Order-Fairness for Byzantine Consensus

Mahimna Kelkar
Cornell University and Cornell Tech

Joint work with
Fan Zhang,
Steven Goldfeder,
and Ari Juels
State Machine Replication (SMR)
also Byzantine consensus, linearly-ordered log

Transactions from clients → Consensus Nodes → Agree on a consistent ordered transaction log
State Machine Replication (SMR)
also Byzantine consensus, linearly-ordered log

**Consistency** or **Safety**
Honest nodes output the same log

**Liveness**
New TXs are incorporated soon
State Machine Replication (SMR)
also Byzantine consensus, linearly-ordered log

• **No restriction** on the actual ordering

• Often **easy to manipulate**
• Almost all classical consensus protocols are leader-based
  • E.g., PBFT, Paxos, Hotstuff etc.

  • Leader node **can propose any ordering**
    • Adversarial leader can arbitrarily manipulate ordering

• No previous protocol guarantees fair ordering.
Why is *fair ordering* important?
Why is *fair ordering* important?

- 2014 exposé on *high-frequency trading* on Wall Street.
- HFT characteristics
  - Front-running
  - Arbitrage
- Investigation and fines after Lewis’ book (FBI, SEC, etc.)
Why is fair ordering important?

- HFT back in a new form on decentralized exchanges
- Wild west without much regulation

Daian et al. (IEEE S&P 2020)
Why is fair ordering important?

Independent Theoretical Motivation

• Natural Analog of Validity condition in Byzantine Agreement (BA)
• Validity forgotten when BA generalized to SMR

If all honest nodes are input value \( v \), then all honest nodes will agree on \( v \).

Agreement Validity

If all honest nodes are input \( m_1 \) before \( m_2 \), then all honest nodes will agree on \( m_1 \) before \( m_2 \).

Order-Fairness
Comparison to current techniques

• Censorship Resistance [HoneybadgerBFT, Omniledger etc]
  • Reordering and insertion still possible

• Random leader election [Algorand, Ouroborous etc]
  • Adversarial leader can still order unfairly

• Threshold Encryption [HoneybadgerBFT]
  • Transactions ordered before content is revealed
  • Can still reorder transactions from colluding client first
  • Possible to blindly reorder

Order-Fairness is strictly stronger than previous notions
Defining Fair Ordering
Model

• Permissioned system with $n$ nodes, $f$ of which may be adversarial

• Clients can collude with protocol nodes
Model

• **External Network**
  - Communication between clients and protocol nodes
  - Clients send transactions to **all** nodes
  - Adversary $\mathcal{A}$ *not* in charge of message delivery

• **Internal Network**
  - Communication amongst protocol nodes
  - Adversary $\mathcal{A}$ handles all message delivery
**Model: Synchrony Definitions**

\[ \Delta_{ext} \text{ - External Synchrony} \]

\[ \Delta_{int} \text{ - Internal Synchrony} \]

**If** a transaction is input to some node in round \( r \),
**then** all honest nodes will receive it as input by round \( r + \Delta_{ext} \).

**If** a message is sent by an honest node in round \( r \),
**then** all recipient(s) will receive it by round \( r + \Delta_{int} \).
So how do we define the **fair ordering**?

**Definition (informal):** $\gamma$-Receive-Order-Fairness

If $\gamma n$ nodes are input $m_1$ before $m_2$, then all honest nodes will deliver $m_1$ before $m_2$.

$\frac{1}{2} < \gamma \leq 1$
Condorcet Paradox

• Global ordering can be **non-transitive** even when individual orderings are transitive
Condorcet Paradox

- Global ordering can be non-transitive even when individual orderings are transitive.

Alice:
1. x
2. y
3. z

Bob:
1. y
2. z
3. x

Carol:
1. z
2. x
3. y

x ≪ y
Condorcet Paradox

- Global ordering can be non-transitive even when individual orderings are transitive
Condorcet Paradox

• Global ordering can be non-transitive even when individual orderings are transitive
Condorcet Paradox

• Global ordering can be **non-transitive** even when individual orderings are transitive

Alice

1. $x$
2. $y$
3. $z$

Bob

1. $y$
2. $z$
3. $x$

Carol

1. $z$
2. $x$
3. $y$

$x \ll y$

$y \ll z$

$z \ll x$

Cyclic Ordering!
Theorem (informal): Impossibility of Receive-Fairness

For any $n, f \geq 1$ and $\gamma$, no protocol can achieve all of consistency, liveness and $\gamma$-receive-order-fairness when $\Delta_{ext} \geq n$. 
Block-Order-Fairness

Definition (informal): \(\gamma\)-Block-Order-Fairness

If \(\gamma n\) nodes are input \(m_1\) before \(m_2\), then all honest nodes will deliver \(m_1\) no later than \(m_2\).
Block-Order-Fairness

Definition (informal): $\gamma$-Block-Order-Fairness

If $\gamma n$ nodes are input $m_1$ before $m_2$, then all honest nodes will deliver $m_1$ no later than $m_2$.

- Key Idea: Deliver transactions with non-transitive ordering in the same block
Why can’t we just order based on **median** timestamp?

- A single adversarial node can cause unfair ordering

<table>
<thead>
<tr>
<th>Round Number</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( t_{x_1} )</td>
<td>( t_{x_1} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( t_{x_2} )</td>
<td>( t_{x_2} )</td>
<td></td>
<td></td>
<td>( t_{x_1} )</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( t_{x_2} )</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>( t_{x_1} )</td>
<td></td>
<td>( t_{x_1} )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>( t_{x_2} )</td>
<td>( t_{x_2} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why can’t we just order based on median timestamp?
• A single adversarial node can cause unfair ordering

<table>
<thead>
<tr>
<th>Round Number</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(tx_1)</td>
<td>(tx_1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(tx_2)</td>
<td>(tx_2)</td>
<td></td>
<td>(tx_1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(tx_2)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>(tx_1)</td>
<td>(tx_1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>(tx_2)</td>
<td>(tx_2)</td>
<td></td>
</tr>
</tbody>
</table>

\[2 = \text{med}(tx_1) \leq \text{med}(tx_2) = 3\]
Why can’t we just order based on **median** timestamp?

- A single adversarial node can cause unfair ordering

<table>
<thead>
<tr>
<th>Round Number</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$tx_1$</td>
<td>$tx_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$tx_2$</td>
<td>$tx_2$</td>
<td></td>
<td></td>
<td>$tx_2$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$tx_1$</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>$tx_1$</td>
<td>$tx_1$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>$tx_2$</td>
<td>$tx_2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$3 = med(tx_1) \leq med(tx_2) = 2$
Fair Ordering Protocols
Aequitas: A Fair-Ordering Protocol

Inputs:
- $tx_1$
- $tx_2$
- $tx_3$

Gossip Stage

Agreement Stage

Finalization Stage

Output:
- $B_0$
- $B_1$
- $B_k$
The Gossip Stage

(1) Honest nodes broadcast transactions they to all nodes as they are received

(2) Honest nodes store broadcasts received from other nodes in local logs $\text{locallog}_i^j$ contains $i$’s view of broadcasts by $j$

Guarantees that honest nodes have consistent local logs
The Gossip Stage

• **FiFo (First-In-First-Out) Broadcast**
  • Messages broadcast by an honest sender are delivered in the *same order* as they were broadcast
  • Messages broadcast by an adversarial sender are delivered in a consistent order by all honest nodes

• Can be realized from standard reliable broadcast [HDvR 07]
Agreement Stage

• Agree on which local logs to use to order a transaction

• Can be done using standard Byzantine agreement

Guarantees that honest nodes use the same local logs to finalize a transaction
Finalization Stage

• The finalization stage orders the transaction in the final output log

• Leaderless
  • No extra communication
Finalization Stage

Ordering two transactions

- If many (e.g., $\gamma n - f$) local logs contain $tx'$ before $tx$, then $tx$ is said to wait for $tx'$

- Relations between transactions are viewed in a **dependency** or **waiting graph**.
  - Vertices represent transactions
  - Edge $(a,b)$ represents $b$ waiting for $a$
Leaderless Finalization

What if there is no clear winner in the two transactions?

Two problems to solve

1. Graph may not be complete or even connected.
   - Some transactions may not be comparable

2. Graph may not be acyclic.
Leaderless Finalization

Key Idea

• Wait for common descendant for transactions without an edge in the graph
• Order using maximum number of dependents
Leaderless Finalization

- Graph can still have cycles
- To get a total ordering, compute the **condensation** graph by collapsing the strongly-connected components
- Deliver transactions in the same component into the **same block**.
• Synchronous protocol requires \( n > \frac{2f}{2\gamma-1} \)
  
  i.e., \( n > 2f \) even when \( \gamma = 1 \)

• Asynchronous protocol requires \( n > \frac{4f}{2\gamma-1} \)
Some Caveats

• Only Achieves **Weak-Liveness**
  
  • New transactions must be input *sufficiently late* in order to deliver current transactions
  
  • Conventional Liveness achieved when external network has *small synchrony bound*
Some Caveats

• Adversary can unfairly order if it controls the entire Internet, i.e. if it can also control a client’s connection to the consensus protocol nodes

• In our modeling, this is handled by assuming adversary does not control the external network
A general order-fairness compiler

• FiFo-broadcast and Byzantine Agreement are weak primitives
  • They can be realized from any consensus protocol

• General compiler that takes any consensus protocol and transforms it into one that also provides order-fairness
Final Thoughts

• Our work is the first to formalize order-fairness and provide protocols that realize it

• Order-Fairness is important for many blockchain applications
  • Decentralized exchanges (2.4 billion USD market)
  • ICO token sales (12 billion USD market)
  • Decentralized Finance in general
Thank you

mahimna @ cs.cornell.edu

ia.cr/2020/269