



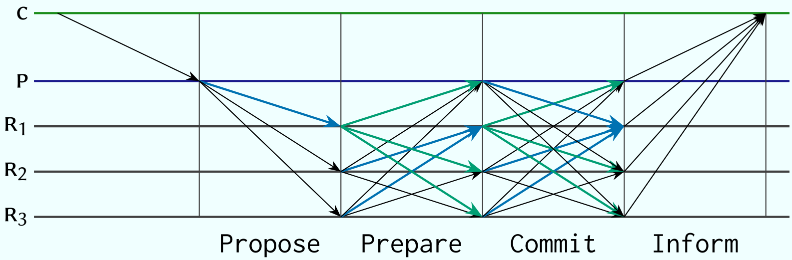
Primary Send $(\mathbf{n}-1)$ Propose, send $(\mathbf{n}-1)$ Commit.

Receive $(\mathbf{n}-1)$ Prepare, receive $(\mathbf{n}-1)$ Commit.

Total: $\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}$.

Backup Send $(\mathbf{n}-1)$ Prepare, send $(\mathbf{n}-1)$ Commit.

Limitations of Primary-Backup Consensus



Primary Send $(\mathbf{n}-1)$ Propose, send $(\mathbf{n}-1)$ Commit.

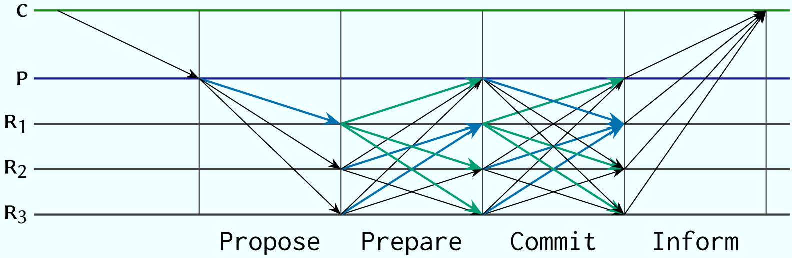
Receive $(\mathbf{n}-1)$ Prepare, receive $(\mathbf{n}-1)$ Commit.

Total: $\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}$.

Backup Send $(\mathbf{n}-1)$ Prepare, send $(\mathbf{n}-1)$ Commit.

Receive one Propose, receive $(\mathbf{n}-2)$ Prepare, receive $(\mathbf{n}-1)$ Commit.

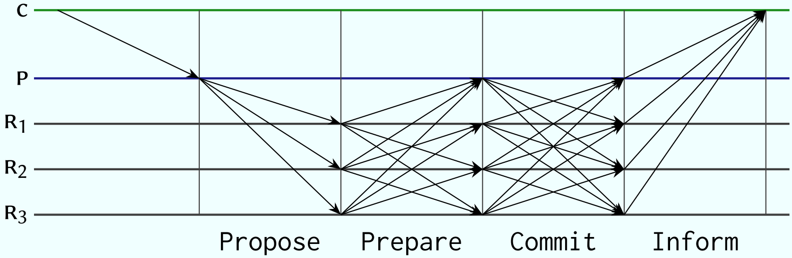
Limitations of Primary-Backup Consensus



Primary Send $(\mathbf{n}-1)$ Propose, send $(\mathbf{n}-1)$ Commit.
Receive $(\mathbf{n}-1)$ Prepare, receive $(\mathbf{n}-1)$ Commit.
Total: $\mathbf{m}(\mathbf{n}-1)s_t + 3(\mathbf{n}-1)s_m$.

Backup Send $(\mathbf{n}-1)$ Prepare, send $(\mathbf{n}-1)$ Commit.
Receive one Propose, receive $(\mathbf{n}-2)$ Prepare, receive $(\mathbf{n}-1)$ Commit.
Total: $\mathbf{m}s_t + 4(\mathbf{n}-1)s_m - s_m$.

Limitations of Primary-Backup Consensus



Bandwidth ratio between primary and backups

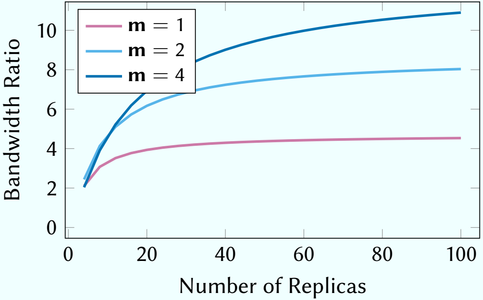
$$R_{\text{PBFT-}\mathbf{m}} = \frac{\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}}{\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}}.$$

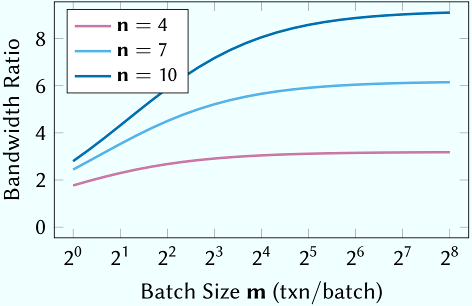
Limitations of Primary-Backup Consensus

Bandwidth ratio between primary and backups

$$R_{\text{PBFT-}\mathbf{m}} = \frac{\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}}{\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}}.$$

Assumption: $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$





Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

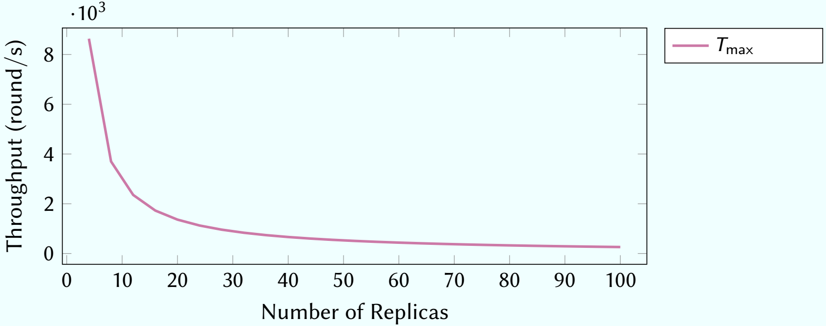
$$T_{\mathsf{max}} = \frac{B}{(\mathbf{n} - 1)s_{\mathrm{t}}}.$$

Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

$$T_{\mathsf{max}} = \frac{B}{(\mathbf{n}-1)s_t}.$$

Assumption: $B = 100\,\text{MiB/s}$, $s_t = 4048\,\text{B}$, $s_m = 256\,\text{B}$

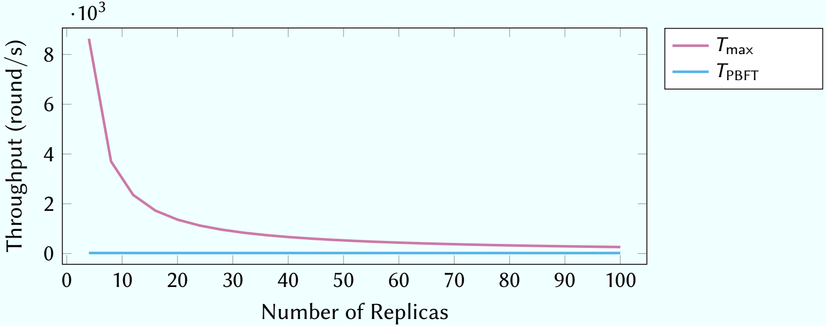


Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

$$T_{\mathsf{max}} = \frac{B}{(\mathbf{n}-1)s_{\mathsf{t}}}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$

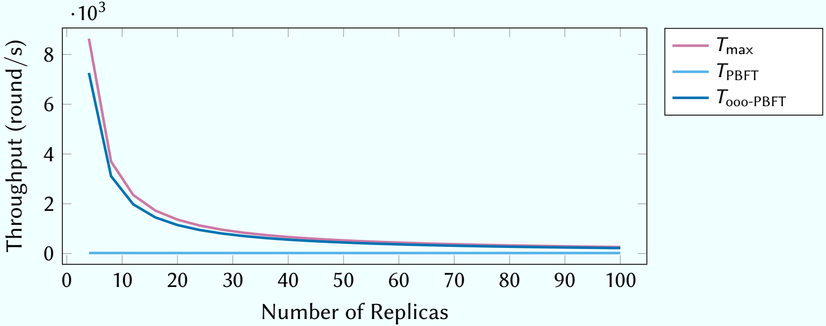


Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

$$T_{\mathsf{max}} = \frac{B}{(\mathbf{n} - 1)s_t}.$$

Assumption: $B = 100\ \mathrm{MiB/s}$, $s_t = 4048\ \mathrm{B}$, $s_m = 256\ \mathrm{B}$

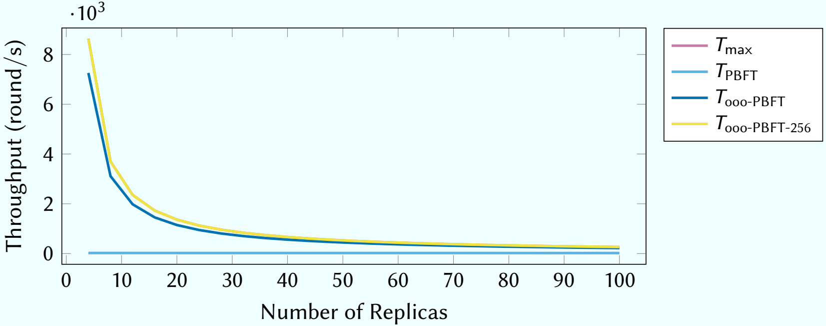


Maximum Throughput of Primary-Backup Consensus

Consider failure-free replication: Primary *only* proposes, no other communication.

$$T_{\mathsf{max}} = \frac{B}{(\mathbf{n}-1)s_{\mathsf{t}}}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $s_{\mathsf{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$



Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

Consider the communication of one of the $\mathbf{z}$ primaries.

Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

Consider the communication of one of the $\mathbf{z}$ primaries.

- As *primary* of its own instance:

  Send $(\mathbf{n} - 1)$ `Propose`, send $(\mathbf{n} - 1)$ `Commit`.

  Receive $(\mathbf{n} - 1)$ `Prepare`, receive $(\mathbf{n} - 1)$ `Commit`.

  Total: $\mathbf{m}(\mathbf{n} - 1)s_{\mathrm{t}} + 3(\mathbf{n} - 1)s_{\mathrm{m}}$.

Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

Consider the communication of one of the $\mathbf{z}$ primaries.

- As *primary* of its own instance:

  Send $(\mathbf{n}-1)$ `Propose`, send $(\mathbf{n}-1)$ `Commit`.

  Receive $(\mathbf{n}-1)$ `Prepare`, receive $(\mathbf{n}-1)$ `Commit`.

  Total: $\mathbf{m}(\mathbf{n}-1)s_t + 3(\mathbf{n}-1)s_m$.
- As *backup* of the other $\mathbf{z}-1$ instances ($(\mathbf{z}-1)$ times):

  Send $(\mathbf{n}-1)$ `Prepare`, send $(\mathbf{n}-1)$ `Commit`.

  Receive one `Propose`, receive $(\mathbf{n}-2)$ `Prepare`, receive $(\mathbf{n}-1)$ `Commit`.

  Total: $(\mathbf{z}-1)(\mathbf{m}s_t + 4(\mathbf{n}-1)s_m - s_m)$.

Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

Consider the communication of one of the $\mathbf{z}$ primaries.

- As *primary* of its own instance:

  Send $(\mathbf{n}-1)$ `Propose`, send $(\mathbf{n}-1)$ `Commit`.

  Receive $(\mathbf{n}-1)$ `Prepare`, receive $(\mathbf{n}-1)$ `Commit`.

  Total: $\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}$.
- As *backup* of the other $\mathbf{z}-1$ instances ($(\mathbf{z}-1)$ times):

  Send $(\mathbf{n}-1)$ `Prepare`, send $(\mathbf{n}-1)$ `Commit`.

  Receive one `Propose`, receive $(\mathbf{n}-2)$ `Prepare`, receive $(\mathbf{n}-1)$ `Commit`.

  Total: $(\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}})$.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

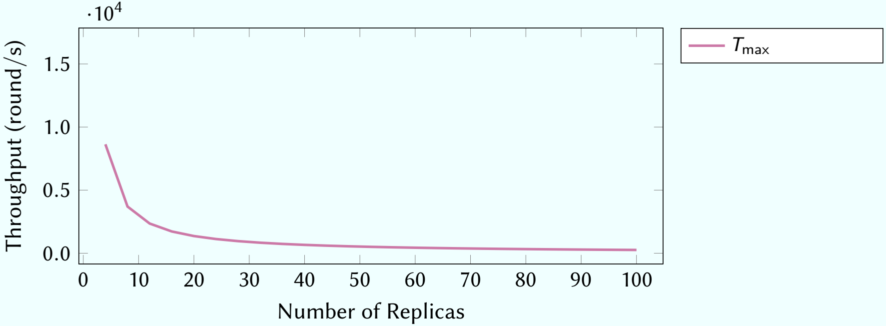
Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$



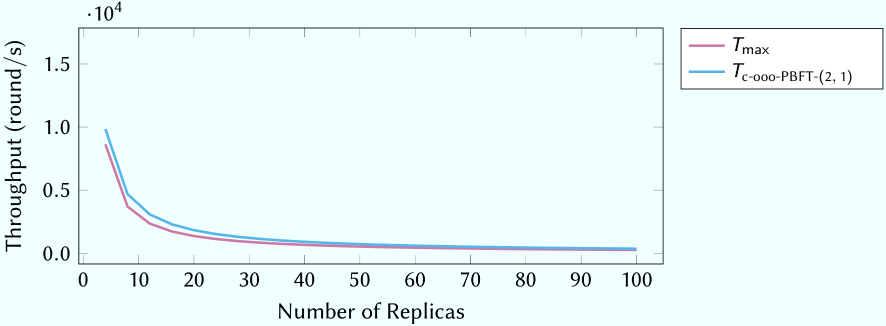
Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

Assumption: $B = 100\,\mathrm{MiB}/\mathrm{s}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$



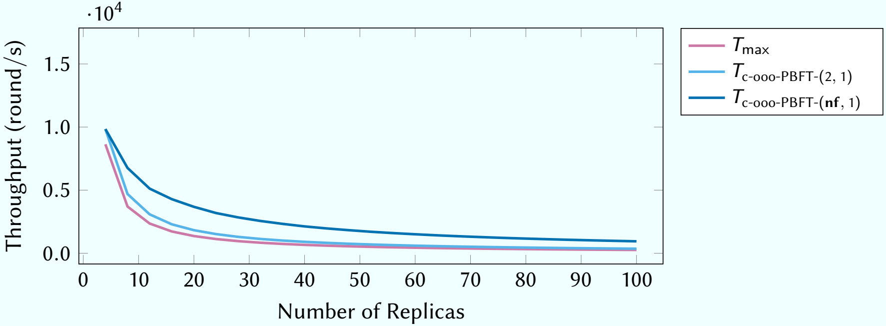
Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

Assumption: $B = 100\,\mathrm{MiB/s}$, $s_{\mathrm{t}} = 4048\,\mathrm{B}$, $s_{\mathrm{m}} = 256\,\mathrm{B}$



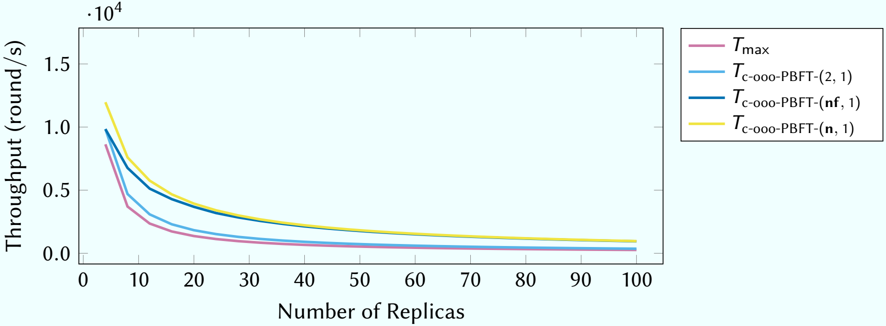
Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

Assumption: $B = 100\,\text{MiB}/\text{s}$, $s_{\mathrm{t}} = 4048\,\text{B}$, $s_{\mathrm{m}} = 256\,\text{B}$



Concurrent Consensus

Idea: Multiple instances of PBFT, each with a distinct primary

$1 \leq \mathbf{z} \leq \mathbf{n}$ primaries: $\mathbf{z}$ simultaneous rounds of consensus that decide the next $\mathbf{z}$ requests.

$$T_{\text{c-ooo-PBFT-}(\mathbf{z},\mathbf{m})} = \frac{\mathbf{z}\mathbf{m}B}{(\mathbf{m}(\mathbf{n}-1)s_{\mathrm{t}} + 3(\mathbf{n}-1)s_{\mathrm{m}}) + ((\mathbf{z}-1)(\mathbf{m}s_{\mathrm{t}} + 4(\mathbf{n}-1)s_{\mathrm{m}} - s_{\mathrm{m}}))}.$$

Assumption: $B = 100\ \mathrm{MiB/s}$, $s_{\mathrm{t}} = 4048\ \mathrm{B}$, $s_{\mathrm{m}} = 256\ \mathrm{B}$