

Avalanche

Snow Consensus Family
Cross-Chain Interoperability

Xunan Dai

Snow Consensus Family (2018)

A new family of consensus protocols, inspired by gossip algorithms

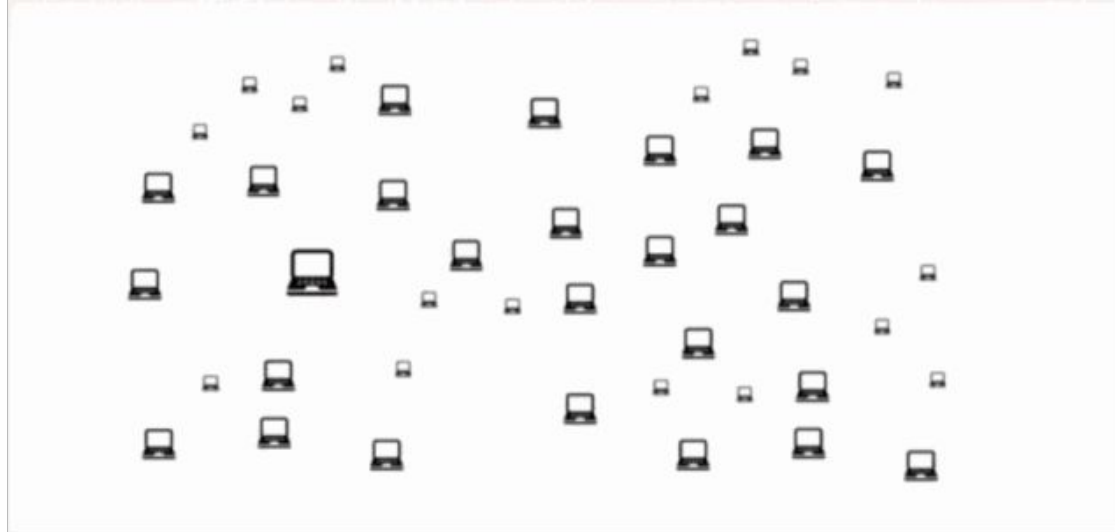
Features: Best of all worlds

- Quick finality, low latency (~2 sec)
- High throughput (1000-10000 TPS)
- Scales (10k to 10m nodes)
- Robust, no need for precise membership (Permissionless)
- Quiescent, green, sustainable (PoS)

Operation

Goal: All node will be **colored** identically with high probability by setting a pair of appropriate parameters.

Like: 50/50 -> 51/49 -> 55/45 -> Ideal state “100/0”

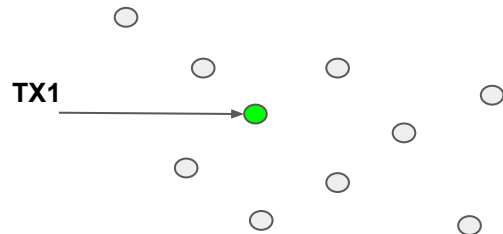


Slush

Every node is honest, **tx** can not be changed

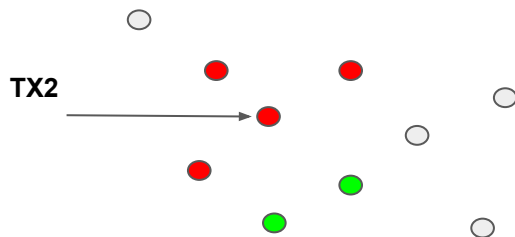
Sample **k**: 5 **α** : 50% **m**: 10

1.



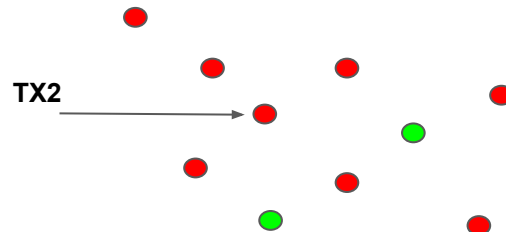
Init color: TX1

2.



3 > 2: TX1 -> TX2

3.



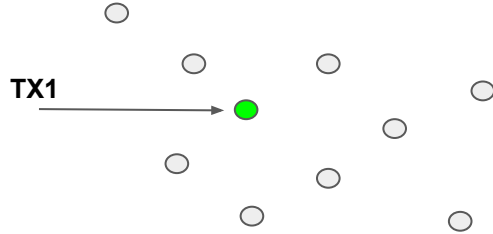
After **m** round

Accept **TX2**

Keep liveness, not safety

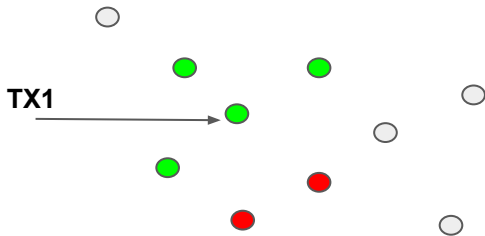
Snowflake

1.



Init color = TX1, cnt = 0

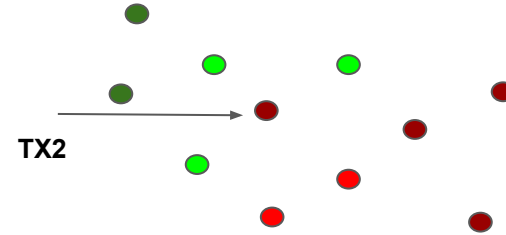
2.



$3 > 2$: keep TX1, last color = TX1, cnt ++ = 1

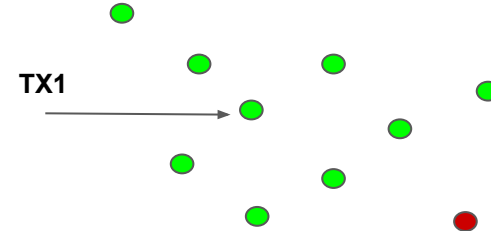
Sample k : 5 α : 50% β : 30

3.



$3 > 2$: TX1 -> TX2, last color = TX2, cnt = 0

4.

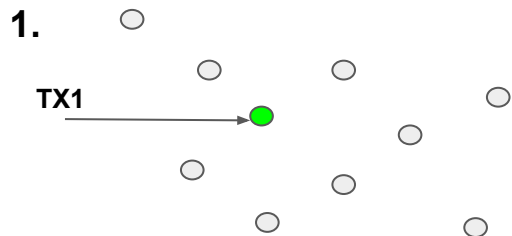


cnt > β , Accept TX1

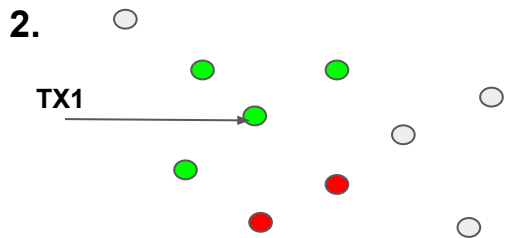
Keep safety, not liveness

Snowball (more efficient)

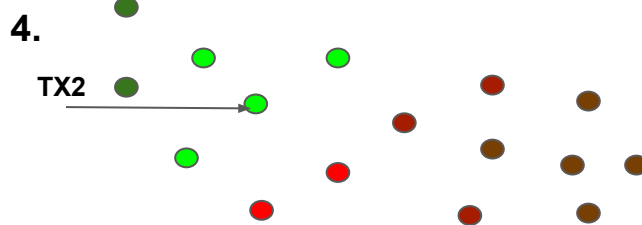
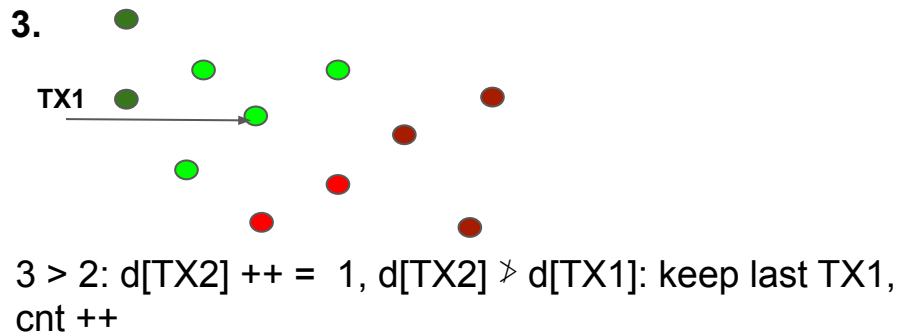
Sample $k: 5$ $\alpha: 50\%$ $\beta: 20$



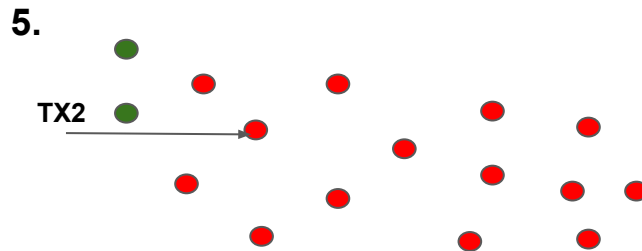
Init = last color = TX1, cnt = 0, d[TX1] = 0, d[TX2] = 0



$3 > 2$: d[TX1] ++ = 1, d[TX1] > d[TX2]: keep last TX1, cnt ++



$5 > 0$: d[TX2] ++ = 2, d[TX2] > d[TX1]: TX1 -> TX2, last color = TX2, cnt = 0

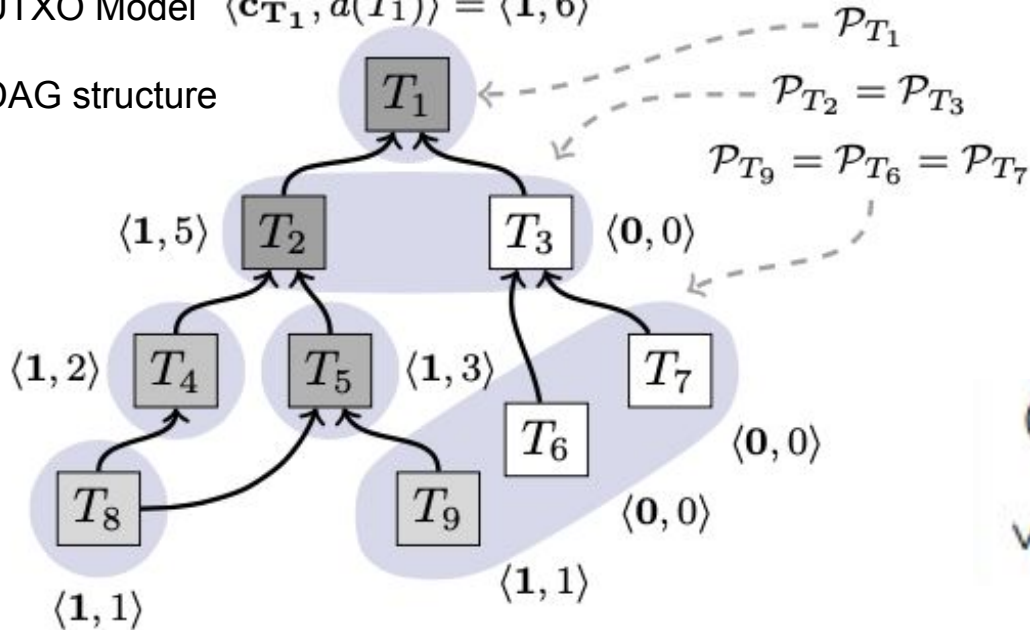


cnt > β , Accept TX2

Avalanche (use Snowball in blockchain)

UTXO Model $\langle \mathbf{c}_{T_1}, d(T_1) \rangle = \langle 1, 6 \rangle$

DAG structure



If no conflicting tx && its entire ancestry are
preferred, **chit** = 1

else If collect **k** positive responses > **α**, **chit** = 1

$d(T) = \sum(\text{chit})$ (sum of chits in its progeny)

If $d(T) > d(\text{Preferred})$:

 Pref = T

 (update preferred tx of each conflict set)

$((\forall T' \in \mathcal{T}, T' \leftarrow T : \text{ISACCEPTED}(T'))$
 $\wedge |\mathcal{P}_T| = 1 \wedge \mathcal{P}_T.\text{cnt} > \beta_1) \quad // \text{ safe early}$
 $\vee (\mathcal{P}_T.\text{cnt} > \beta_2) \quad // \text{ consecutive counter}$

PoS in Avalanche

Proof-of-Stake is the method by which the AVA network prevents ***sybil attacks***.

For Avalanche, any node can become validator, any validator of the subnetwork can propose ***Block / Vertex*** and send it to other validators to vote.

The more AVA staked, the more likely the validator to be sampled.

Every single participant in the staking protocol is rewarded equitably and proportionally based on ***proof-of-uptime*** and ***proof-of-responsiveness***

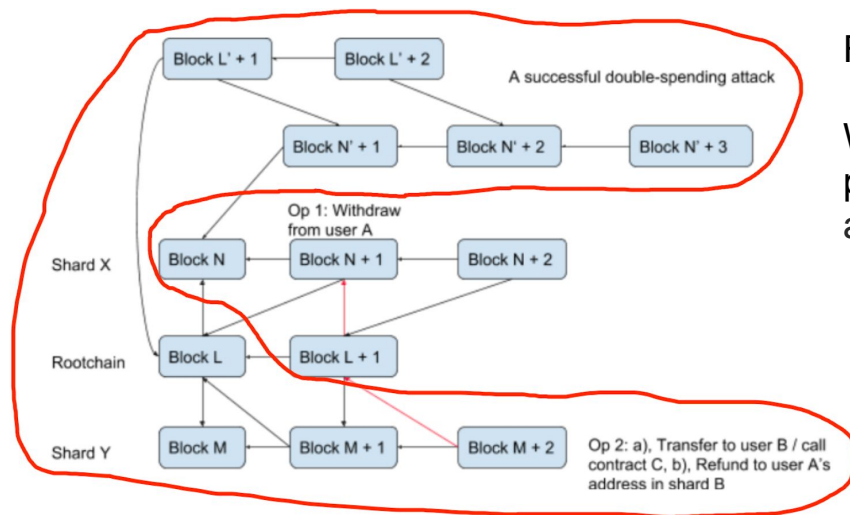
Block / Vertex

Vertex is to DAG blockchain, using **Avalanche** Protocol as mentioned before.

Block is to linear blockchain, using **Snowman** Protocol, not in this scope.

Basically they're a collection of **txs**.

Avalanche vs Quarkchain

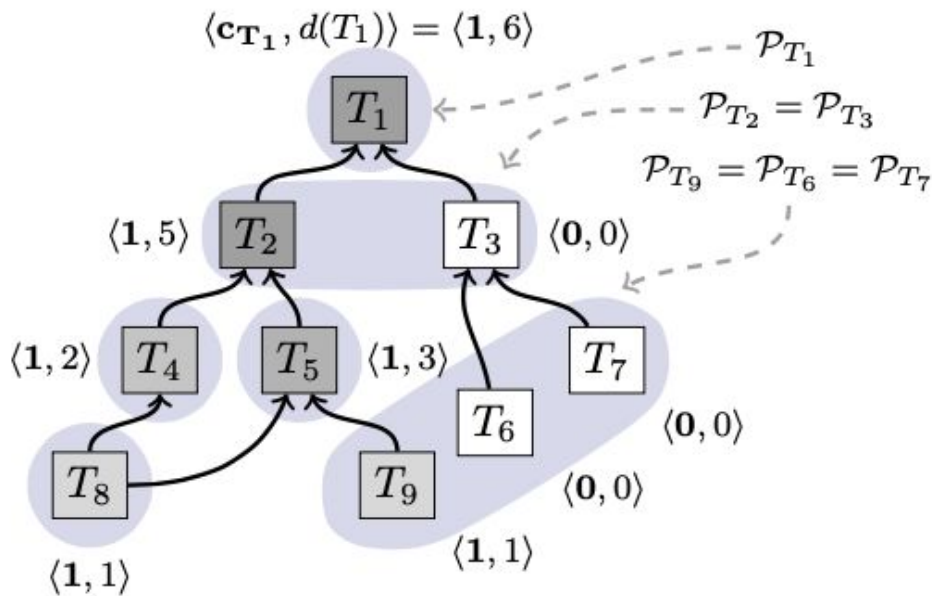


A Successful Double-Spending Attack

For Quarkchain,

We also use DAG structure, each minor block has two pointers: one links to the previous minor block in the shard and another one links to the latest root block.

1. Given two DAGs, we could easily determine which DAG should be chosen by using root-chain first fork rule.
2. We could optimize the system performance by using multi-threading or clustering



For Avalanche

By only providing two sub-graphs cannot decide which one is correct without other parameters.

Therefore, Avalanche needs **timestamp** to calculate **chit**, **confidence** ... to finally decide which one is correct.

That is the major difference between with us.

Subnetwork

A **Subnet**, or Subnetwork, is a dynamic set of **validators** working together to achieve consensus on the state of a set of blockchains. Each blockchain is validated by exactly one Subnet. A Subnet can validate arbitrarily many blockchains. A node may be a member of arbitrarily many Subnets.

Subnetwork

Default Subnet

6 blockchains validated by this subnet



Blockchains (6)

Validators (277)

Pending Validators (2)

Control Keys (0)

Name	Virtual Machine ID
------	--------------------

 P-Chain	
 X-Chain	juYyQTxGMJLuGwa55kdP2p;
 C-Chain	mgj786NP7uDwBCcq6YwThz
 Simple DAG Payments	sqjdyTKUSrQs1YmKDTUbdUh
 Simple Chain Payments	sqjchUjzDqDfBPGjIQq2tXW1U
 Simple Timestamp Server	tGas3T58KzdlHhBDMnH2Tvr

Subnetwork

21Zg8zBbWtFg5G4fz49DZzsBoSf1UdRqXGxxCwrhZJSw1Q5aqr


1 blockchain validated by this subnet

Blockchains (1)

Validators (1)

Pending Validators (0)

Control Keys (1)

Name	Virtual Machine ID
 My Denali Private Subnet AVM	juYyQTxGMJLuGwa55kdP2p2zSUyS5Q5...

Cross-chain (Default Subnet)

X-chain

call: **exportAVA**

```
curl -X POST --data '{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "avm.exportAVA",
  "params": {
    "to": "MvmGVpprbKQ9EBb7cTHaGVNbNjNMvCozu",
    "amount": 10000,
    "username": "freshmanD",
    "password": "dai123456DD."
  }
}' -H 'content-type:application/json;'
127.0.0.1:9650/ext/bc/X
```

P-chain

call: **importAVA**

```
curl -X POST --data '{
  "jsonrpc": "2.0",
  "method": "platform.importAVA",
  "params": {
    "username": "freshmanD",
    "password": "dai123456DD.",
    "to": "MvmGVpprbKQ9EBb7cTHaGVNbNjNMvCozu",
    "payerNonce": 1
  },
  "id": 1
}' -H 'content-type:application/json;'
127.0.0.1:9650/ext/bc/P
```

call: **issueTx**

```
curl -X POST --data '{
  "jsonrpc": "2.0",
  "method": "platform.issueTx",
  "params": {
    "tx": ""
  },
  "id": 1
}' -H 'content-type:application/json;'
127.0.0.1:9650/ext/bc/P
```

Workflow

Each user of a node has a shared memory, including each blockchain's shared memory

If transfer from X -> P:

1. Check user's balance on X-chain > sending amount
2. If valid, issue a *tx* on X-chain -> Accept / Reject
3. P-chain loads *tx* info from X-chain's shared memory
4. Add transfer amount on P-chain account, issue a *tx* on P-chain -> Accept / Reject

More like a **Relay** method, access the two chains' shared memory and commit on each of them

Open Talk:

If X-chain reject, balance won't change

If X-chain accept, P-chain reject, rebuild X-chain shared memory by deleting the rejected **tx**.

If X-chain accept, P-chain unknown, it's like a malicious behaviour, the money will be costed as punishment.