Sync HotStuff: Simple and Practical State Machine Replication

Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, Maofan Yin
BFT State Machine Replication: Why?
BFT State Machine Replication: Definition

- All-to-all, reliable, bidirectional channels
- f out of all n nodes could exhibit arbitrarily faulty behavior
- (Replica Coordination) other n-f correct replicas process the same sequence of requests

Any two replicated sequence $s_1$ and $s_2$: $s_1 \subseteq s_2 \lor s_2 \subseteq s_1$
BFT SMR $\rightarrow$ BFT Consensus

- *(Agreement)* All correct nodes must agree on the same value. "Safety"
- *(Termination)* All nodes must eventually decide on an output value. "Liveness"
- *(Validity)* If all correct nodes receive the same input value, then they must all output that value. $\rightarrow$ "Non-triviality"
“In this paper, it is shown that every protocol for this problem has the possibility of non-termination, even with only one faulty process.”
Model & Known Solutions

- Termination → “Probability of 1”
  - Asynchronous model (Ben-Or ’83)

- Always safe no matter what, and terminate when network is synchronized
  - Partially synchronous model (DLS ’88)

- Use synchronous assumption
  - Synchronous model (LSP ’82)
BFT Consensus: Why Another Protocol?
BFT Consensus: Why Another Protocol?

The Saddest Moment, Mickens 2013
HotStuff '19 (Partially Sync Protocol)
BFT Consensus: Why *Synchronous*?
Sync BFT: Now and Then

“Forget about it...”

- Why not partially sync?
- Lock-step
  - Low throughput
  - High latency Δ
- Impractical Δ assumption

“...or not?”

- Liveness always guaranteed
- Introduce asynchrony
  - Higher throughput
  - Low latency (optimistic)
- More practical model (sluggish mobile)
Sync HotStuff: Contributions

- Inspired by chain-style BFT consensus
- Uses Quorum Certificates (QC) and the locking mechanism as in HotStuff
- Sync HotStuff:
  - Leader-based: a long-standing leader and views
  - Assumes standard synchronous model
  - Not a lock-step protocol — commits are scheduled with asynchrony
  - Near-optimal latency: $2\Delta + O(\delta)$ (e.g. Dfinity: $9\Delta + O(\delta)$; PiLi: $40\Delta-65\Delta$)
- Extensions:
  - Add-on$_1$: Sluggish Mobile model: allow at most $(d < f)$ honests replicas to violate the $\Delta$ bound at any given time
  - Add-on$_2$: Optimistic Responsiveness ($O(\delta)$ latency when $\frac{3}{4}$ votes in-time)
Sync HotStuff: Practical Performance

Fig. 10: Performance as function of faults at $\Delta = 50$ ms, optimal batch size, and 0/0 payload.
Sync HotStuff: Data Structure

Blockchain-style
Sync HotStuff: Quorum Certificate (QC)

Proof of the existence of f+1 votes

We can expect at most f+1 responses within Δ time

Byz. Replica

Correct Replica

f+1: QC

≥1
Quorum Certificate (QC)

Certified Block

\[ B_x \xrightarrow{\text{hash}} \text{cmd} \xrightarrow{\text{hash}} B_y \xrightarrow{\text{hash}} \text{cmd} \]

\[ C_V(B_k) \]

QC of \( B_k \) formed in view \( v \)
Sync HotStuff: Steady State Protocol

Blockchain-style

Diagram of blockchain-style synchronization with delays and states represented as nodes $B_1$ to $B_6$.
Sync HotStuff: Steady State Protocol

1) $2\sigma$ is bounded by $2\Delta$, which guarantees sufficient time for detecting equivocation.

2) always needs to forward a message if it comes from a single source!
Sync HotStuff: View Change Protocol

What if the Leader is Byzantine?

Before $2\Delta$ timer goes off, if there is an conflict block, blame the leader! (Leader is dishonest)

Or if the leader is not able to propose $p$ blocks within $(2p + 4)\Delta$ during of time, blame the leader! (Leader makes no progress)
Sync HotStuff: View Change Protocol

- **Quit view.** Abort all commit timers in the current view
  - Equivocation: the blame message comes with the proof of equivocation; quit view
  - Timeout: broadcasts a blame to others (don’t quit view now, until...)
  - Got f +1 blames: broadcast them, and also quit view

Remark: if an honest node quits view...

- due to equivocation, all honest nodes will quit view **within Δ time**
- due to timeout, all honest nodes will quit view **within Δ time**

**View change is strictly synchronized!**
Sync HotStuff: View Change Protocol

- **Quit view.** Abort all commit timers in the current view
  - Equivocation: the blame message comes with the proof of equivocation; quit view
  - Timeout: broadcasts a blame to others (don’t quit view now, until...)
  - Got f +1 blames: broadcast them, and also quit view

- **Status.** Wait for \( \Delta \) time (to collect enough votes so as to update highest certified block). Pick a highest certified block \( C_v(B_{k'}) \), lock on \( C_v(B_{k'}) \), send \( C_v(B_{k'}) \) to the new leader \( L' \), and enter the next view (\( v + 1 \)).

- **New-view.** The new leader \( L' \) waits for \( 2\Delta \) time after entering view \( v + 1 \) (\( \Delta \) Status wait + \( \Delta \) Status message delay) and broadcasts \(<\text{new-view, v + 1, } C_v(B_{k'})>_{L'} \) where \( C_v(B_{k'}) \) is a highest certified block known to \( L' \).

- **First vote.** Upon receiving \(<\text{new-view, v + 1, } C_v(B_{k'})>_{L'} \), if \( C_v(B_{k'}) \) has a rank equal to or higher than r’s locked block, forward \(<\text{new-view, v + 1, } C_v(B_{k'})>_{L'} \) to all other replicas and broadcast \(<\text{vote, } B_{k'}, v + 1>_r \). (Make sure all correct replicas can form the QC of \( B_{k'} \).)
Lemma 1. If an honest replica $r$ directly commits $B_i$ in view $v$, then
- no equivocating block is certified in view $v$
- every honest replica locks on a certified block that ranks equal to or higher than $C_v(B_i)$ before entering view $v + 1$. (Carry on the lock on the last committed block in a view)
Lemma 2 (Unique Extensibility). If an honest replica directly commits $B_l$ in view $v$, then any certified block that ranks equal to or higher than $C_v(B_l)$ must extend $B_l$. (The lock is extended)

By contradiction, assume $S \neq \emptyset$ contains the certified blocks that doesn't extend $B_l$.

Let $C_{v*}(B_{k*})$ be the lowest ranked block in $S$, then $v^* > v$, otherwise it contradicts with Lemma 1 (i).

$B_{k*-1}$ also does not extend $B_l$.

Some honest node must

- vote for $B_{k*}$ in view $v^*$ <new-view, $v^*$, $C_{v^*}(B_{k*})$> where $v^* < v^*$,
  - by lemma 1 (i) $C_{v^*}(B_{k*})$ must rank higher than or equal to $C_v(B_l)$
  - not possible due to the minimality of $C_{v*}(B_{k*})$
- or vote for <propose, $B_{k*}$, $v^*$, $C_{v*}(B_{k*}-1)$>
  - Not possible due to the minimality of $C_{v*}B_{k*}$
Sync HotStuff: View Change Protocol

- **Theorem 3 (Safety).** No two honest replicas commit different blocks at the same height.
  - Follows easily from Lemma 2

- **Lemma 4 (Liveness).** All honest replicas keep committing new blocks.
  - View changes are synchronized
  - Leader is uniquely determined by the view number
  - View change until an honest replica is the leader
Sync HotStuff: More...

- **Extensions:**
  - Add-on₁: **Sluggish Mobile** model: allow at most \( d < f \) honests replicas to violate the \( \Delta \) bound at any given time
  - Add-on₂: **Optimistic Responsiveness** \( O(\delta) \) latency when \( 3/4 \) votes in-time

Requires just few simple changes to the protocol/proof.

More in our paper!
Thanks
More to discovered in our paper!

About Me: https://www.cs.cornell.edu/~tedyin/