TREEBLOCK Version 1.3

Author: Anh Le , Advisor: Dr. Rex Yeap

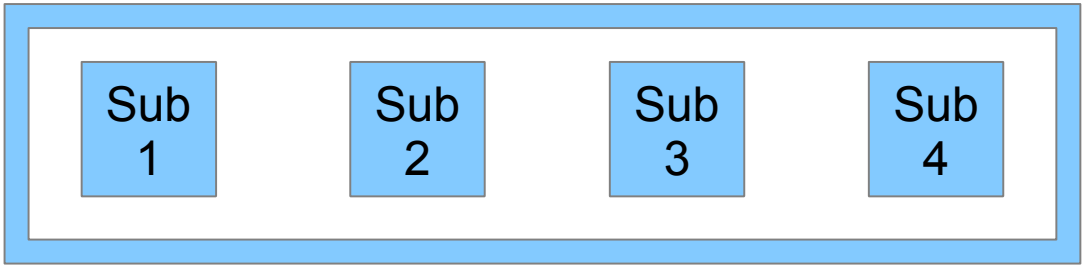
Scalable – High Performance – More Secure

A Public Blockchain Applies for General Purpose

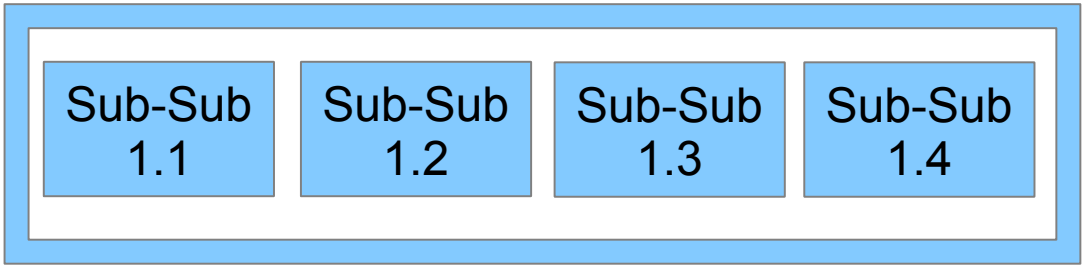
Table of Content

1. Hierarchy Sharding Concept

Ex: 1 Shard with 4 Sub Shards



1 Sub Shard with 4 Sub-Sub Shards



1.1 Hierarchy Sharding's Advantage

Fast Synchronization

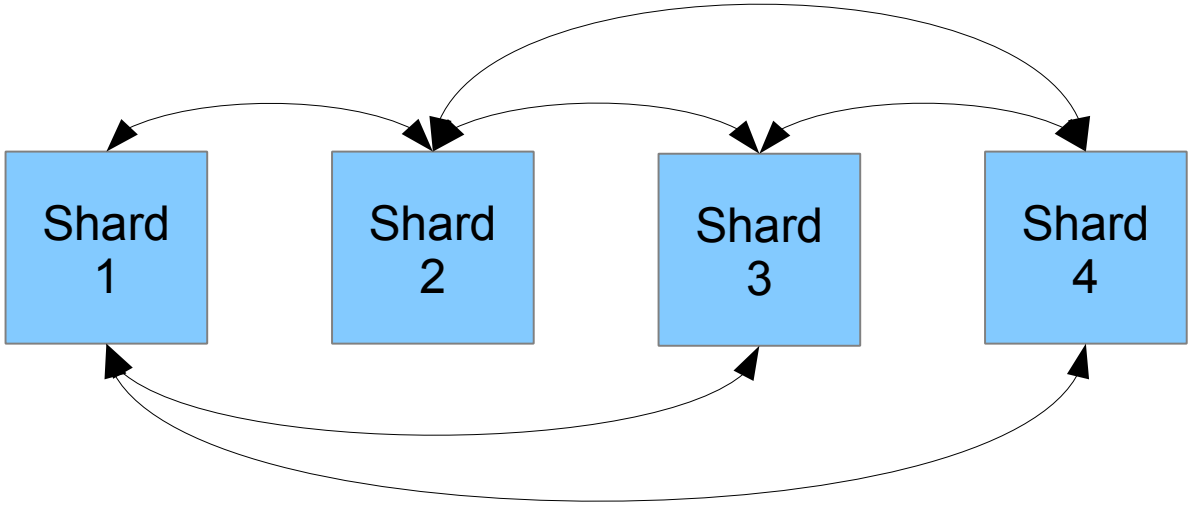
Cross Check Faster

Search Faster

Less Data on per Shard

Can Be Dividing More Shards

1.2 Cross Check & State Synchronization among Shards



Ex: with 4 Shards, we need to check

$$C_n^k = \frac{n!}{k!(n-k)!}$$ = 4! : (2!2!) = 6 pairs

k=2, n=4

1.3 Cross Check & State Synchronization with Horizontal Sharding

Ex: with 100 Shards, we need to check
100! : (2!98!) = 4,950 pairs

What about 1,000 Shards or more like other existing Blockchains?
1,000! : (2!998!) = 499,500 pairs

1.4 Cross Check & State Synchronization with Hierarchy Sharding

Ex: If we divide 100 Shards into 10 Main Shards and each Main Shard has 10 Sub Shards, so we have

$$[(10!:(2!8!))*10] + [10!:(2!8!)] = 495 \text{ pairs}$$

Can we divide more levels? Yes!

1.5 Cross Check & State Synchronization With Hierarchy Sharding

Ex: And, If we divide 100 Shards into 5 Main Shards and each Main Shard has 5 Sub Shards, each Sub Shard has 4 Sub-Sub Shards

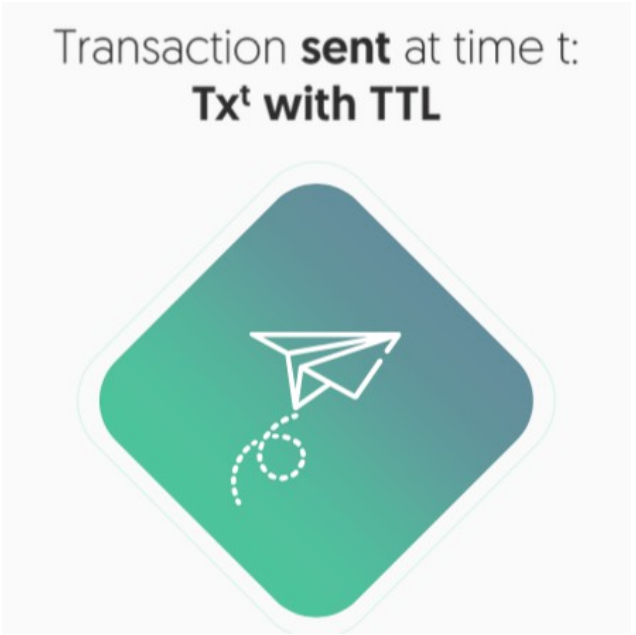
[(4!:(2!2!))*5*5] + [5!:(2!3!)] = 170 pairs

Much more effective!

How do we divide shards with Hierarchy Architecture?

2. Time Sharding Concept

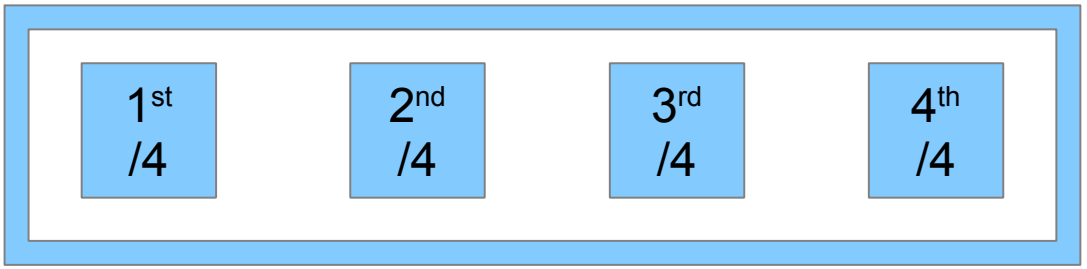
We use Tx's timestamp



2.1 Time Sharding Concept

Ex: Case 1

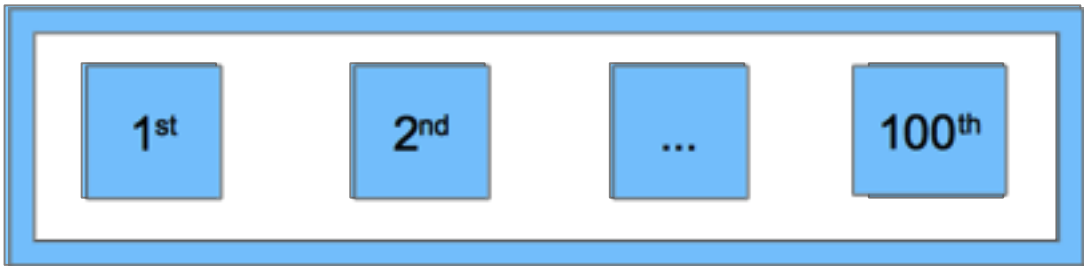
We divide 1 Second into 4 Segments
1 Segment = 250,000 mili-sec



2.2 Time Sharding Concept

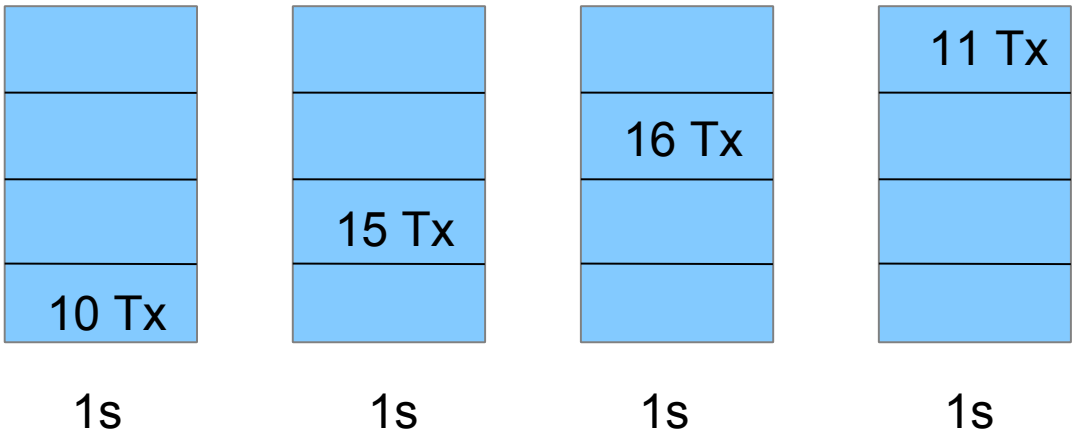
Ex: Case 2

We divide 1 Second into 100 Segments
1 Segment = 10,000 mili-sec

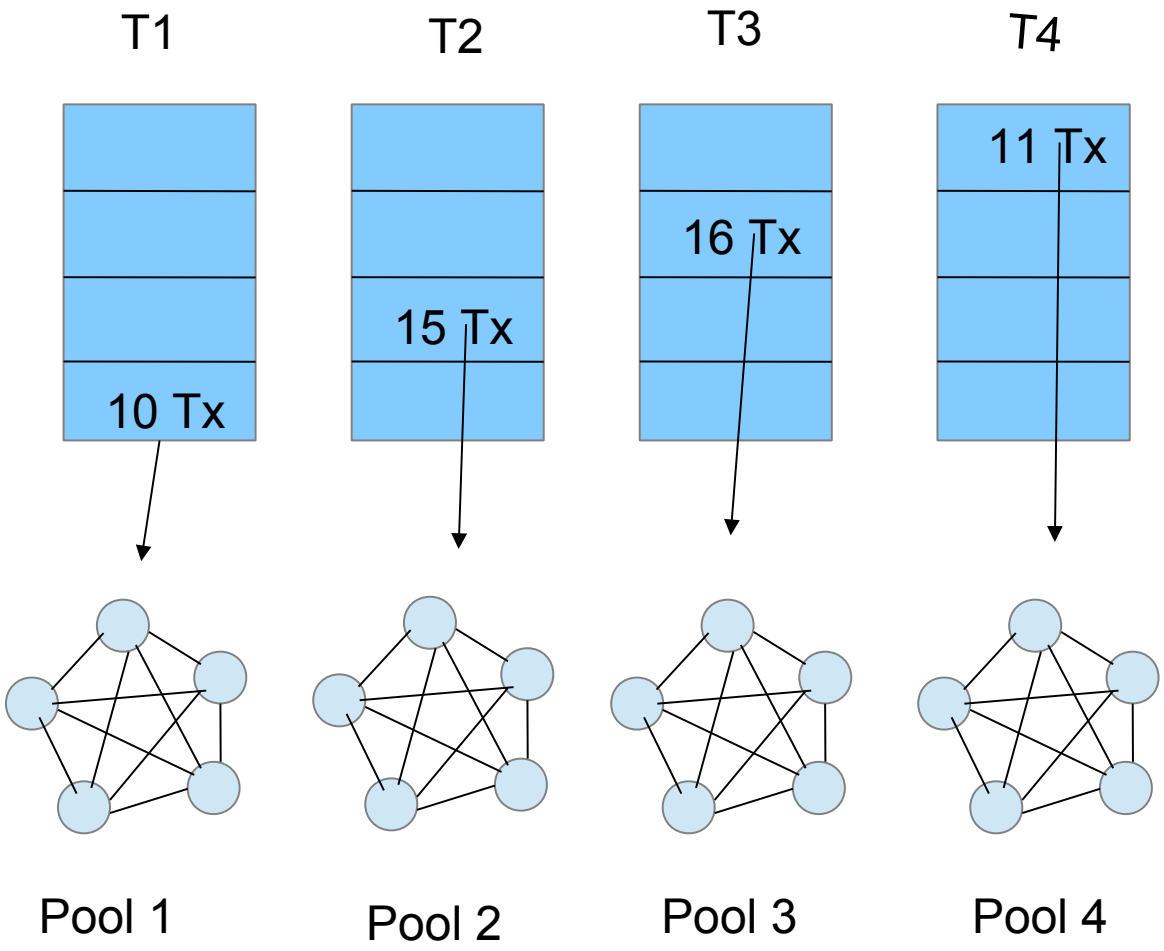


2.3 Time Sharding Concept

Ex: Case 1 with 52 Tx in 1 Sec
1 Time Segment = ¼ Sec

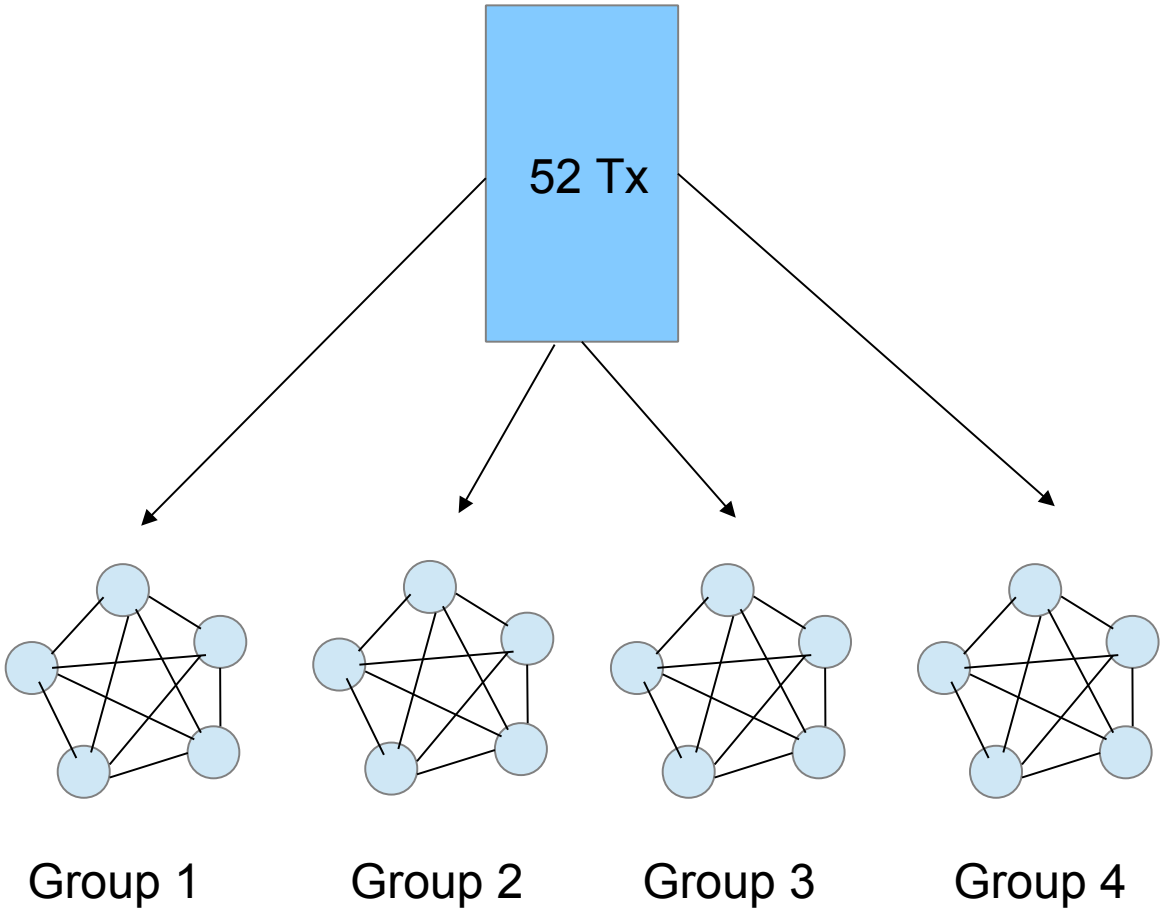


2.4 Processing Tx in 1 Second with Time Sharding



1 Pool = Groups of Nodes

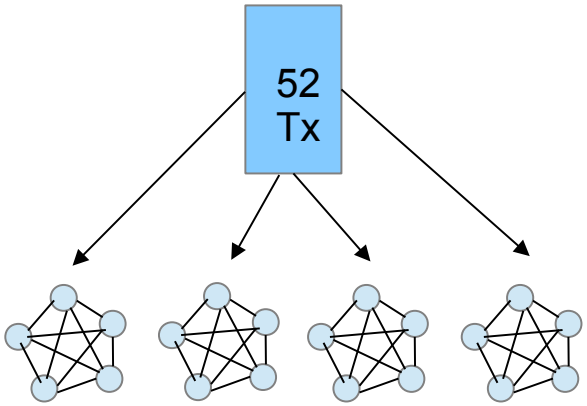
2.5 Processing Tx in 1 Second Without Time Sharding



Current Sharding Solutions

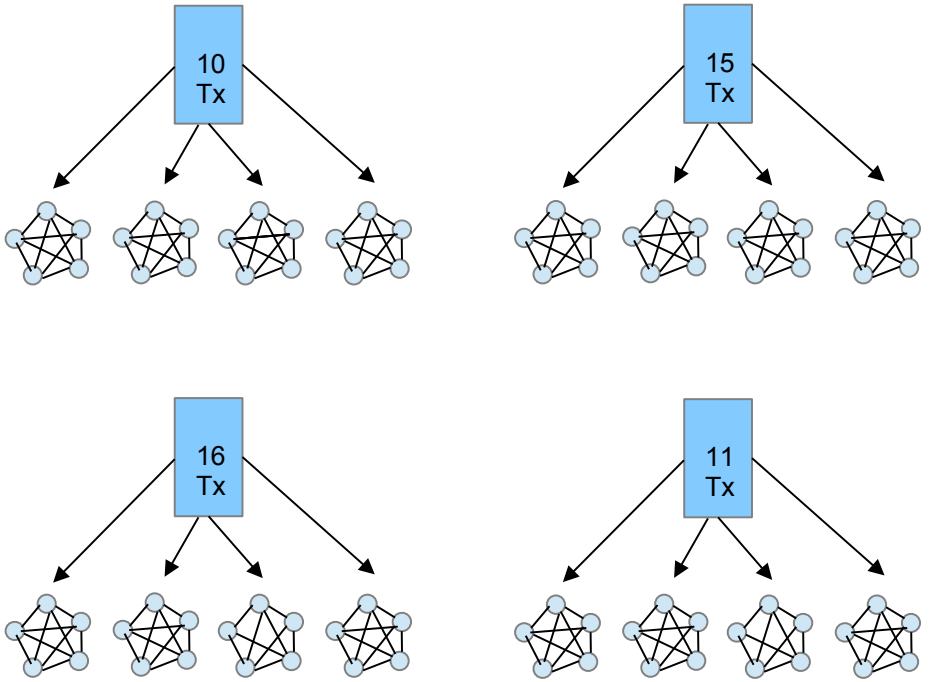
2.6 Normal Sharding vs Time Sharding

Normal



VS

Time Sharding



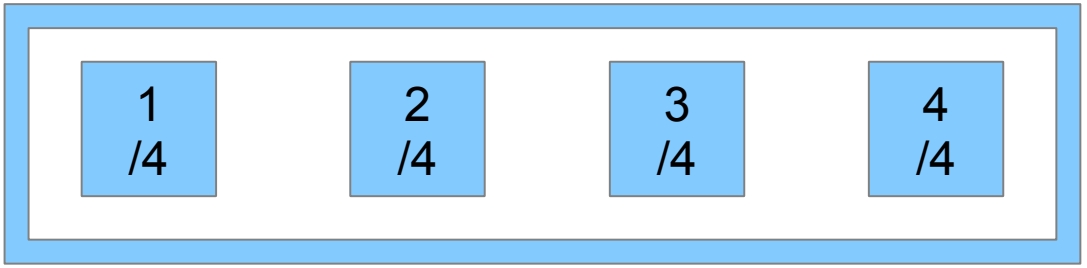
3. Hierarchy Sharding needs Time Sharding

FOR

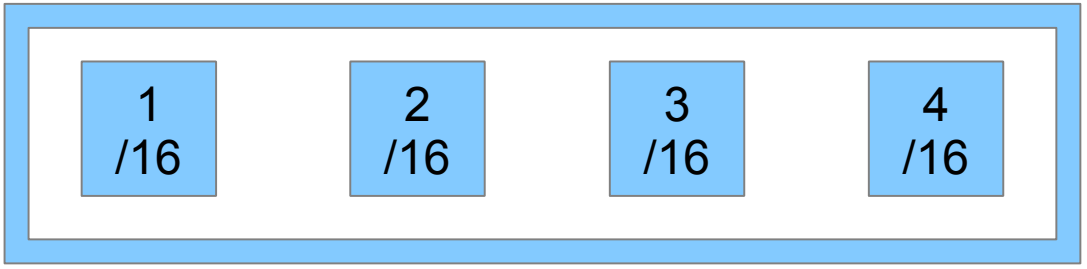
- Rules to Divide Shards
- Rules for Input
- Rules for Output
- Rules for Search
- Rules for Cross-Check
- Rules for Synchronization

3. Hierarchy Sharding needs Time Sharding

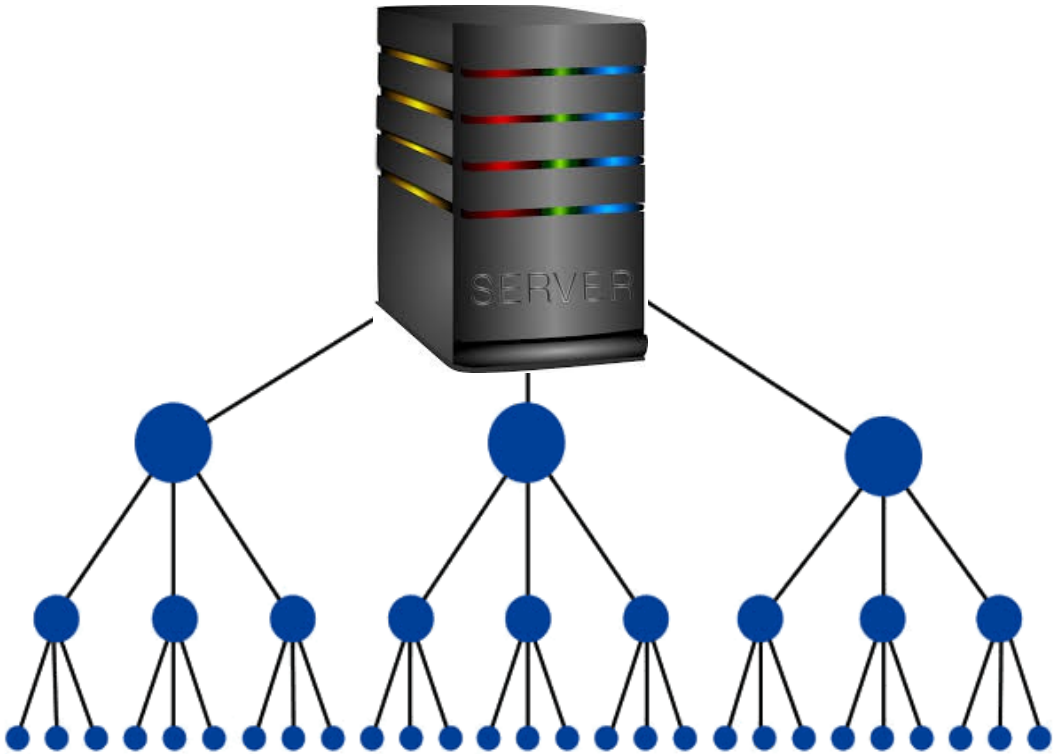
Ex: 1 Main Shard has
4 Sub Shards for 4 Time Segments



Ex: Each Sub Shard has 4 other Sub-Sub Shards for other 4 Time Segments



4. Family's Node



- Family's Representative
- Families Maintain Blockchain
- Created by Trusted Network
- Simplify Voting Process
- Harder for 51% Attack
- Producing Block
- Earn Tx's Fees
- Divide Benefit to Family Members
- Maintain a Family Fund

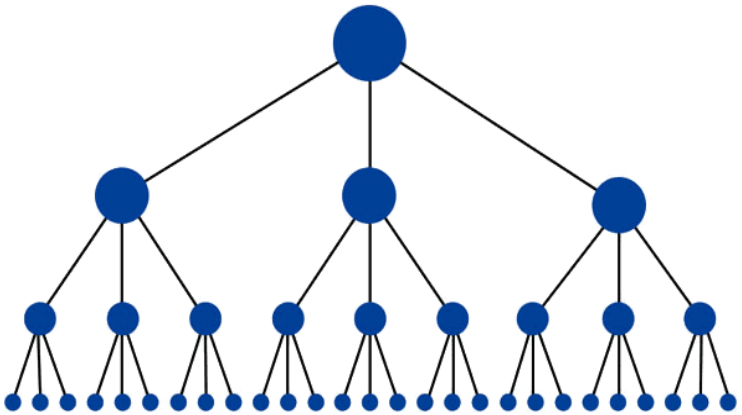
5. Block Producers Mgmt App

Easy Set Up a Block Producer (Family's Node) for Family (Non-Tech Person)

Easy to Manage Family Fund to Operate Block Producer

Legacy to Elevate Education, Finance, .etc for Next Generations.

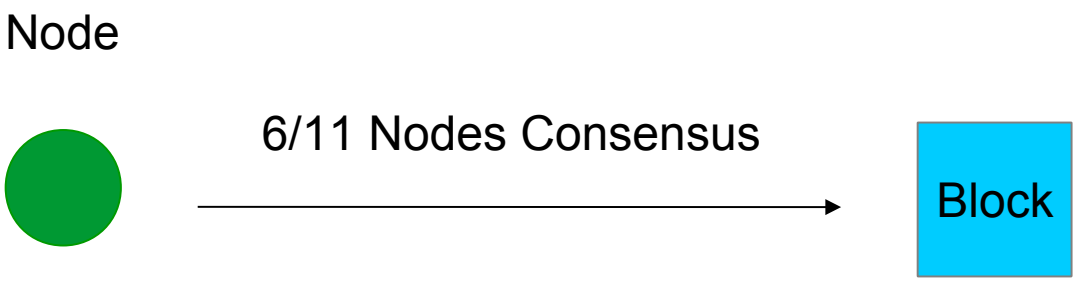
Transparent among Family Members



6. Preferred Consensus Algorithm “DPoS”

Definition: One Delegated Node produces Block at a time slot

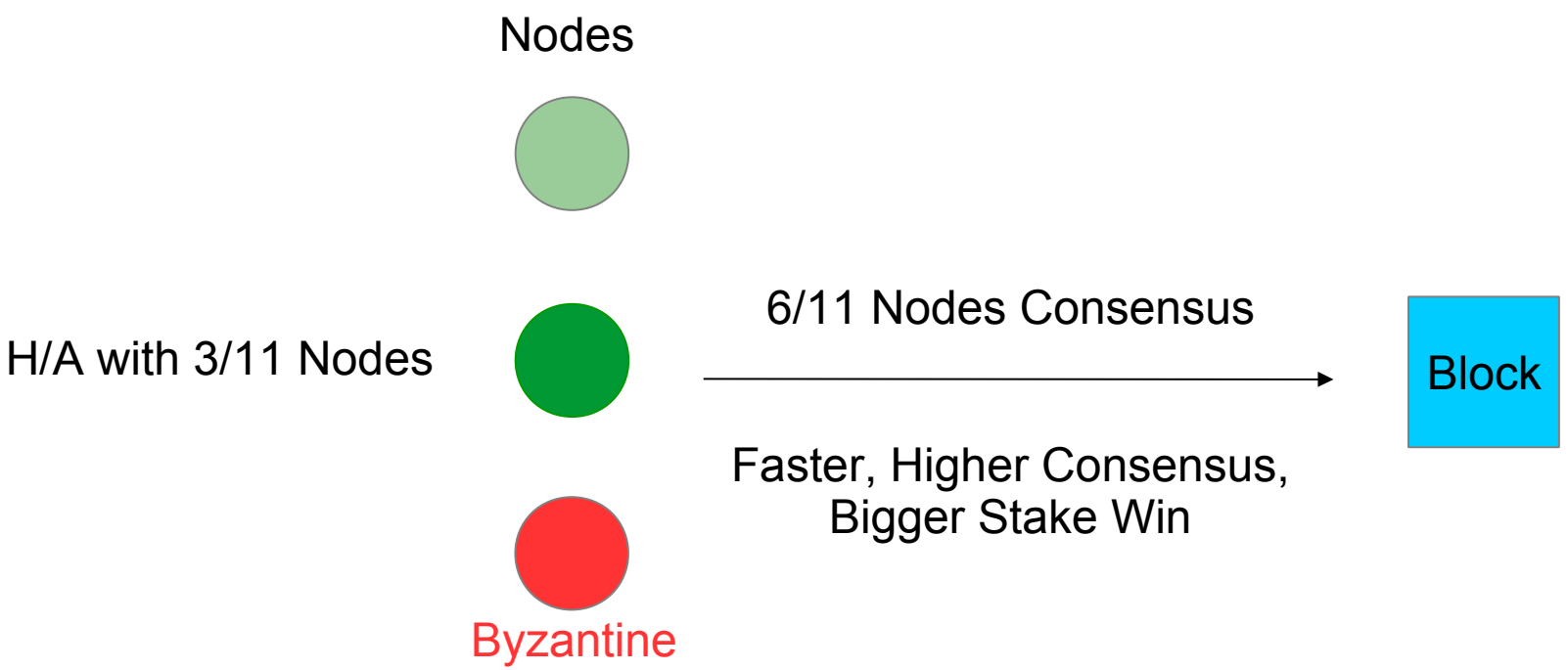
Ex: DPoS = 11



6. Introduce & Suggest m-DPoS = multiple DPoS

Definition: Many Delegated Nodes compete to produce Block at a same time slot

Ex: m-DPoS = 3-11



7. Producing Block Paradigm With Time Sharding

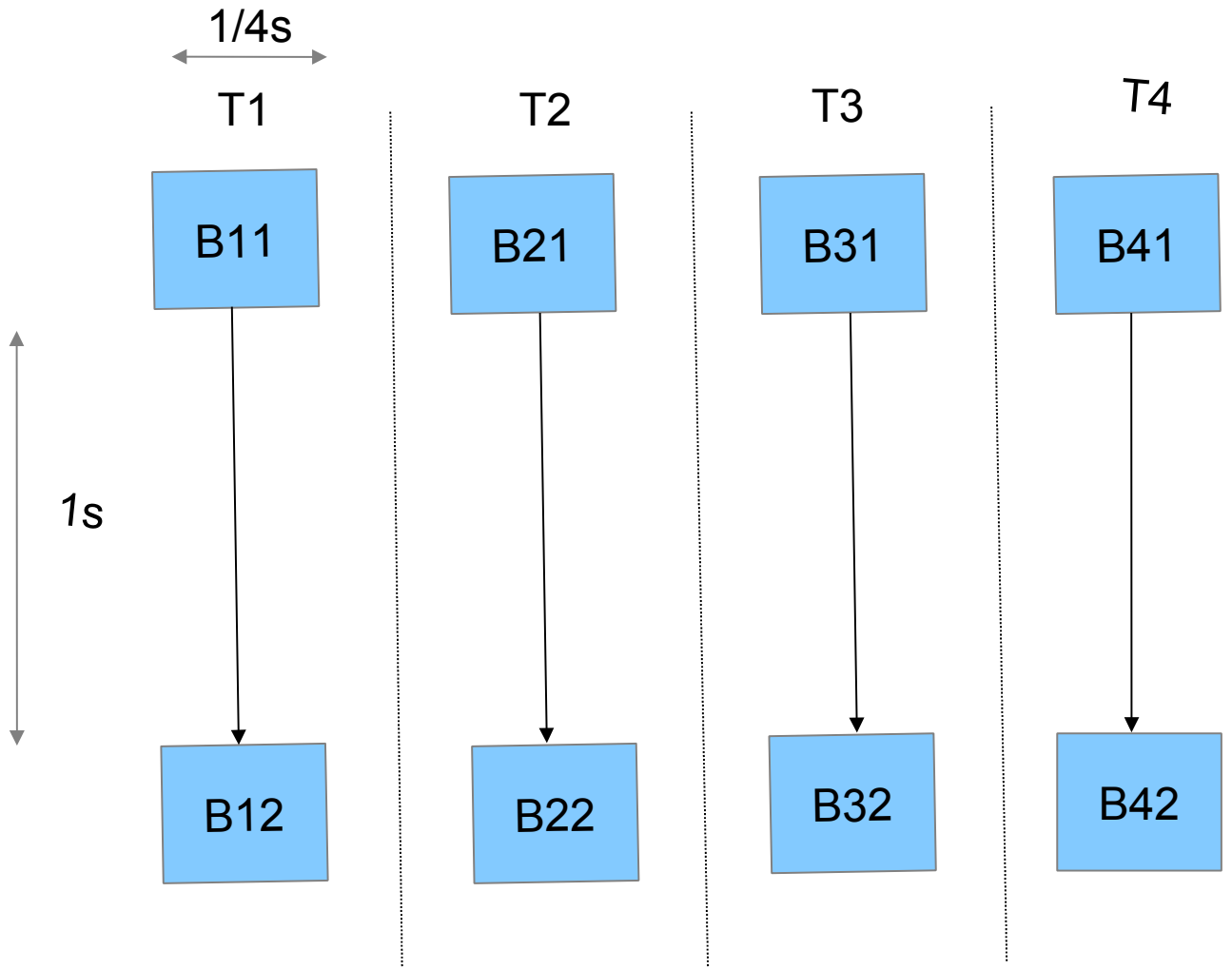
2 Ways to Produce Block with This Architecture

PARALELLY vs SEQUENTIALLY

7.1 Parallelly Producing Block With Time Sharding

Example: BlockTime is 1 Second

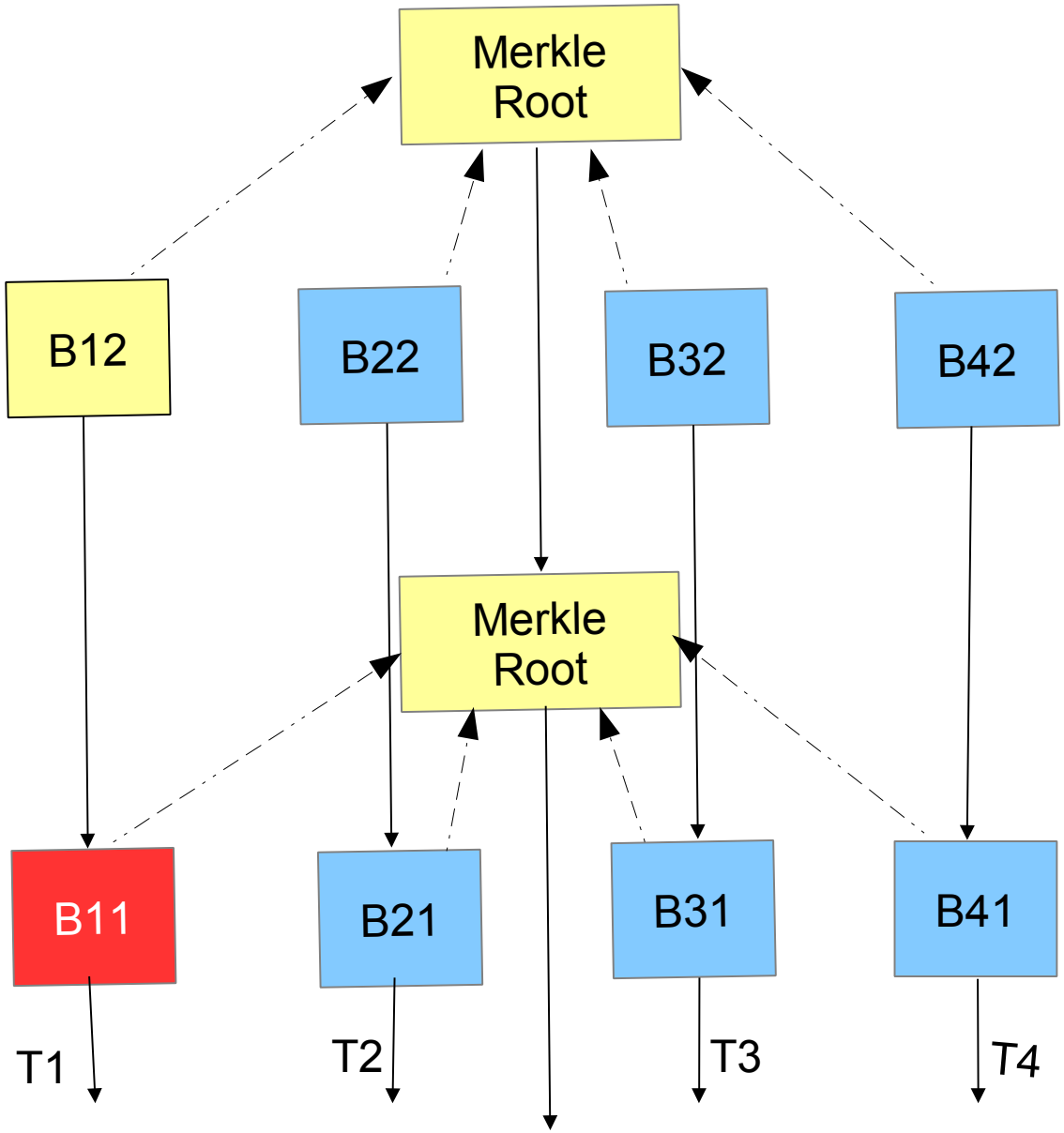
7.2 Parallelly Abstract Time-Chains



7.3 Producing Merkle Root with m-DPoS cross Time-Chains

“m” Delegated Nodes of Time-Chains will cross check
&
compete to produce Merkle Root of previous Blocks

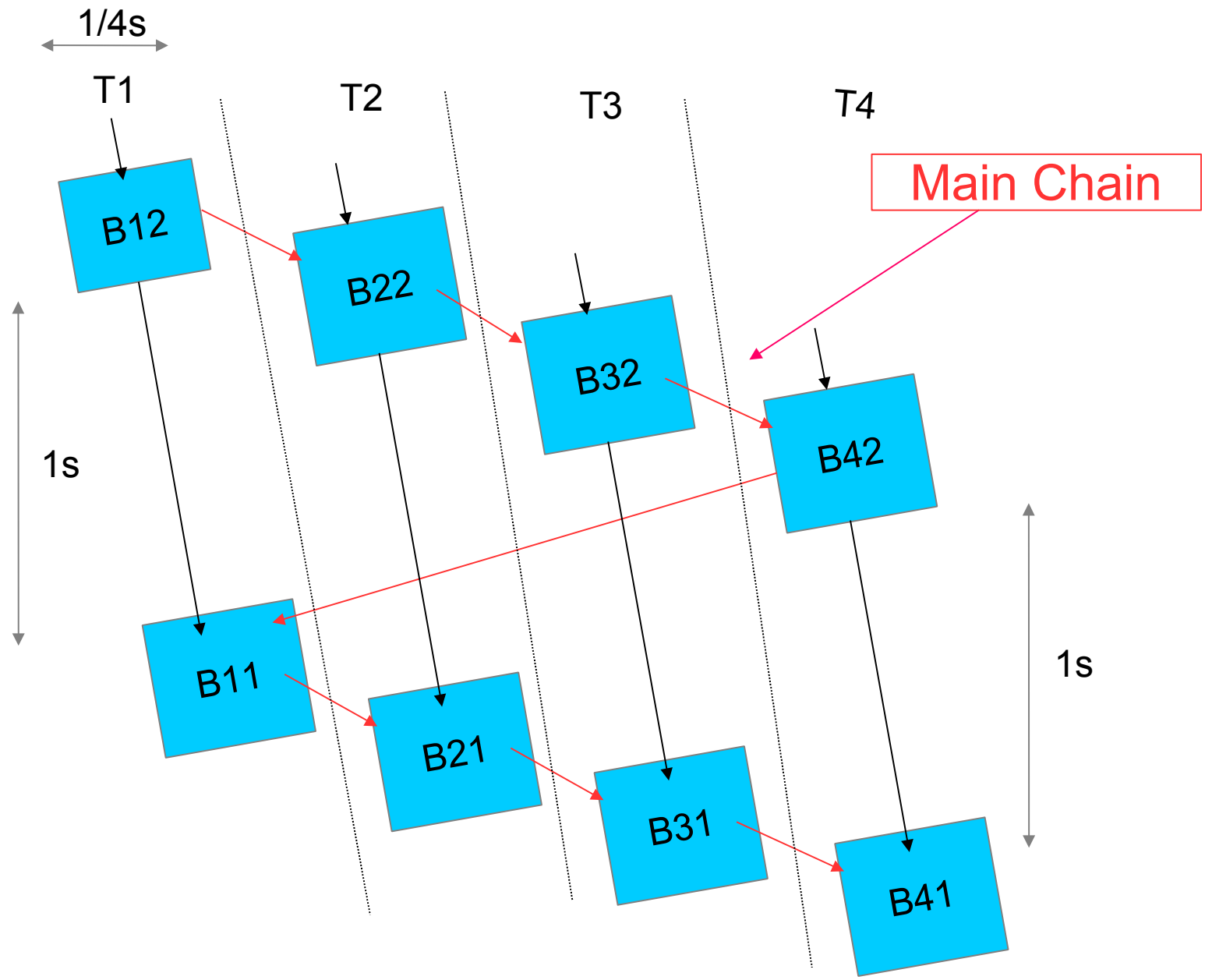
7.4 Block Attack on Parallel Time-Chains



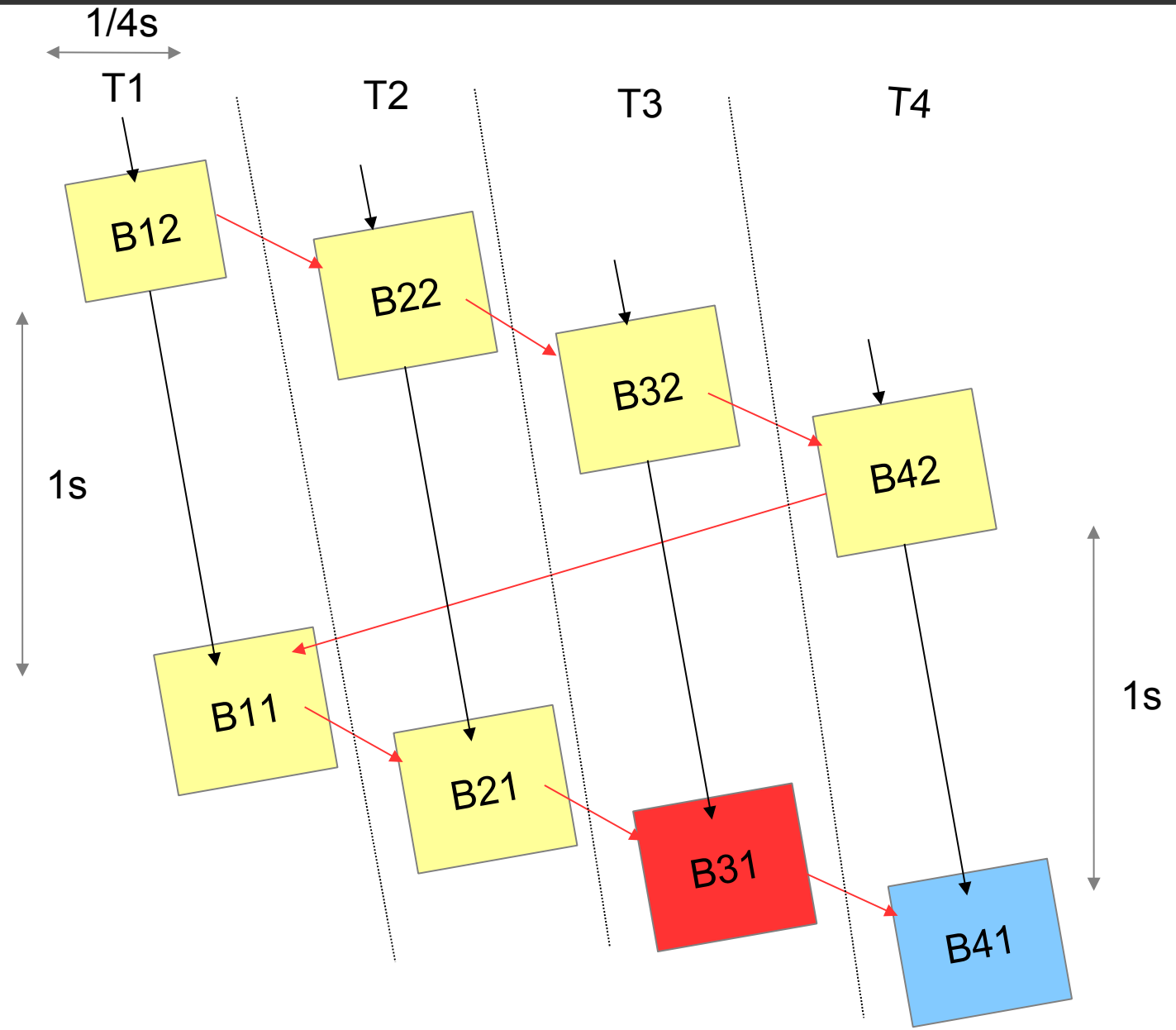
7.5 Sequentially Producing Block With Time Sharding

Example: Every 1 Second per Time-chain
Or
Every Time Step based on Time Segment overall

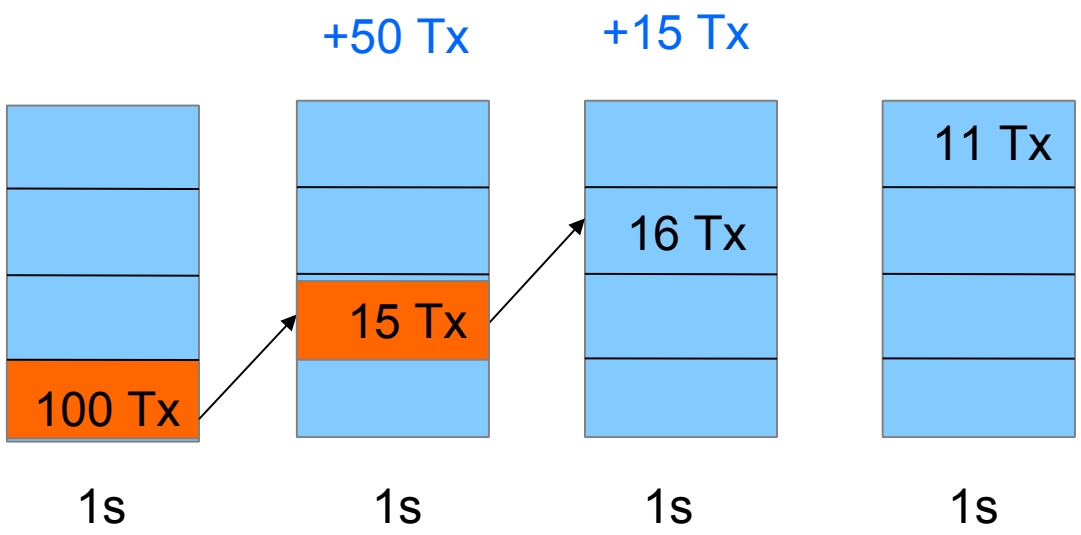
7.6 Sequential Abstract Time-Chains



7.7 Block Attack on Sequential Time-Chains



7.8 Case: Overload & Load Balance
Ex: Maximum 50 Tps per Time-Chain



Shift and Process overloaded Tx on other Time-Chains after Cross-Check

7.9 Parallelly vs Sequentially

- Heavy traffic when parallelly cross-check between Time-Chains per cycle
- Better if extend block producing time, 2 s or more
- Better with high internet speed and Node power
- Scale more with smaller Time Segments

- Sacrifice some times to check cross Time-Chains per cycle
- Able to work with shorter cycle for block producing
- Better with high internet speed and Node power
- Scale more with smaller Time Segments

8. Transaction Model

COMBINING

UTXO	ACCOUNT / BALANCE
Good for processing Tx parallelly & cross Shards	Good for State Checking before processing Tx

Digital Signature Algorithm

EC-Schnorr

Fast
&
Support Multiple Signatures

What to Wait for in Next Versions

Incentive Tokenomy

Virtual Machine

Smart Contract

Open Discussion

FOUNDER & CTO

Anh Le

anh@treeblock.io