Dissecting BFT Consensus: In Trusted Components we Trust!

Suyash Gupta  
UC Berkeley

Sajjad Rahnama  
Expolab  
UC Davis

Shubham Pandey  
Expolab  
UC Davis

Natacha Crooks  
UC Berkeley

Mohammad Sadoghi  
Expolab  
UC Davis
Why Should this Talk Interest you?

**Bad News**

Trusted Hardware cannot be used to efficiently reduce replication factor of BFT protocols to $2f+1$.

**Good News**

Trusted Hardware can be used to design more efficient and scalable $3f+1$ BFT protocols.
Replicated State Machine
Replicated State Machine

- **Safety** → Consistent log of operations.
- **Liveness** → Replicas should make progress.
- **Responsiveness** → Client should receive response.
Byzantine Fault Tolerant RSM

\[ n \text{ replicas} \& \text{ at most } f \text{ byzantine} \rightarrow n \geq 3f+1 \]

Run Byzantine Fault Tolerant (BFT) Consensus
Byzantine Fault Tolerance Consensus

Transaction

Alice

Result

Non-Equivocation Phase

Request is prepared

Persistence Phase

Request is committed

Execution Phase

n = 3f+1 replicas
Non-Equivocation

Create a *Prepare Quorum*:

**No two** prepare quorums can exist for different transactions at the same sequence number.

Every quorum *needs to intersect* in at least one honest replica.
Persistence

If a new leader is elected,
RSM should ensure that
previously committed requests persist.
Execution

Client needs $f+1$ matching responses.

Ensures execution by one honest replica.

Proof of request commitment not sufficient.
The Ugly Side of BFT

Crash Fault Tolerant Systems

2f+1 replicas

Byzantine Fault Tolerant Systems

3f+1 replicas

Equivocation is root cause of higher replication factor
Maybe Trusted Hardware Can help?

Keystone
Trusted Byzantine Fault-Tolerance Consensus

Trusted component *attest* order of each transaction.

Replicas cannot equivocate.

*A2M, TrInc, MinBFT, MinZZ, CheapBFT, Hotstuff-M, Damysus*

*Trust-BFT protocols* $\rightarrow 2f+1$ enough for safety
Trust-Byzantine Fault Tolerance Consensus

Transaction

Alice

Result

n = 2f+1 replicas

Non-Equivocation Phase

Request is prepared

Persistence Phase

Request is committed

Execution Phase
Trust-Byzantine Fault Tolerance Consensus

Alice

Transaction

Result

f+1 replicas vote prepare.

Request is prepared

f+1 replicas vote commit

Request is committed

Any replica that commits, executes.
So Are We Done?

Unfortunately No!
Hidden Pitfalls with Trust-BFT Protocols

- Algorithmic Pitfall
  - Limited Responsiveness
  - Loss of Safety under Rollbacks
  - Lack of Parallelism

- Measurement Pitfall
  - Instead of focusing on reducing replication ➔ Focus on increasing Throughput per Machine.
Limited Responsiveness

Transaction

Alice

Quorum Size = $f+1 = 2$

Honest replica that commits, executes.

Request is prepared

Request is committed

Honest replica sends Result
Alice Stuck!

Alice needs $f+1 = 2$ matching responses.

Alice receives only 1 response.
No progress for Alice

View Change

Resend

Need at least f+1

??
Lack of Parallelism

- Every message sent requires an attestation bound to specific sequence number.
- Replicas cannot run consensus on two transactions in parallel!
- We show that despite 2f+1 replicas, Trusted-BFT protocols are slower than BFT.
Loss of Safety under Rollbacks

- Trusted Enclaves can be rolled back!
  - On enclave rollback, safety cannot be guaranteed.

- Possible Solution? Make use of TPMs or persistent counters!
  - Too slow $\Rightarrow$ 180ms per access.
  - Very few writes $\Rightarrow$ TPMs allow at most 1 million writes.
  - Trust-BFT protocols require $O(n)$ accesses per consensus phase.
A novel suite of protocols.

Guarantee both liveness and responsiveness.

Require access to trusted component only once per consensus.

Employing TPMs to avoid enclave rollbacks is now much less expensive!
Magical Ingredients behind FlexiTrust Protocols

- Switch back to replication factor 3f+1.
  - Larger Quorums guarantee responsiveness.

- Trusted hardware accessed only by the primary before sending proposal.
  - Guarantees non-equivocation.
  - Permits replicas to participate in multiple consensus invocations in parallel.
  - Helps to reduce phases and communication.
Flexi-ZZ Protocol!

Single phase, Linear, Handles f failures, Only needs Trusted counters.
Evaluation on ResilientDB*
Throughput per Machine

<table>
<thead>
<tr>
<th>Replicas (in f)</th>
<th>Total Replicas (in n)</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flexi-ZZ</td>
<td>MinZZ</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>49</td>
<td>33</td>
</tr>
<tr>
<td>24</td>
<td>73</td>
<td>49</td>
</tr>
<tr>
<td>32</td>
<td>97</td>
<td>63</td>
</tr>
</tbody>
</table>

- MinZZ $\rightarrow$ Single phase like FlexiZZ but $n \geq 2f+1$.
- For these experiments, we deployed up to 80k clients.
Scalability

Number of replicas (f=8)

- $N = 17 \rightarrow \text{PBFT-EA, MinBFT, MinZZ, OPBFT-EA}$
- $N = 25 \rightarrow \text{PBFT, FlexiBFT, FlexiZZ}$

![Graph showing scalability with different numbers of replicas and throughput in transactions per second (txns/s). The graph has a line plot with markers indicating different consensus algorithms: PBFT-EA, MinBFT, MinZZ, OPBFT-EA, Flexi-BFT, Flexi-ZZ, and PBFT. The x-axis represents the number of replicas (in f), and the y-axis represents throughput (txns/s). The graph shows an improvement in scalability with a 95.5% increase at $N = 25$.](image-url)
Conclusions:

- Simply reducing replication will not yield higher throughput.
- Existing Trust-BFT protocols limit responsiveness and scalability.
- **FlexiTrust** protocols advocate meaningful application of BFT consensus.

Reach me:

- Twitter: suyash_sg
- Email: suyash.gupta@berkeley.edu
- Web: https://gupta-suyash.github.io/