Algorand: Scaling Byzantine Agreements for Cryptocurrencies

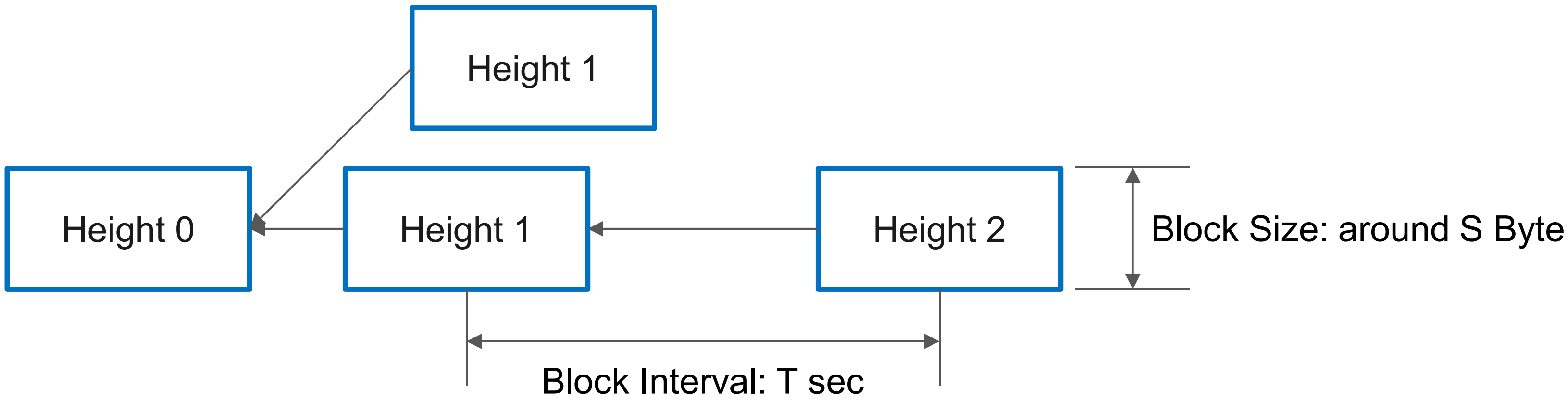
Hyunjin Kim
KAIST

Introduction

- Previous Presentations: “How secure PoW is?”
  –Attack on Bitcoin Mining pool
  –Attack on Bitcoin Communication
  –Attack on Bitcoin Consensus mechanism

➔ Then, “How fast PoW data generation is?”

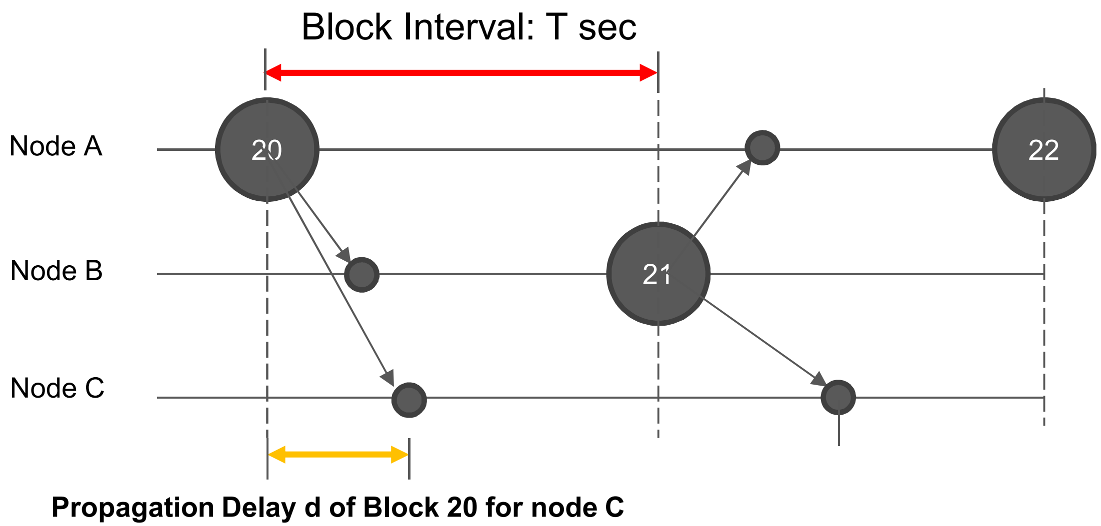
Transaction Throughput of PoW



- Transaction Processed with average speed of S/E[T] byte/sec
- For Bitcoin, protocol sets S: 1MB, E[T]: 600sec

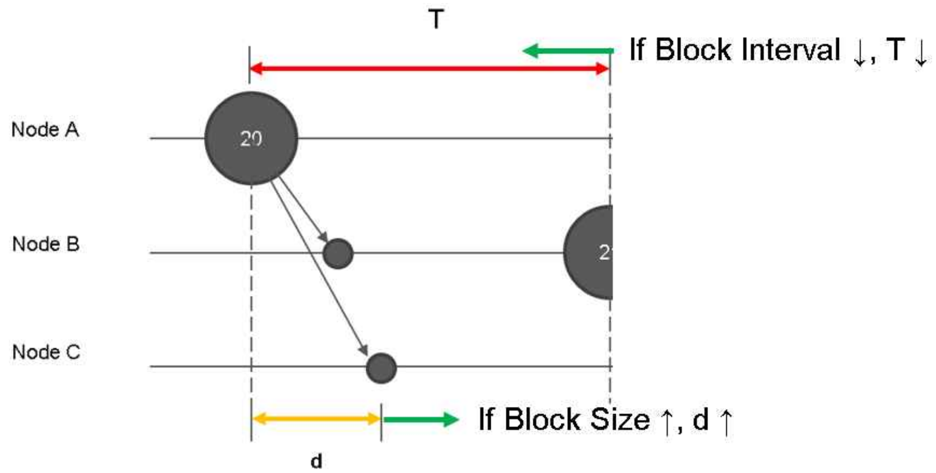
Changing S: ∞ or E[T]: 1/ ∞?

- No, Because of the Propagation Delay.



Wasted Hash Power

- In next block generation,
  node C wastes d/T of its hash power



➔ Cannot improve performance dramatically by Block size increment or Interval decrement

Content

- Various Consensus Mechanisms on Permissionless Blockchain
  - Proof of X
  - Hybrid Consensus
  - Multiple Committee Consensus
- Algorand
  - VRF and cryptographic sortition
  - Block Proposal
  - Gossip Protocol
  - Byzantine Agreement*

Various Consensus Mechanisms

From SoK: Consensus in the Age of Blockchains
(S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, G. Danezis)

Proof-of-X

Lottery based on 'Undeniable Proof'

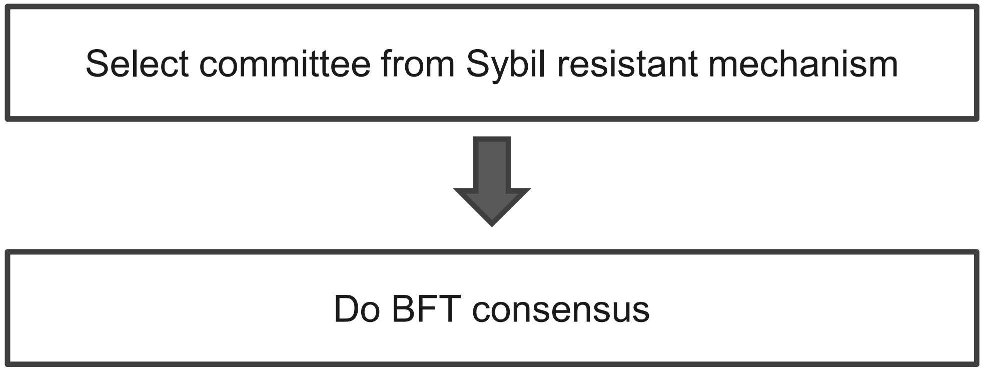
Proof of Stake: 'Undeniable Proof' = logged coin
Proof of Capacity: 'Undeniable Proof' = signed distributed file storage proof
Proof of Elapsed Time: 'Undeniable Proof' = signed waiting time

Hybrid Consensus

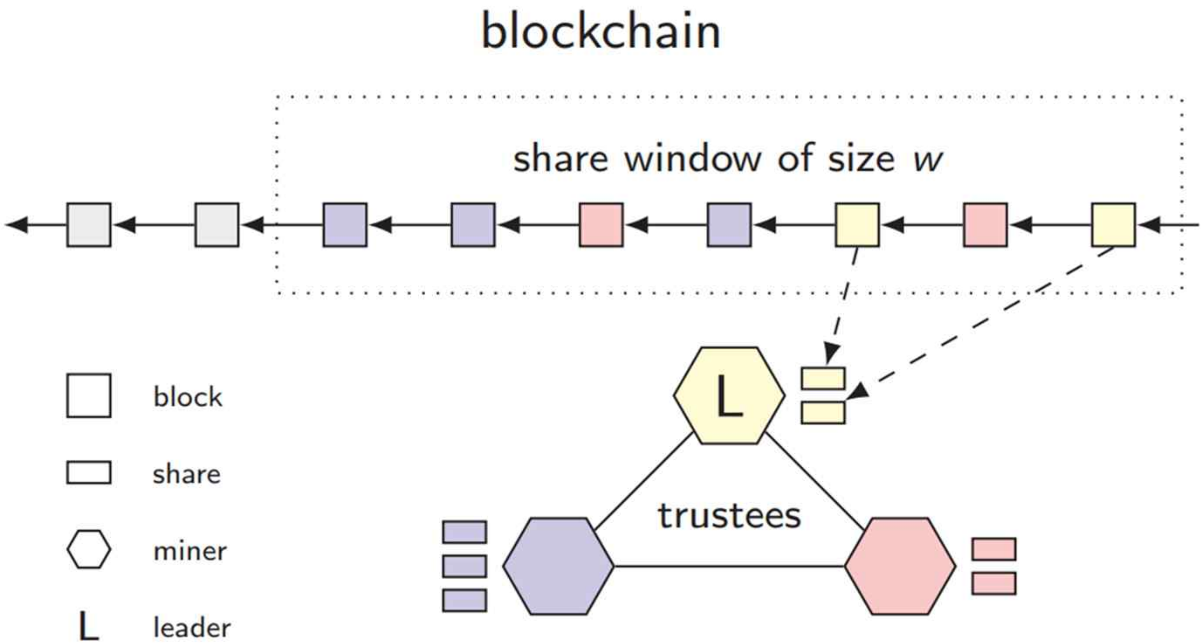
Previous Two Approaches

Proof of X	BFT consensus
Sybil Resistant, but slow	Fast, but no Sybil Resistant

Hybrid Consensus



Example: ByzCoin

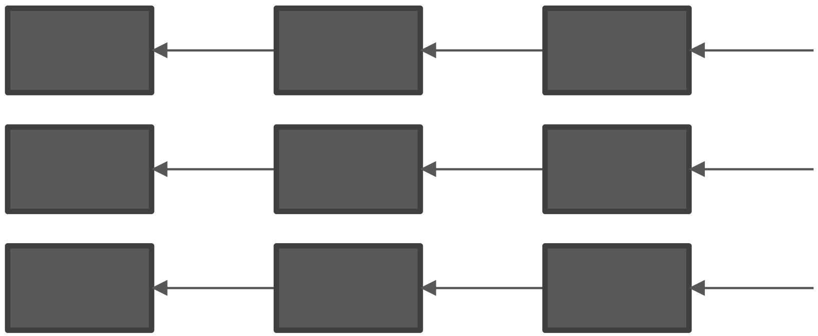


1. Miner of Accepted Block get voting weight for each block

2. Miner with voting right do PBFT consensus for one block

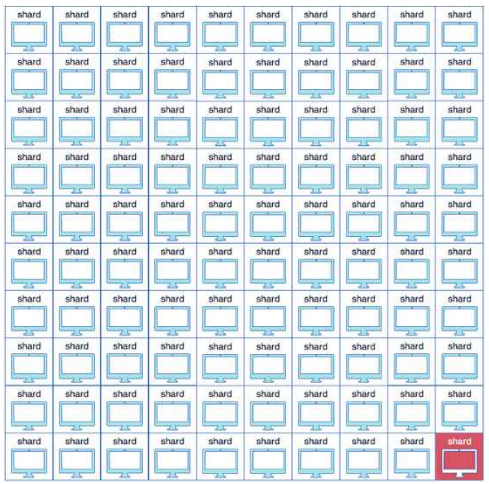
Multiple Comittee Consensus

- Simple solution for transaction throughput: Make another chain, each miner only manage one chain



What can be Problem?

Challenge 1 on Multiple Committee



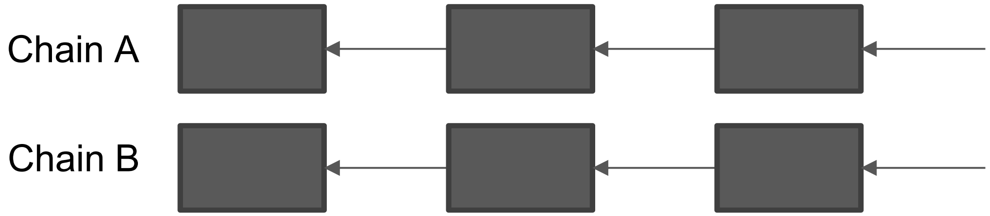
1% Attack

> “In 100 shards system, it takes only 1% of network hash rate to dominate the shard.”

Solution: Well randomized miner distribution mechanism

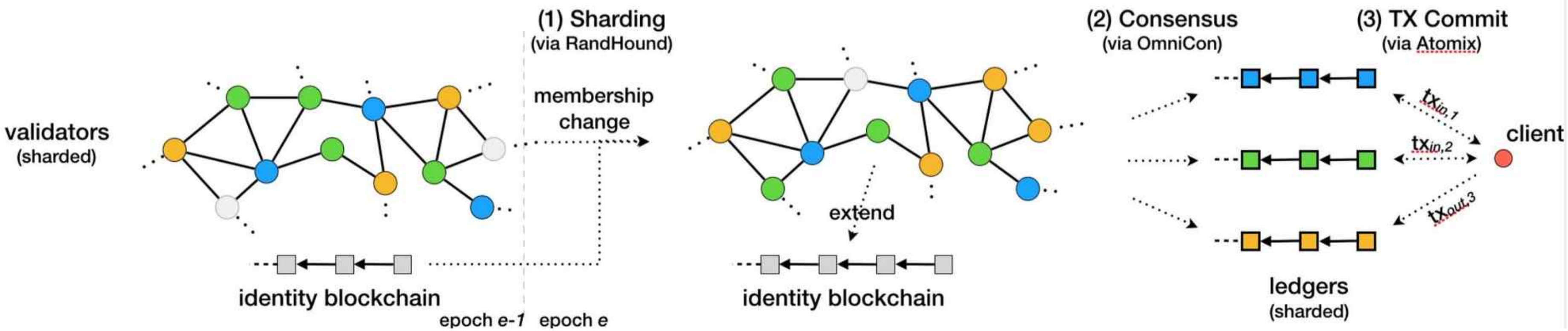
Challenge 2 on Multiple Committee

- How address chain A and chain B communicate?



Solution: Periodic global block generation, consensus mechanism between A and B

Example: Omniledger



Performance Comparison - PoW

System	Throughput	Latency
Bitcoin	7tx/s	600s
Bitcoin-NG	7tx/s	<1s
GHOST	-	-
DÉCOR+HOP	30tx/s	60s
Spectre	-	-

Performance Comparison - PoX

System	Committee Formation	Throughput	Latency
Ouroboros	Lottery	257.6tx/s	20s
Praos	Stake	-	-
Snow-white	Stake	100-150tx/s	-
PermaCoin	PoW/PoR	-	-
SpaceMint	PoS	-	600s
Intel PoET	Hardware Trust	1000tx/s	-
REM	Hardware Trust	-	-

Performance Comparison - Hybrid

System	Committee Formation	Throughput	Latency
ByzCoin	PoW	1000tx/s	10-20s
Algorand	Lottery	90tx/h	40s
Hyperledger	Permissioned	110k tx/s	<1s
RSCoin	Permissioned	2k tx/s	<1s
Elastico	PoW	16 blocks/110s	110s/16blocks
Omniledger	PoW/PoX	10k tx/s	1s
Chainspace	Flexible	350tx/s	<1s

Algorand: Scaling Byzantine Agreements for Cryptocurrencies

Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, Nickolai Zeldovich
ACM SOSP'17

Why Algorand is explained, instead of other?

1. Good tx throughput without sharding mechanism
   –Sharding can be independently applied over Algorand mechanism

2. Less centralized tendency from less incentivization

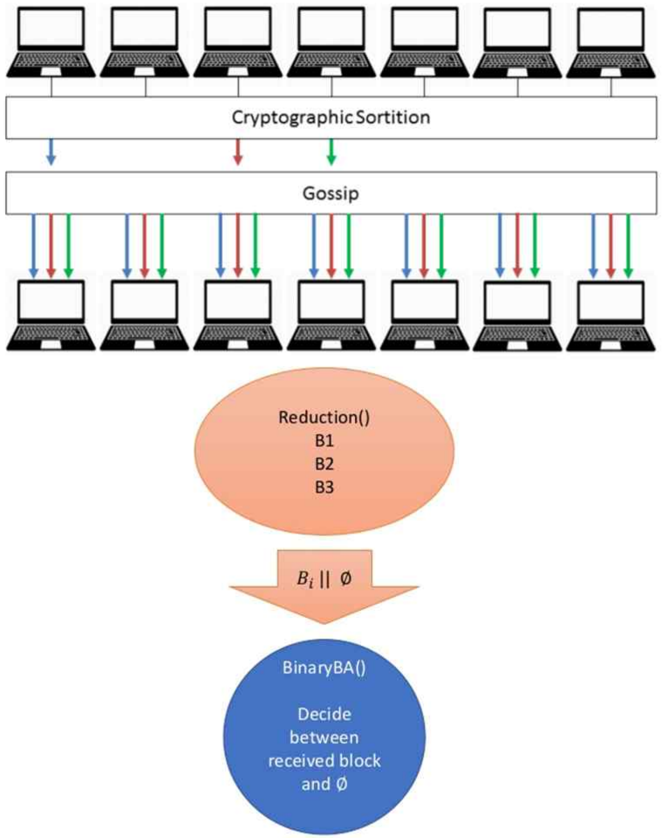
Purpose of Algorand

1. Short latency with high transaction throughput
   -transaction processing under 1 minute
2. Scaling to many users, resistant to Sybil attacks
3. No divergent view even in temporarily partitioned network

Design Overview

1. Block Proposal Phase
   → block proposal based on VRF
   → propagated by gossip protocol

2. Agreement Phase
   → committee selection based on VRF
   → selected committee



Assumptions

- Adversary’s money(coin) should not be over ⅓ of total money

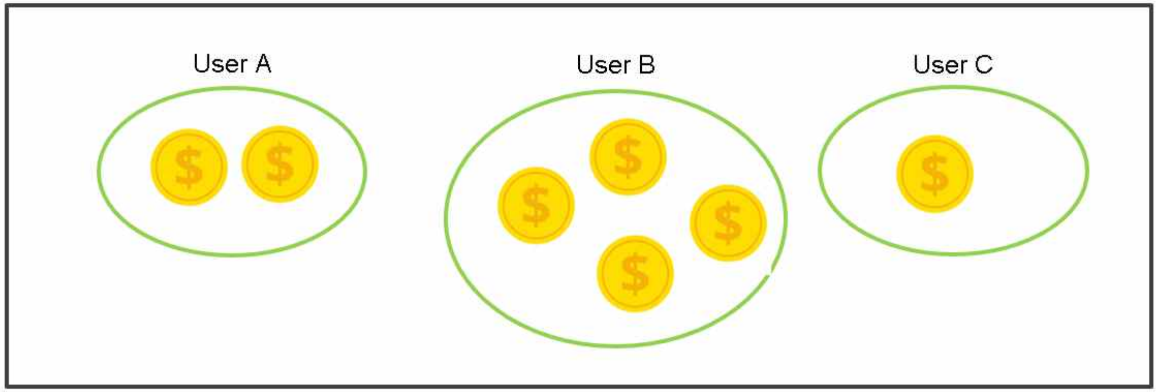
- Safety
  –If one honest user accepts transaction A, then the any transaction accepted from all honest users will be based on the log containing transaction A
  –This should be hold even for temporarily partitioned users (disconnected users)
  –Safety holds on weak synchrony
    long asynchronous periods(less than 1 day~1 week),
    followed by some strongly synchronous periods(more than few hour~1 day)

Assumptions

- Liveness
  - –All hones nodes make progress of logs within roughly one minute
  - –Liveness holds on strong synchrony
    - Most honest users(95%) can receive message of other honest users on bounded time

Cryptographic Sortition with VRF

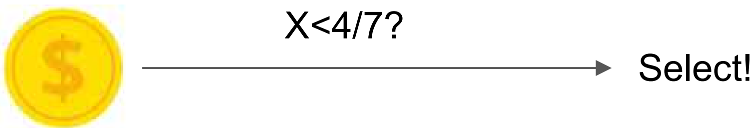
Someone want to randomly select about 4 tokens from total token,
How to do that?



Cryptographic Sortition with VRF

1. For each  , write random number in [0, 1)
2. If the number is less than 4/7, select it.

➔ So Simple!



Verifiable Random Function

1. Random Hash Generation: $(\text{Hash}, \pi) \leftarrow \mathrm{VRF}_{sk_A}(s)$
   (Hash: random value, π: proof, sk_a: a's secret key, s: string)
2. Hash Generation Proof: $\mathrm{VerifyVRF}_{pk_a}(\text{Hash}, \pi, s)$
   ⇒ Prove with a's public key and π, whether Hash is generated from s and π

Why VRF is needed?

1. A node can generate random value from its secret value
2. Other nodes can prove the random value is indeed using the secret value
   ⇒ attacker cannot change hash result rapidly by just changing value, or changing secret key
3. Other nodes cannot expect the hash result before the node announce the hash and proof
   ⇒ attacker is too late to make DoS attack, since the result is already propagated

Cryptographic Sortition with VRF

"I will get random value."

"I will roll dice on my coins based on the value."

procedure Sortition($sk, seed, \tau, role, w, W$):

$\langle hash, \pi \rangle \leftarrow \mathrm{VRF}_{sk}(seed || role)$

$p \leftarrow \frac{\tau}{W}$

$j \leftarrow 0$

while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p) \right)$ **do**

  j++

return $\langle hash, \pi, j \rangle$

Cryptographic Sortition with VRF

"Is the value is really random?"

"Let's see how many coins are selected."

procedure VerifySort($pk, hash, \pi, seed, \tau, role, w, W$):

if $\neg VerifyVRF_{pk}(hash, \pi, seed||role)$ **then return** 0;

$p \leftarrow \frac{\tau}{W}$

$j \leftarrow 0$

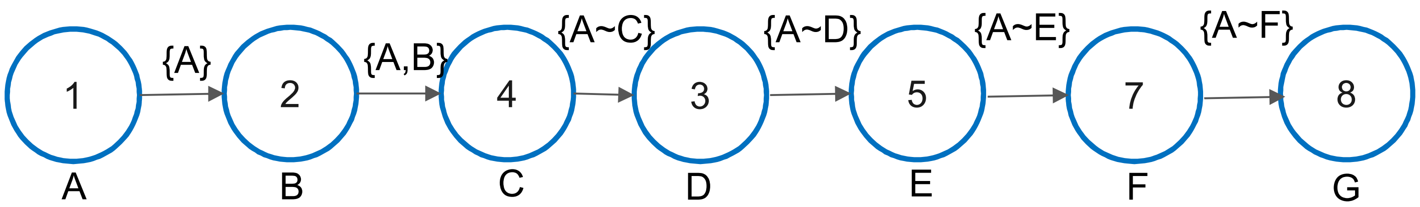
while $\frac{hash}{2^{hashlen}} \notin \left[\sum_{k=0}^{j} B(k; w, p), \sum_{k=0}^{j+1} B(k; w, p)\right)$ **do**

$\quad j{+}{+}$

return j

Block Proposal

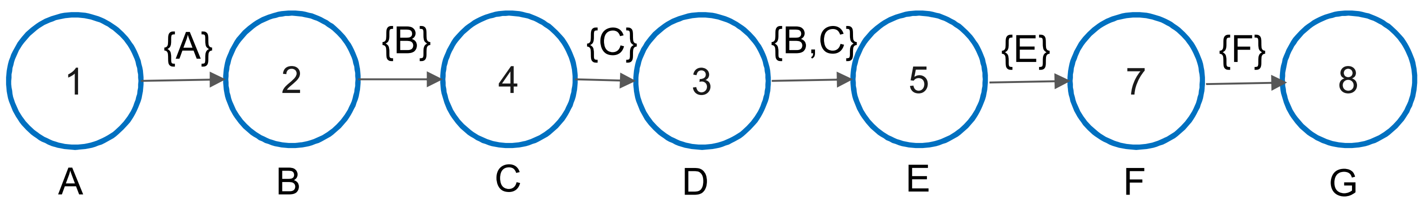
- Simply, We can think about all user rolling dice(Cryptographic Sortition) and say it to neighbor!



Problem: Too many messages (21 messages on example)!
How to solve this?

Block Priority Number

1. Make a priority number, send own block with the number
2. Only accept the blocks with higher number, update highest number
3. Wait some times for block propagation



Only 7 messages on example

Byzantine Agreement*

- Two phase agreement process for proposed blocks
- commitee group's member is selected by cryptographic sortition before Reduction Phase

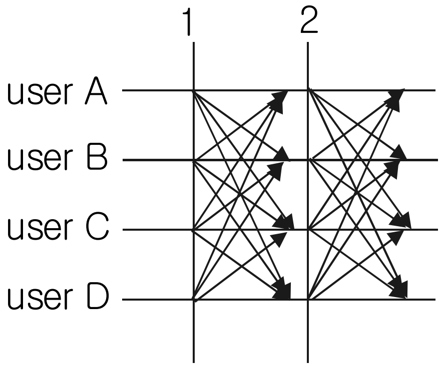
1. Reduction Phase
   –each committee member either decides a proposed block or decides an empty block
2. Binary Byzantine Agreement Phase
   –each committee member decides a block with the result from Reduction Phase

Reduction Phase

Two steps for reduction

1. Votes for hash of highest priority block

2. Votes again for the hash picked by more than T(2/3) of committee member
   –If there is no majority, decides to vote on empty block



Binary Byzantine Agreement Phase

Iterate three process until the user knows majority value
If maximum steps reached, recovery process follows

```
CommitteeVote(ctx, round, step, τ_STEP, r)
r ← CountVotes(ctx, round, step, T_STEP, τ_STEP, λ_STEP)
if r = TIMEOUT then
    r ← block_hash
else if r ≠ empty_hash then
    for step < s′ ≤ step+3 do
        CommitteeVote(ctx, round, s′, τ_STEP, r)
    if step = 1 then
        CommitteeVote(ctx, round, FINAL, τ_FINAL, r)
    return r
step++
```

```
CommitteeVote(ctx, round, step, τ_STEP, r)
r ← CountVotes(ctx, round, step, T_STEP, τ_STEP, λ_STEP)
if r = TIMEOUT then
    r ← empty_hash
else if r = empty_hash then
    for step < s′ ≤ step+3 do
        CommitteeVote(ctx, round, s′, τ_STEP, r)
    return r
step++
```

```
CommitteeVote(ctx, round, step, τ_STEP, r)
r ← CountVotes(ctx, round, step, T_STEP, τ_STEP, λ_STEP)
if r = TIMEOUT then
    if CommonCoin(ctx, round, step, τ_STEP) = 0 then
        r ← block_hash
    else
        r ← empty_hash
step++
```

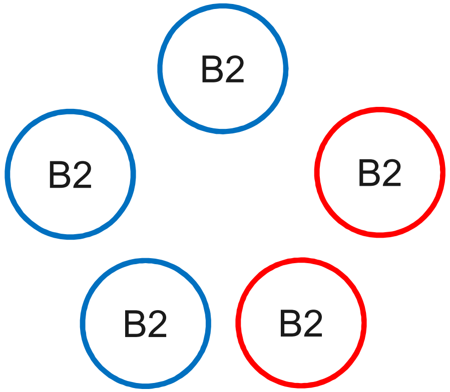
BinaryBA Phase 1

If there's majority value, return with the value.

CommitteeVote(ctx, $round$, $step$, τ_{STEP}, r)
$r \leftarrow$ CountVotes($ctx, round, step, T_{\text{STEP}}, \tau_{\text{STEP}}, \lambda_{\text{STEP}}$)
if r = *TIMEOUT* **then**
  $r \leftarrow block\_hash$
else if $r \neq empty\_hash$ **then**
  for $step < s' \leq step + 3$ **do**
    CommitteeVote(ctx, $round$, s', τ_{STEP}, r)
  if $step$ = 1 **then**
    CommitteeVote(ctx, $round$, FINAL, τ_{FINAL}, r)
  return r
$step$++

BinaryBA Phase 1 – case 2

Some nodes can timed out by adversary.
Finished node vote for them.



CommitteeVote(ctx, $round$, $step$, τ_{STEP}, r)
$r \leftarrow$ CountVotes($ctx, round, step, T_{\text{STEP}}, \tau_{\text{STEP}}, \lambda_{\text{STEP}}$)
if r = *TIMEOUT* **then**
  $r \leftarrow block\_hash$
else if $r \neq empty\_hash$ **then**
  for $step < s' \leq step + 3$ **do**
    CommitteeVote(ctx, $round$, s', τ_{STEP}, r)
  if $step = 1$ **then**
    CommitteeVote(ctx, $round$, FINAL, τ_{FINAL}, r)
  return r
$step$++

BinaryBA Phase 2

Consensus of Timed out users
Same thing happens on phase 1

CommitteeVote(ctx, $round$, $step$, τ_{STEP}, r)
$r \leftarrow$ CountVotes($ctx, round, step, T_{\text{STEP}}, \tau_{\text{STEP}}, \lambda_{\text{STEP}}$)
if r = $\textit{TIMEOUT}$ **then**
  $r \leftarrow empty\_hash$
else if $r = empty\_hash$ **then**
  for $step < s' \leq step + 3$ **do**
    CommitteeVote(ctx, $round$, s', τ_{STEP}, r)
  return r
$step$++

BinaryBA Phase 3

Phase 3 for mitigating adversary's attack (splitting committee network)

-adversary can split final decision if it knows each node's decision

-the attack is prevented eventually with $\frac{1}{2}$ probability

$\text{CommitteeVote}(ctx,\ round,\ step,\ \tau_{\text{STEP}},\ \text{r})$

$r \leftarrow \text{CountVotes}(ctx, round, step, T_{\text{STEP}}, \tau_{\text{STEP}}, \lambda_{\text{STEP}})$

$\textbf{if}\ r = \textit{TIMEOUT}\ \textbf{then}$

$\quad \textbf{if}\ \text{CommonCoin}(ctx, round, step, \tau_{STEP}) = 0\ \textbf{then}$

$\quad\quad r \leftarrow block\_hash$

$\quad \textbf{else}$

$\quad\quad r \leftarrow empty\_hash$

$step{+}{+}$

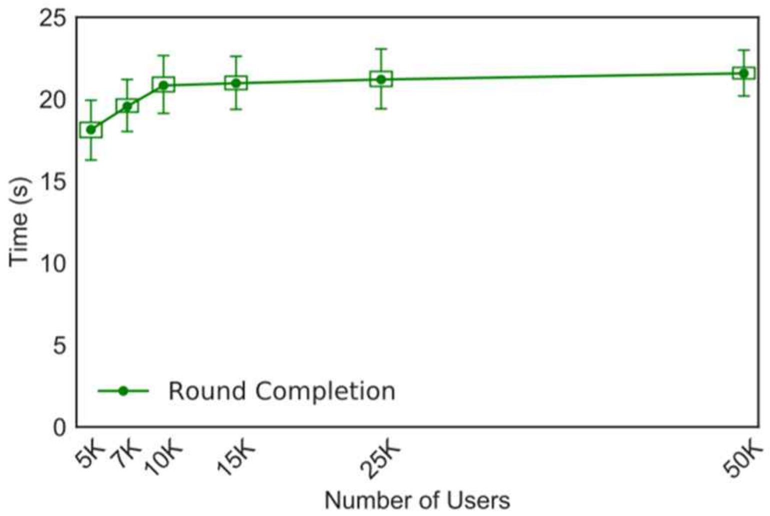
Evaluation Results

Key Evaluation Points:

1. What is the latency of Algorand, how does it scales over the number of the users?
2. What throughput can Algorand achieve?
3. How does Algorand perform when users misbehave?

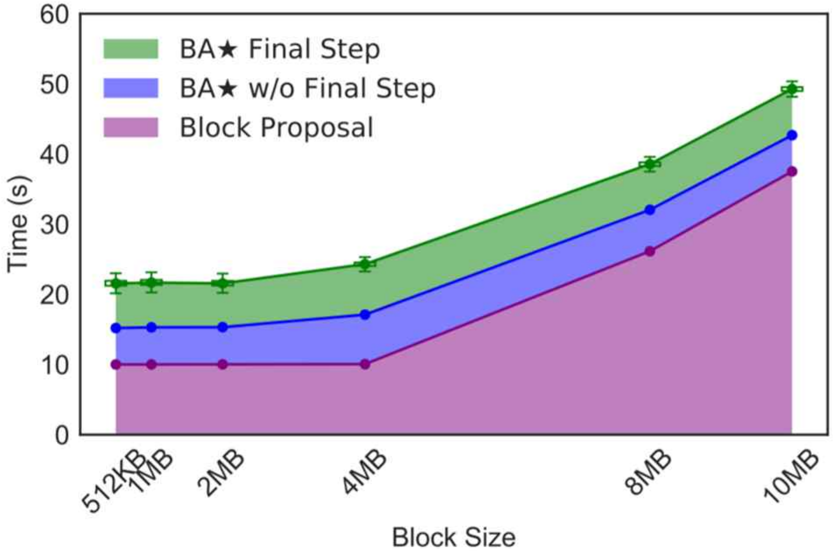
Latency Evaluation Results

One round of agreement takes less than 1 minute for 5K~50K users (100~1000VMs, 50 users per machine)



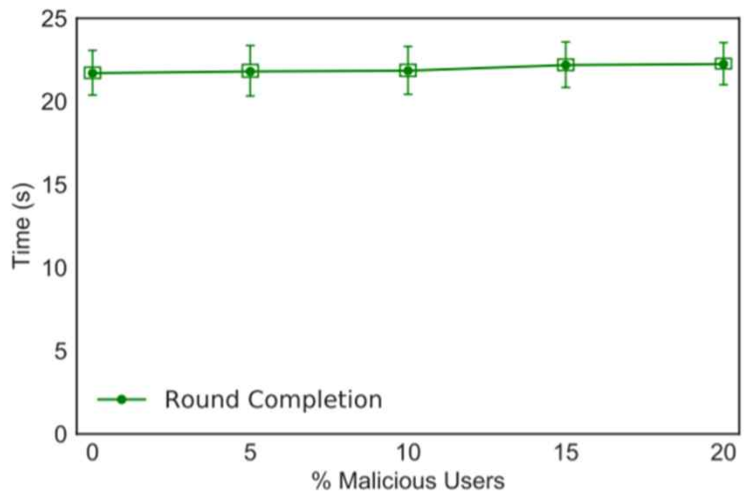
Throughput Evaluation Results

10MB block is added to the blockchain within 1 minute (with 1000VMs, 50 users per machine)



Latency over malicious users

Block generation latency does not change on malicious user changes



Limitation

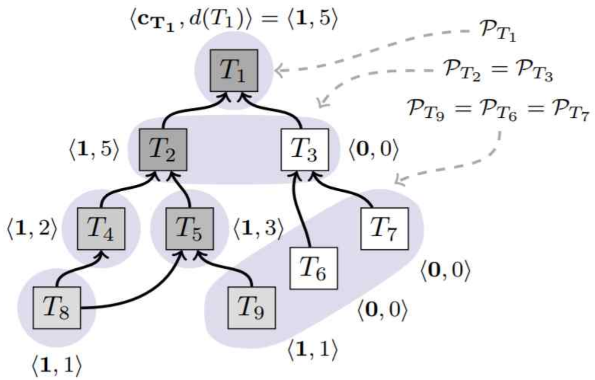
1. Lack of Incentive mechanism
   –It may not attract many users as other blockchain systems

2. Still high latency
   –1 minute latency still can make limited application usage

3. High bootstraping costs
   –users need to fetch large amount of data for node setup

Follow-up Paper

- Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies (Team Rocket, 2018)
  –Scalable to many users, by using verifiable random function
  –Modify chain design into DAG: improve transaction throughput



Questions?