Casanova Consensus

Scalable - Fast - Secure
Choose three.

Blockchain & Consensus

- The consensus algorithm is the low-level mechanism for ensuring that everyone has the same ledger and smart contract state (eventually).

- Algorithm needs to solve the ***Byzantine Generals Problem***
  - BGP is the formal statement of the intuitive problem that computers fail in really unexpected ways.
  - Network needs to be able to find consensus anyway.

Motivation: Security, Performance, Reliability

- Proof-of-Work is probabilistically **secure** and **live**, but very slow and expensive.
- Proof-of-Stake consensus improves performance, but has other drawbacks.
  - **Solana:**
    - Beta network has had significant downtime.
    - Most capacity currently used for voting, instead of regular transactions.
    - Costly to be a validator.
    - Only probabilistically secure.
    - Censorship is possible in a leader protocol.
  - **Avalanche:**
    - Only probabilistically secure.
    - Liveness issues with large validator counts.

Motivation: We Want ***Fast, Safe, and Live Consensus***

Casanova is the ***leaderless, trustless, safe,*** and ***live*** consensus protocol that we are using to build the Silvermint blockchain.

Solutions to the major issues:

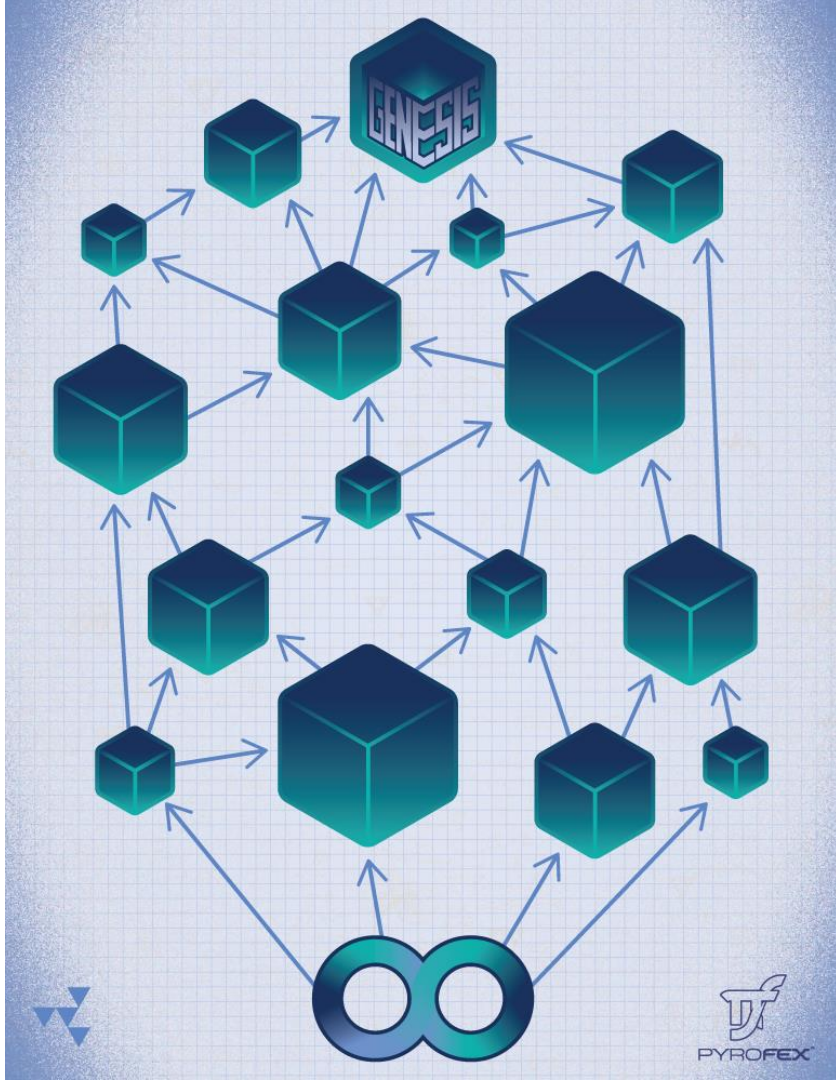
- ***Proof-of-Stake***, so we use normal computing hardware.
- ***Deterministic security*** with true finalization.
- ***Provably live*** under weak synchrony assumption.
- ***Extremely low-latency:*** Milliseconds/seconds instead of minutes/hours.
- ***High throughput:*** at least 100,000+ TPS.

Casanova: Basic Ideas

- Networks scale when you can deploy many nodes, each one contributing capacity to the network.
- Capacity equals space for transactions, so forming consensus using "voting messages" must not overwhelm transactions.
- Transaction Acceptance and Confirmation must be fast in the usual case.
- It should be hard (expensive) to DoS the network.

Directed Acyclic Graph (DAG) of Blocks

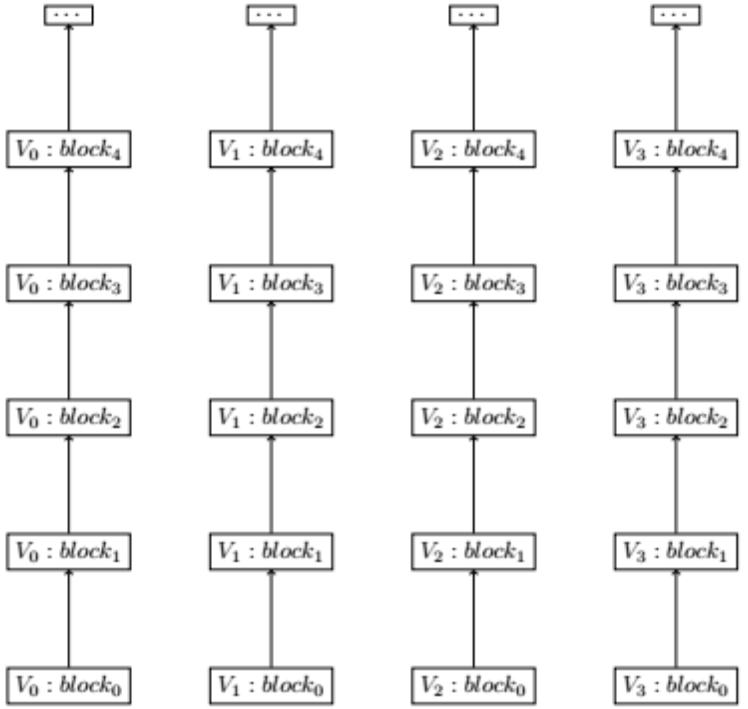
- Casanova uniquely uses a DAG to permit nodes to produce blocks all the time without leader election.
- Each block mentions the blocks observed by the node thus far.
- No voting needed, since the DAG structure deterministically finalizes transactions when operating normally.



DAG Basics: Block Production

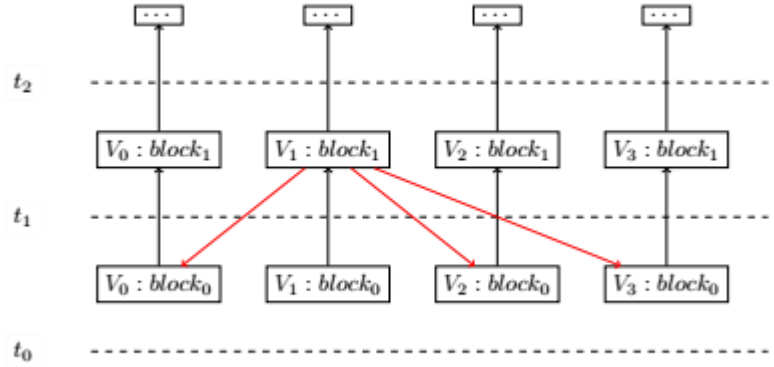
- Every Validator produces blocks all the time on a fixed schedule.
- The network should: ***ACCEPT ALL BLOCKS***, if possible.
- Because: each block adds ***transaction capacity*.**

How does the network "accept" blocks?



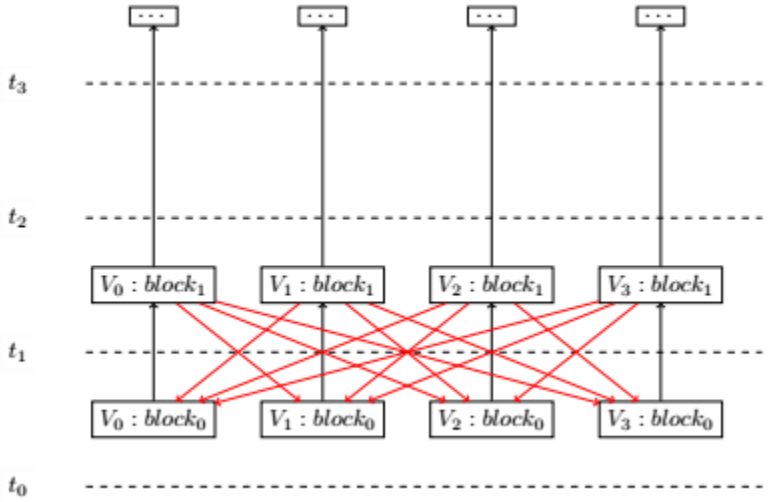
DAG Basics: Block References

- Each validator ***references*** the blocks it ***has already seen*** when it produces a new block.
- The red lines mean: *V_1 has seen these blocks as of $block_1$.*
- I.e., V_{11} ***references*** V_{00}, V_{20}, V_{30}.
- V_1 will no longer accept transactions that conflict with the blocks V_{11} ***referenced.***
- Note: When a block references another block, it is often called a "***child***" of the "***parent***" block it references. I.e., V_{11} is a child of V_{00}, et al.



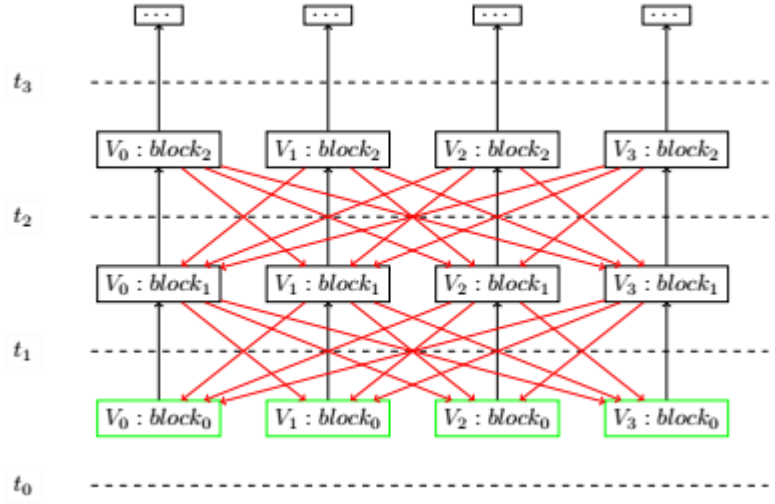
DAG Basics: Block References

- All the blocks produced at t_1 ***reference*** the blocks produced at t_0.
- At t_2, V_0 observes that a fault-tolerant majority (100% in this case) has seen V_{00}, V_{10}, V_{20}, & V_{30}.
- That isn't enough to finalize, but it's almost enough.



DAG Basics: Achieving Finalization

- At t_3, all validators can prove that a fault-tolerant majority has seen the green blocks ***AND*** that the FTM has seen an FTM see those blocks.
- No validators can change their mind anymore.
- So, the transactions in the **green blocks finalize** at time t_3, so long as they don't conflict with any other transactions in green blocks.



DAG Basics: Transaction Finalization

- Casanova finalizes transactions, not blocks.
- Blocks are always accepted by the network.
- Blocks may be empty and contain only references.
- All transactions in a block may be rejected (e.g., if they're all double spends).
- The block is accepted into the DAG anyway, in order to accumulate references, which provide the DAG structure.

DAG Basics: Casanova is Safe and Live

- Safety/Security:
  - Once finalized, ***transactions can never be reverted***.
  - Once finalized, ***transactions can never be re-ordered***.
  - Finalization in Casanova:
    - Nodes can't change their mind about seeing a block.
    - Not enough uncommitted nodes to change the network's mind.
- Liveness:
  - Finalization will occur, given a weak synchrony assumption.
    - Assumption: Messages aren't delayed forever.
    - Implication: Humans have to notice and fix problems promptly or finalization will be slow.

Resolving Conflicts: Conflict Domains

- Transactions in a DAG aren't necessarily ordered relative to each other.
- Nodes must be able to tell when two transactions have to be ordered.
  - E.g., spend the same UTXO.
  - E.g., two users sending bids to a DeFi contract at the same time.
- Network can process all non-conflicting transaction in any order and the result is the same.
  - This is very, very fast.
  - This accounts for 99.9% of all transactions. I.e., it's the usual case in retail payments.
- Formal name for two transactions that conflict is that they both "***occupy the same conflict domain.***"

Resolving Conflicts: Payments

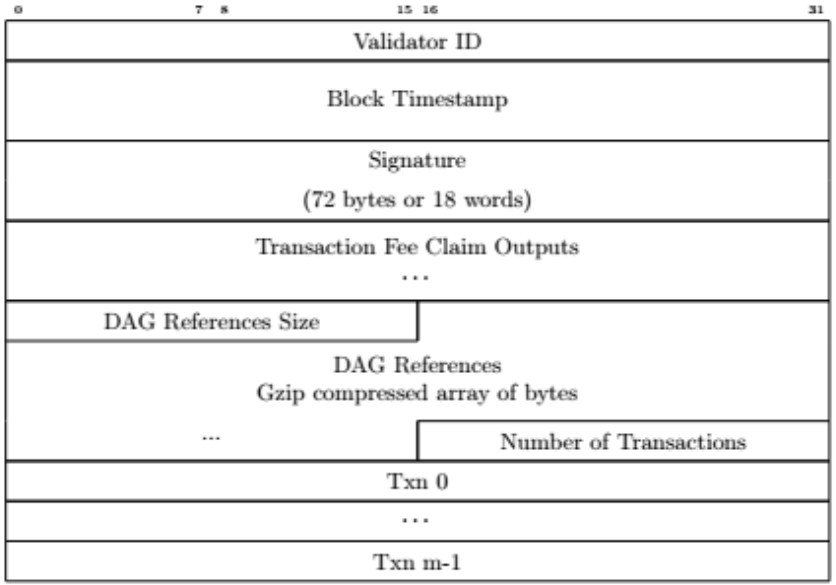
- **Conflict domain is the wallet, UTXO, and a "check number."**
  - Only way for this to happen is a double-spend.
  - User must be malicious.
- Blocks are never rejected or reverted, because nodes use a "line item veto" to determine individual transactions that conflict and handle them separately.
- Network chooses between conflicting transactions deterministically. Usually, ***Whichever transaction has the lower transaction hash, that one wins.*** Can be different if there's a partition where a FTM agrees on the other one before the partition ends.
- Slower than the normal use case, but still quite fast.
- Double spends do not slow down other transactions, so **only the malicious user is harmed** by this.

Resolving Conflicts: Smart Contracts

- **Conflict domain is the smart contract address and an "epoch number."**
  - Can happen when two people make a call to the contract at the same time.
  - Users aren't necessarily malicious.
- Two options for ordering conflicting contract messages:
  - (1) Network chooses between conflicting transactions deterministically: ***Whichever transaction has the lower transaction hash, that one wins.***
    - Advantages: Simple, easy for developers to handle, very very fast.
    - Disadvantages: For DeFi applications, may still permit some limited front-running attacks on bids.
  - (2) Network makes no choice: ***Delivers the transactions to the smart contract with no order and the contract places an order on them.*** E.g., "highest bid gets priority."
    - Advantages: Granular application level control, no frontrunning.
    - Disadvantages: More difficult for developers.

Casanova is Scalable

- The unit of capacity for a blockchain is bytes/second that is ***usable for transactions***.
- In Casanova, all nodes produce blocks.
- Validator bandwidth: 1 Gbps.
- With 2,000 validators:
  - Validators produce 1.25 MB blocks each minute.
  - Blocks can contain more than ***6,000 transactions***.
  - Total capacity over ***140,000 txn/second***.



Optimizations

- Transaction Streaming:
  - Block structure is reversed and transactions are sent ahead of node signatures. This commits a node to including a transaction in just a few milliseconds, so "confirmation" times are <100ms.
- DAG Level
  - Most calculations in the protocol are pure functions of the DAG making them very fast.
  - Blocks can be "archived" in just a few minutes, making the resident block set quite small if the network is operating correctly.

Validators

- Nodes become validators by bonding "stake" using a special transaction.
- The full list of validators as of a given block is a deterministic, purely functional calculation on the DAG.
- Everyone knows what fault-tolerant majority must observe the block to finalize it.
- Attacker must control half of the network to prevent finalization.
- Attacker must control ⅓ of the network and partition the remaining nodes to cause conflicting transactions.
    - Must also prevent node operators in the two groups from talking to each other by phone, email, etc. for 6 months to get away with cashing out their stake.
- Censorship is very difficult, even with >50% of the network.

Major Innovations

- Blocks and transactions are never "rejected."
  - Conflicting transactions have to be ordered and this may cause one to throw an error, but it still exists in the DAG.
- Deterministic List of Validators at every block means that referring to parent blocks is an array of ~2,000 integers (block numbers for each other node).
  - Naively compresses to about 800-1,200 bytes in experiments. Can probably do better.
- Building on a Block is Transitive. You can't build on it without building on everything it builds on.
- Transaction Streaming permits nodes to "confirm" a transaction within milliseconds. Only order can change after that.