ECE 595: Foundations of Blockchain Systems **November 30, 2020**

Lecture 17

Lecturer: Sreeram Kannan *Scribe: Yufei Liu*

Outline: This lecture firstly introduces the blockchain trilemma, and one idea to solve the scaling problem is sharding. There are Multi-Consensus architecture, which is identity-based, and Uniconsensus architecture, which is identity-free. Then we use Dynamic Self Allocation (DSA) to defend the liveness attack. Finally, introduce the Free2Shard DSA which has pretty good performance on both scaling and security.

17.1 Blockchain Trilemma

There's a key problem of blockchain, called Blockchain Trilemma. The blockchain trilemma states there are three important axes when developing a blockchain system: Security, Decentralization and Scalability. An ideal blockchain system should have all these three properties, which means it should be secure, decentralized and scalable.

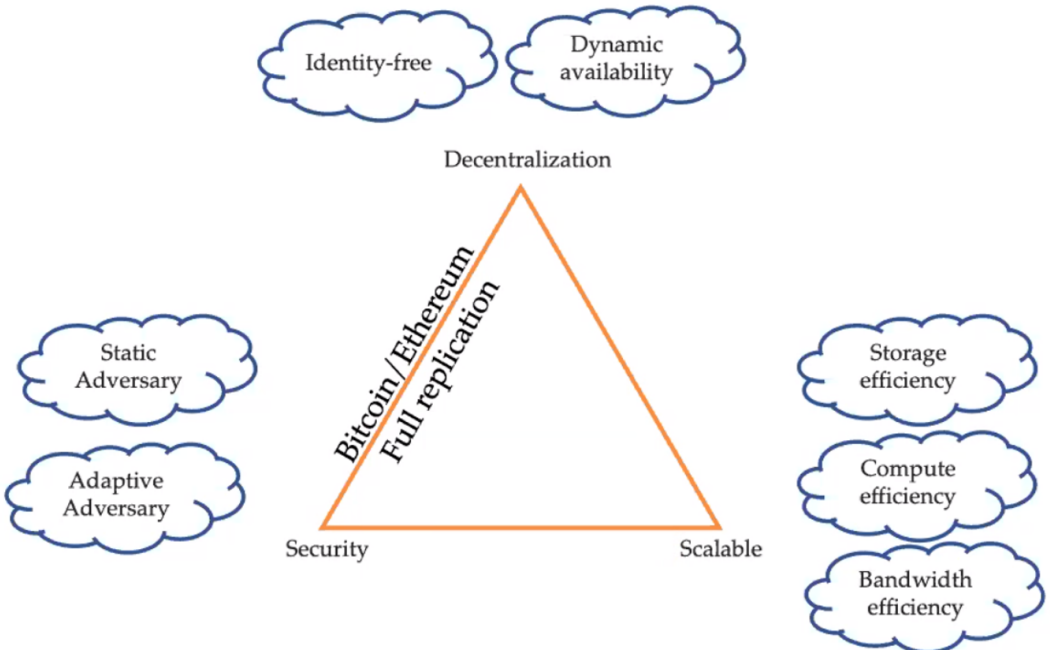

Figure 17.1: Blockchain Trilemma

17.1.1 Security

A secure blockchain system should be able to operate as expected, defend itself from attacks and other unforeseen issues. If some fraction of nodes in the system are adversarial, the system is still able to work as expected, then this system is considered secure.
There are two types of adversary model: static adversary and adaptive adversary. Static adversary model means the status of any node remains the same, that is, if some nodes are honest, then they all remain honest forever; while other nodes which are adversarial remain adversarial forever. Adaptive adversary model means the status of the node may changes over time. Since in reality, some nodes may be honest at the beginning, but a few of them can turn into adversary later due to incentives. Therefore, in the adaptive adversary model, which fraction of the nodes is adversarial will vary over time.

17.1.2 Decentralization

One aspect of decentralization is called dynamic availability, which means nodes can join or leave the system as they please. And as long as the fraction of the adversary nodes is less than the threshold, the system would be fine to keep working. The other aspect of decentralization is called identity-free, which means node has no attribute identity.

17.1.3 Scalability

The scalability of a blockchain system consists of three parts: storage efficiency, compute efficiency and bandwidth efficiency. A blockchain system is scalable if these three efficiencies can increase as the number of nodes in the system increases. The Bitcoin system is not scalable since as more nodes join the system, each node also has to store every block, validate every transaction and download every packet, therefore, none of the three efficiencies have been improved.

17.2 Sharding

To solve the scale problem, there's one idea called sharding. In blockchain systems, ledger is a sequence of blocks, where each block may contain a set of transactions. For current protocol, such as Bitcoin system, every node is required to download the entire blockchain and validate all the transactions. To achieve scaling in this case, one idea is to divide the ledger into K shards, and

each shard is a separate blockchain.

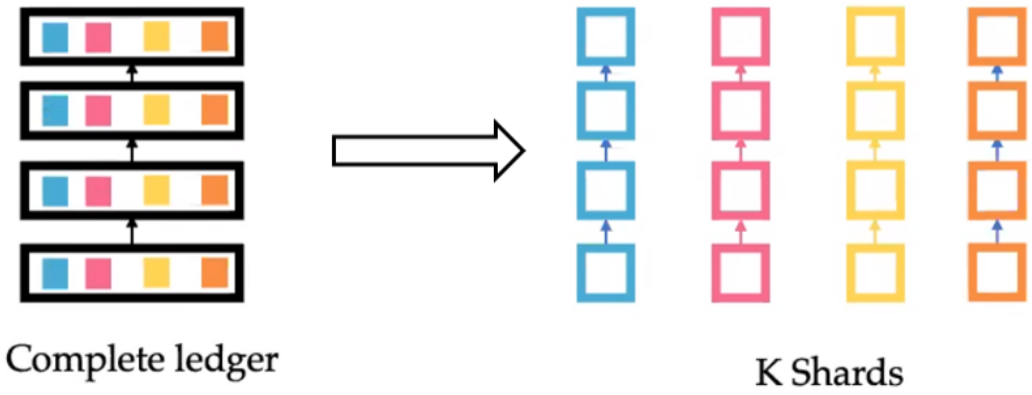

Figure 17.2: Sharding of the blockchain

Two allocation mechanisms are required to realize sharding, where the one is called User-to-Shard Allocation and the other is called Node-to-Shard Allocation. Users here refer to the clients who have user accounts to transact on blockchain, while nodes refer to processing entities who are responsible to maintain the blockchain.

1. User-to-Shard Allocation: Divide different users into different groups. For example, if there are 1000 users, we can divide them into 10 groups with 100 users each group, where the number of shard is 10, i.e. $K = 10$. Thus, there are two types of transactions: (a) Intra-shard transaction, which means the transaction happens inside one shard; (b) Inter-shard transaction, which means the transaction happens cross two different shards.

2. Node-to-Shard Allocation: Divide different nodes into different groups. It must be adversary resistant, which means each shard should have a representative proportion of honest nodes. For example, if the origin population has 80% honest nodes, then each shard should also have roughly 80% honest nodes.

17.3 Multi-Consensus Architecture

Multi-Consensus sharding is one of the major sharding architectures, where each shard has its own consensus procedure.

17.3.1 Performance

1. Security: If group size is large enough, then each group has a representative proportion of honest nodes,i.e. the security is guaranteed.

2. Scaling: Since nodes only need to maintain state of their own shard and each shard runs in parallel with each other, then this mechanism will have a good performance on scaling. The compute efficiency, storage efficiency and bandwidth efficiency would be $O(K)$, where K is the number of shard.

17.3.2 Drawbacks

1. Proportional Representation: Need large number of nodes per shard to ensure honest nodes majority in a shard.

2. Security: Under the adaptive adversary model, the adversary node only needs to corrupt $O(\frac{1}{K})$ honest nodes to attack the security of the system after being allocated into a certain group. For example, if the origin security threshold is $\frac{1}{2}$, the total number of nodes is N and the number of shards is K, then for adversary nodes, corrupting $\frac{N}{2K}$ nodes is sufficient to attack this shard. The security threshold can degrade to $\frac{1}{2K}$. Once a certain shard is non-secure, then the whole system is not secure. So there is a trade-off between security and scaling for this sharding architecture.

17.3.3 About the Trilemma

Multi-Consensus sharding architecture is secure under the static adversary model, but not secure under the adaptive adversary model. Since it uses Node-to-Shard allocation, which is identity based, then the identity-free is also not satisfied. As for the scalability, it can get better performance on improving the three efficiencies by using more shards.

17.4 Identity-free Sharding : Uniconsensus Architecture

The previous Multi-Consensus sharding architecture is not identity-free, so the decentralization is not guaranteed for the blockchain system. If we want the protocol to be identity-free, then we can't use the Node-to-Shard algorithm and the only choice is to allow nodes to self-allocate.

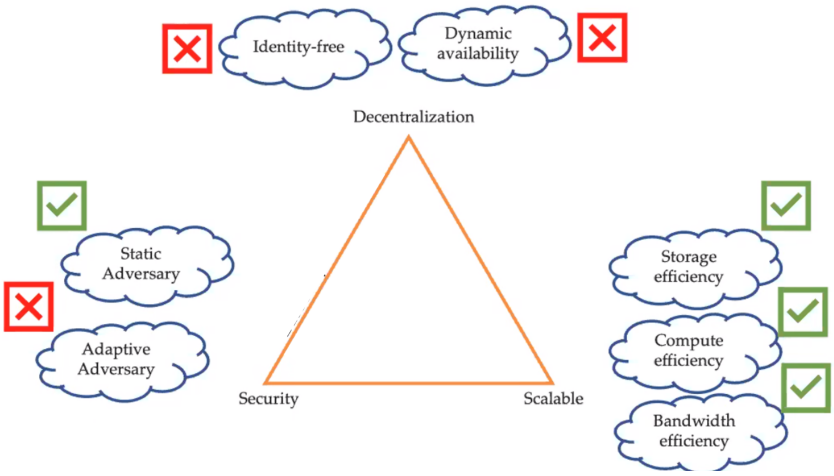

Figure 17.3: Performance of Multi-Consensus Sharding

However, in this case, adversaries can congregate in one shard, where the safety and liveliness can be easily broken. There's another sharding architecture called Uniconsensus architecture, which can solve this problem.

17.4.1 Basic Idea of Uniconsensus Architecture

Instead of running different consensus for different shard, Uniconsensus architecture only maintain one common consensus ordering all the transactions in the blockchain system. In Uniconsensus architecture, shard transaction ordering can be decoupled from validation. All nodes mine shard block and proposer block together, and assume it's in Proof-of-Work protocol, the sortition is based on the hash of the block. And one node can maintain any shard of its own choice, but there is only one main blockchain called proposer chain to order all the blocks.

For example, in Figure 17.5, suppose there are Shard A, Shard B . . . Shard C, and each shard has some blocks, each node can choose one shard as its own please. But only the main proposer chain, which is run by all the nodes in the blockchain system, can determine the orders of blocks from all shards. All the nodes store the main proposer chain, but this chain is considered lite, since each block in this chain only contains the reference of blocks from different shards, not the whole content of blocks. And for nodes in a certain shard, they only mine and store the blocks in their own shard but don't need to download blocks belonging to other shards. Therefore, the computation and storage are lightweight for single node. If a node in Shard C wants to know the

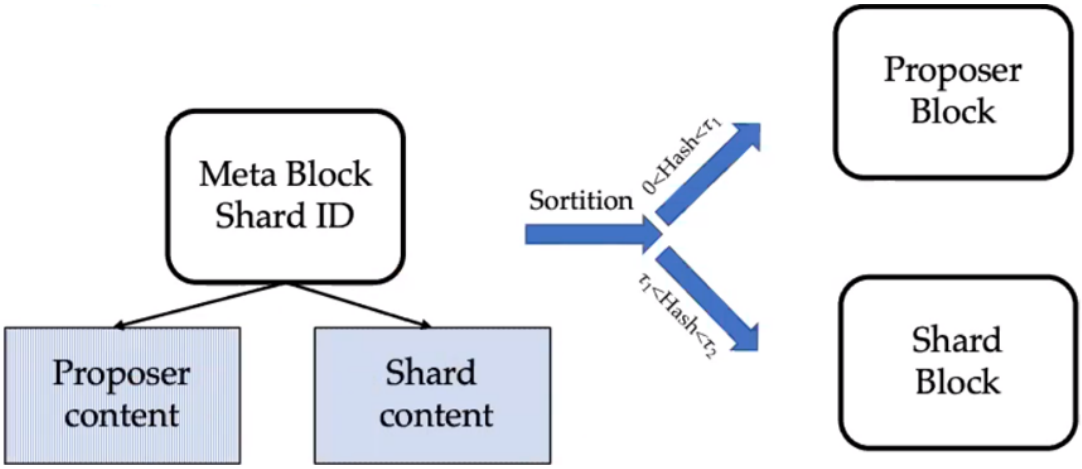

Figure 17.4: Uniconsensus: Sortition

ledger of Shard C, it just needs to read out the all the references of Shard C blocks from the main proposer chain.

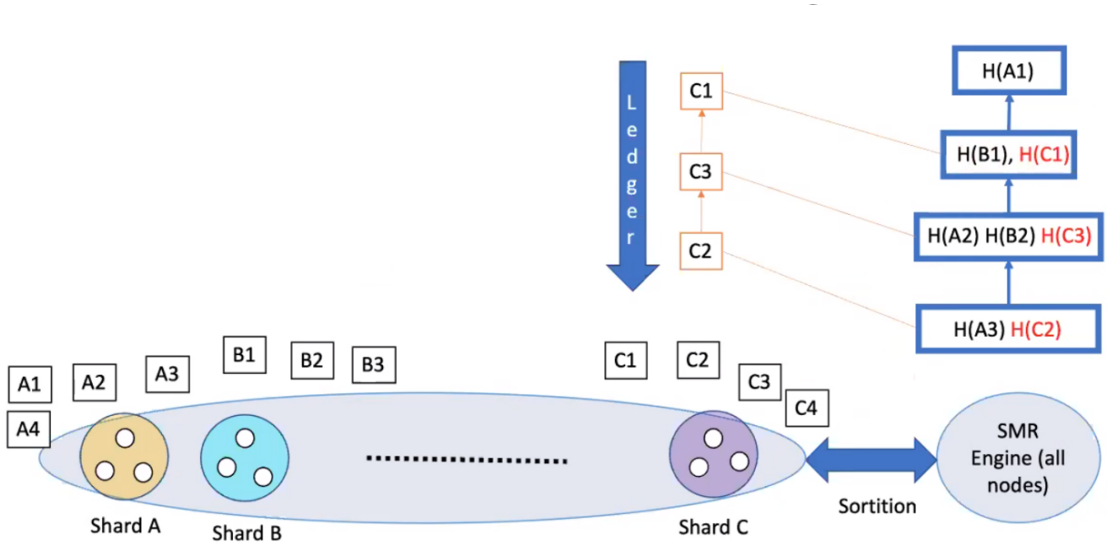

Figure 17.5: Uniconsensus sharding scenario

17.4.2 Safety

In this Uniconsensus architecture, there is only one common consensus for the orders of the blocks, which is the main proposer chain. And this main chain is run by all the nodes in the system, not the nodes in one shard. As a result, adversarial majority in a single shard won't violate the safety of the blockchain system.

17.4.3 Liveliness Attack

Security consists of two aspects: safety and liveliness. There is a kind of liveliness attack for this Uniconsensus architecture, which would violate the liveliness of the blockchain system. Suppose the adversarial nodes congregate in one specific shard, then after they taking over this shard, most blocks in this shard would be mined by adversaries while the honest throughput would fall down to a very small fraction. In this case, the liveliness is not guaranteed in this shard. The worst case shard chain-quality is $O(\frac{1}{K})$.

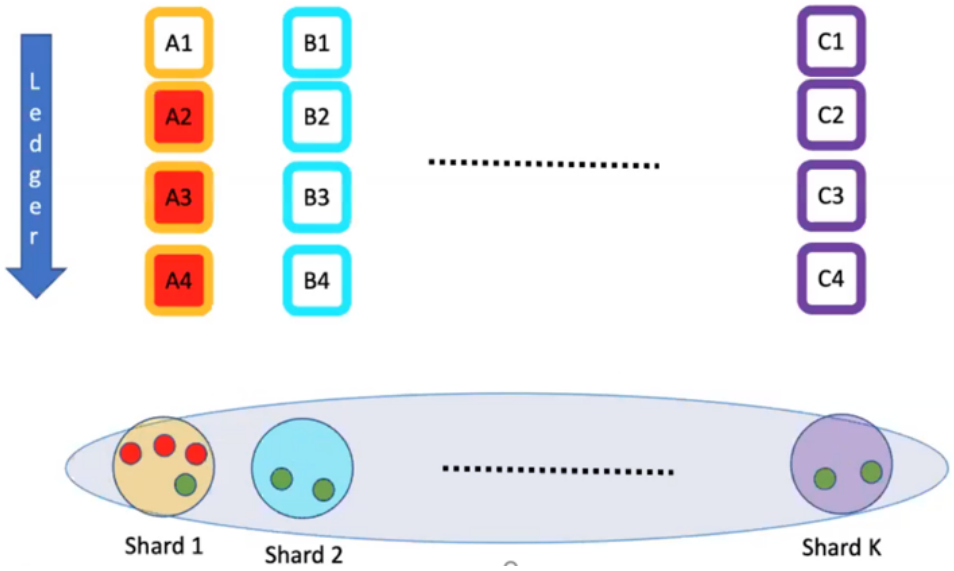

Figure 17.6: Uniconsensus sharding: Liveliness Attack

17.5 Dynamic Self Allocation

Since the worst case shard chain-quality is not good for the previous Uniconsensus architecture when suffering from the liveliness attack, a new algorithm called Dynamic Self Allocation (DSA) can be used to improve the chain-quality. The basic idea of DSA is that honest nodes adapt to adversaries and allocate themselves to new shards, and what we want is the honest nodes to (re)allocate themselves to shards under attack.

17.5.1 Simple DSA

Assume that the adversaries' past allocations are known, and in the simple DSA algorithm, it will adapt to the adversaries by honest nodes following the adversaries' last move. In Figure 17.7, when $T = 1$, there are neutral allocations and when $T = 2$, some adversaries congregate on a given shard A. The honest nodes notice the current allocation of the adversaries, so when $T = 3$, they allocate themselves proportionally to the fraction of adversaries in each shard other than evenly distributing in all shards. In this case, the worst case shard chain-quality is $O(\frac{1}{log(K)})$.

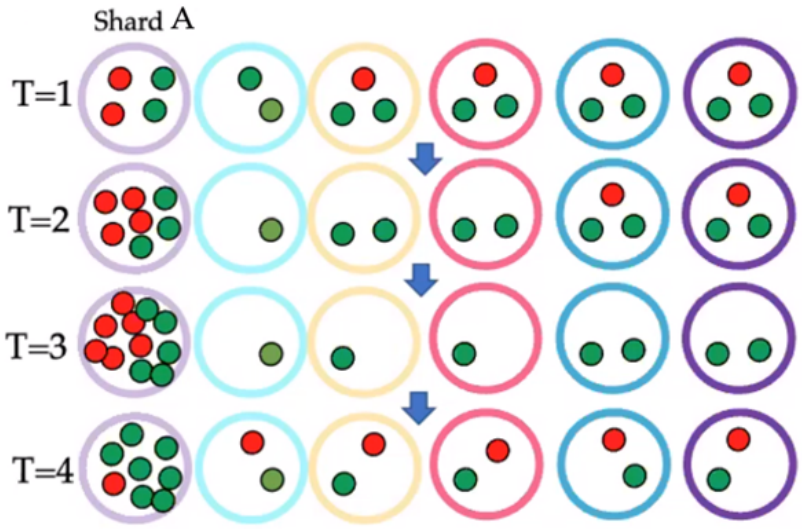

Figure 17.7: Simple DSA scenario

17.5.2 Free2Shard DSA

Although the simple DSA has significantly improve the chain-quality compared to the Uniconsensus sharding without DSA, it still not so ideal and could be better by adopting the Free2Shard DSA. Free2Shard DSA allocates honest mining power to shards proportional to how far behind the average chain-quality of the shard is to a target value. There is an information theoretic bound that the system cannot achieve worst-case shard chain quality better than the global honest mining power. And in practice, the chain-quality will get close to the honest mining power using Free2Shard DSA. It's based on some ideas from dynamic game theory called Blackwell approachability.

17.5.2.1 Advantages

1. Free2Shard DSA supports regimes where $N < K$.

2. Free2Shard DSA enables heterogeneous sharding. In practice, different shards may have different throughput requirements, where throughput can be linked to chain quality. And Free2Shard can handle arbitrary throughput requirements. Figure17.8 shows the simulation results on heterogeneous sharding when $N = 0.1K$.

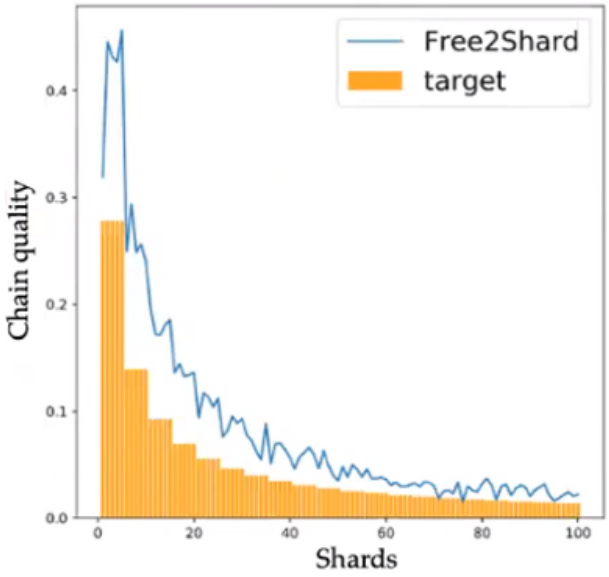

Figure 17.8: Simulation on heterogeneous sharding

3. The results of Free2Shard hold true even if honest nodes can self-allocate at a slow rate and adversaries can rotate instantly.

4. Not all honest nodes need to follow Free2Shard policy, liveliness is achieved even if a small fraction follows the policy.

17.5.2.2 Issues from system view

There are two issues of Free2Shard DSA from system view: Data availability and State commitment.

1. Data availability: Shard block validation is decoupled from proposer chain, so it needs to ensure that referred blocks are available.

2. State commitment: Nodes allocating to new shard need to bootstrap efficiently.

17.5.2.3 Performance

Figure17.9 shows the performance of protocols mentioned in this lecture, and we can see that Free2Shard DSA does well in both the scaling and defending the adaptive adversary.

	Throughput	Computation	Storage	Comm.	Adversary	Adaptive adversary
Bitcoin/Algorand/Prism	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)
Multiconsensus with N2S allocation	O(K)	O(log K)	O(log K)	O(log K)	O(1)	O(1/K)
Uniconsensus without Free2Shard DSA	O(1)	O(log K)	O(log K)	O(log K)	O(1)	O(1)
Free2Shard	O(K)	O(log K)	O(log K)	O(log K)	O(1)	O(1)

Figure 17.9: Performance of protocols