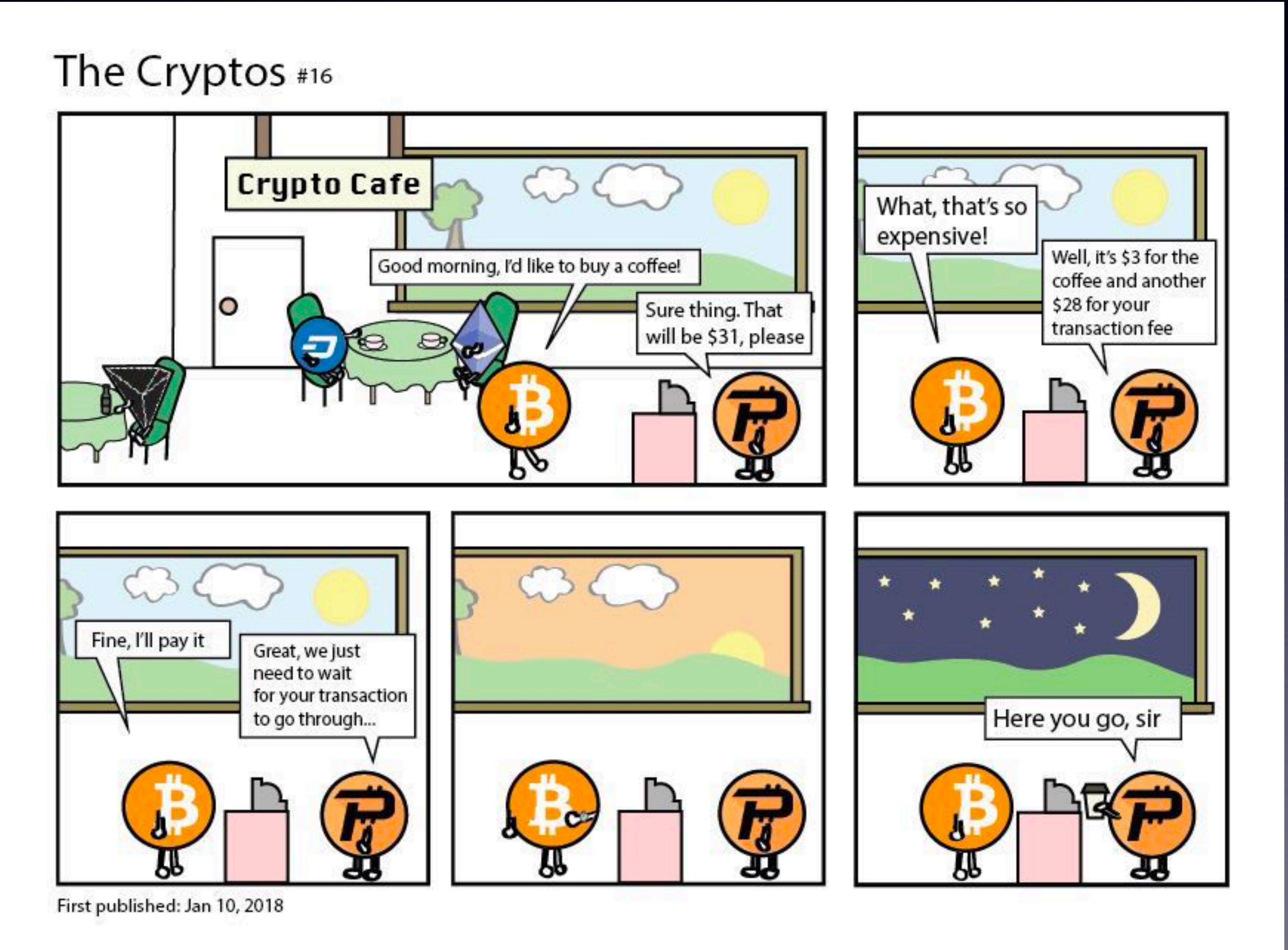
SNARKs and Scalability

Dev Ojha

Current Blockchains are Slow



Bottlenecks In Scaling

- Computational Load (< 20 TPS)
- State Size and I/O (>500GB states)
- Bandwidth
- Consensus Time
- Data Availability
- Lite Client Assumptions

Please stop me for questions about any of these and/or discussions of how these fit into scaling solutions you know of.

With SNARKs

Computational Load
(< 20 TPS)

State Size and I/O
(>500GB states)

Bandwidth

Consensus Time

Data Availability

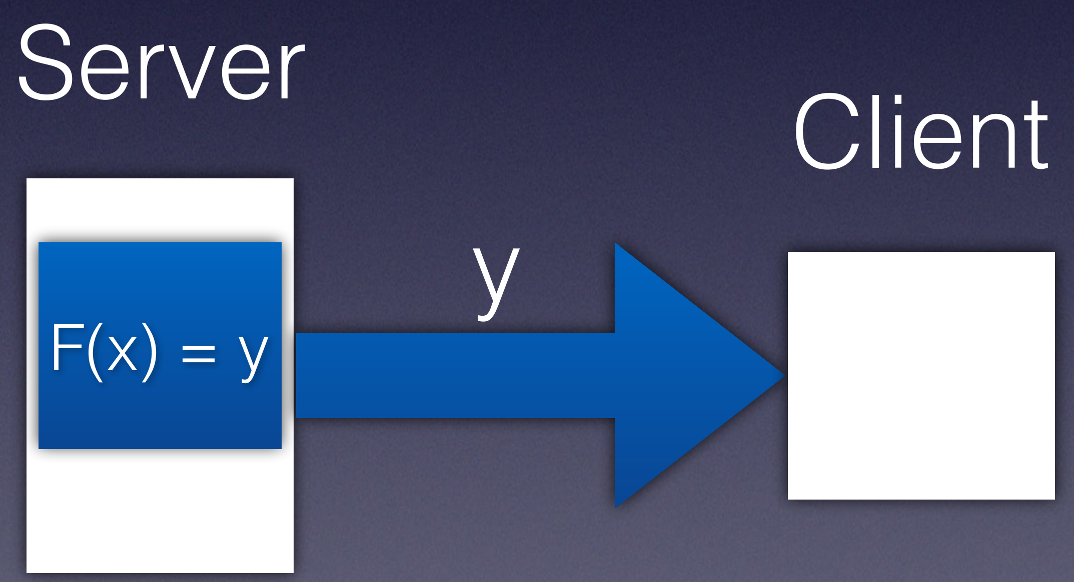
Lite Client Assumptions

(zk)SNARK

A SNARK is a cryptographic proof that a value is the output of a given computation.

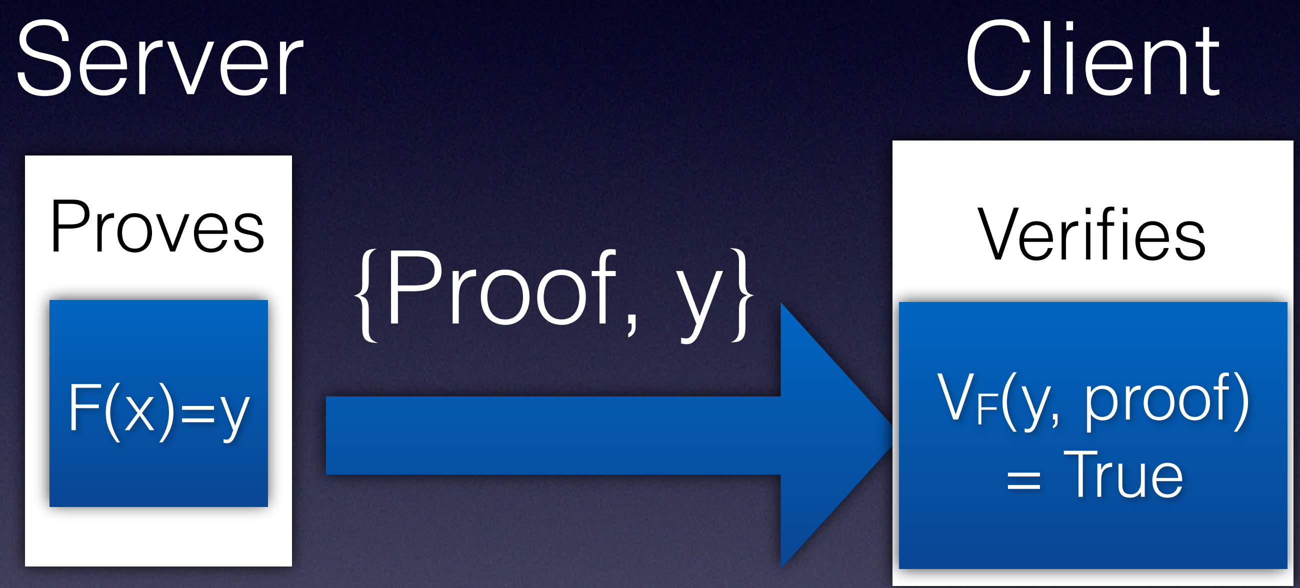
- Let F be a function known to both the server and client.
- Suppose the client wants the output of F on some known input X
- Currently the server just gives the client a value that they claim is F(x), and the client trusts them

Without SNARKs



(zk)SNARK

Suppose F(x) =
H(H(H(H(…H(x || secret))))



- Now the client doesn't have to trust the server!
- This is huge for securing yourself against attacks and malicious companies
- If your phone is a prover, now you can provably hide details.

Stands for

- Zero Knowledge
- Succinct
- Non-Interactive
- Argument
- of Knowledge

Efficiency of SNARKs

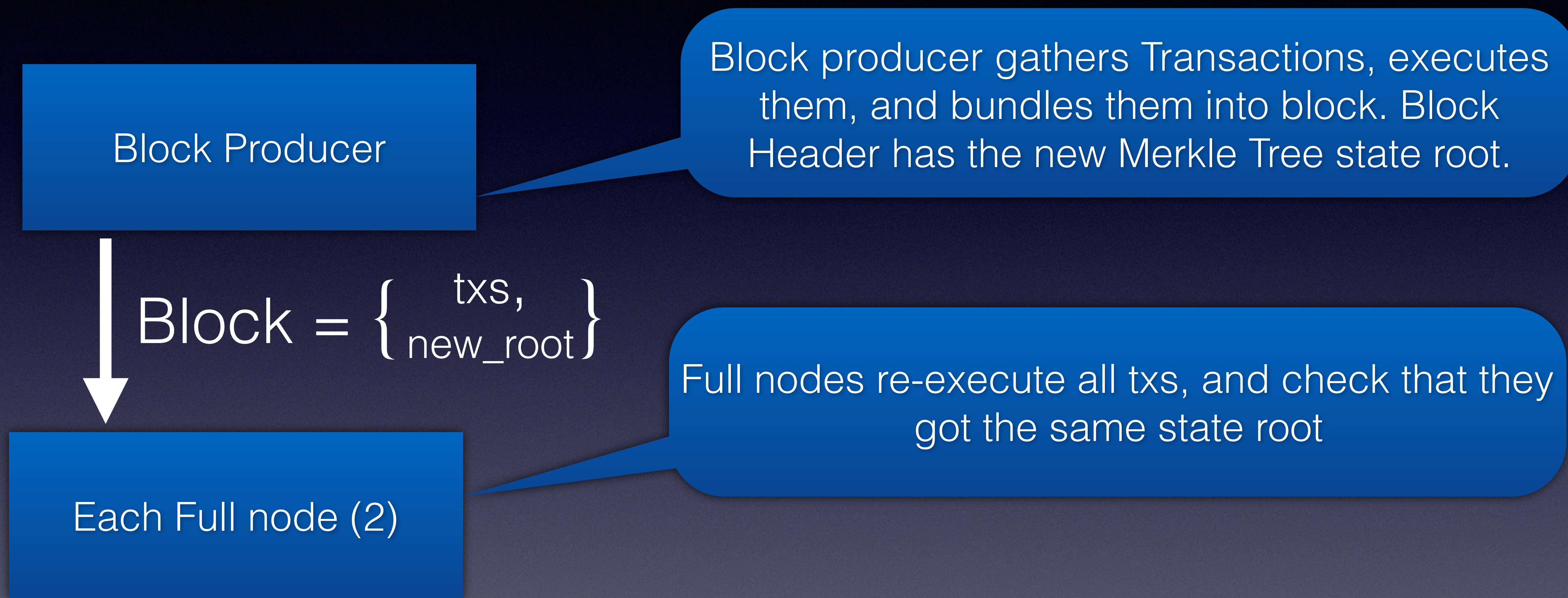
- Continuously Improving
- Verifier time is ~10ms
- A SNARK prover is around 1000 to 5000x overhead on the language it represents.
- SNARK proof size depends on your trust assumption.

Also Post-Quantum

Trust assumption	SNARK name	Proof Size (kb)
Trustless	STARKs	100-200
One Time	Marlin/ Plonk	1-5kb
Algorithm Specific	Groth16	<1KB

Computation in a Blockchain

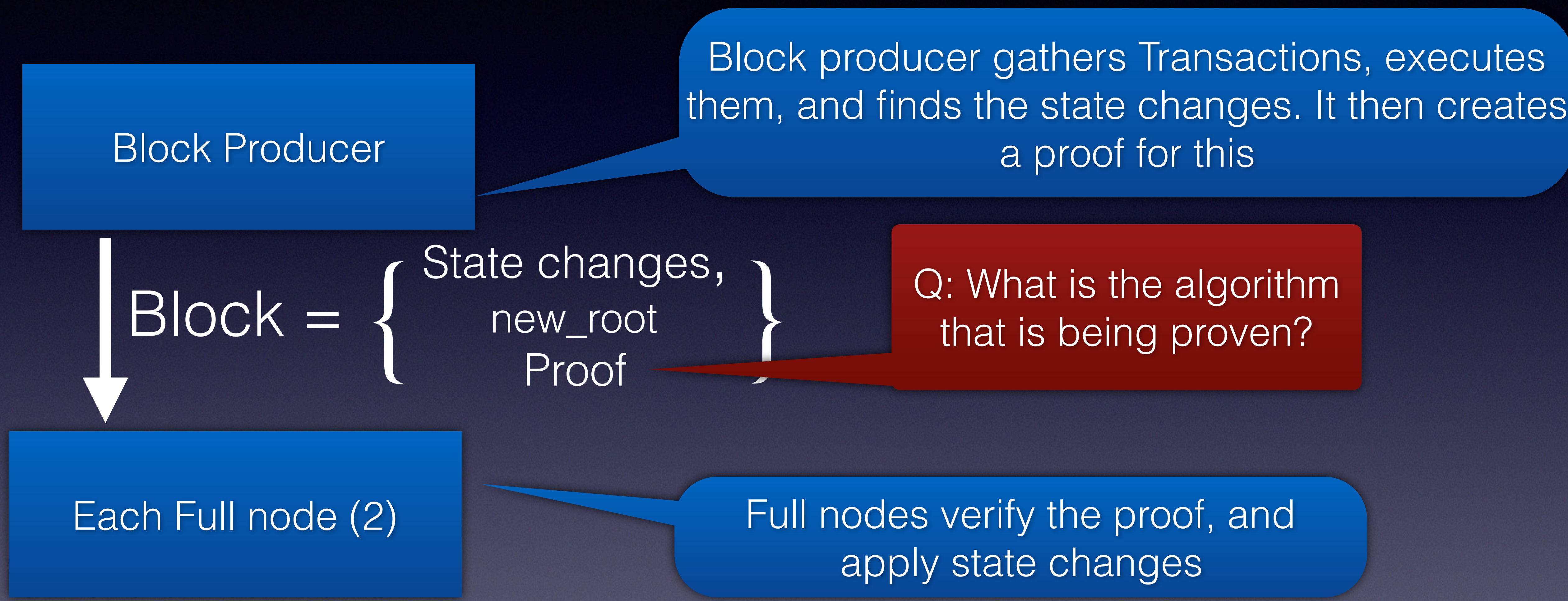
(Ignoring consensus work)



- With N full nodes, and T transactions, there are **TN** tx executions
- The **Slowest** full node must be able to recompute everything

Computation in a Blockchain

With SNARKs



- Full nodes are doing minimal I/O!
- Block producers can pack as much computation as they want into the block using GPU's, parallelism, supercomputers, etc.

To Re-cap, with SNARKs:

Computational Load	You pack as much in as the miner can fit
State Size and I/O (>500GB states)	We have minimal I/O!
Bandwidth	Only communicate state changes, not witness data (e.g. signatures)
Consensus Time	Creating the proof has overheads which could delay consensus if you don’t engineer carefully
Data Availability	Completely orthogonal
Lite Client Assumptions	Up Next!

Lite Clients

A lite client is a small, computationally restricted node such as a phone or IOT device.

- A lite client receives every single block header*
- They check that consensus was applied correctly between each block
- They then delete the old headers and just keep the headers they care about
- This requires magically trusting state to be computed correctly between each block.

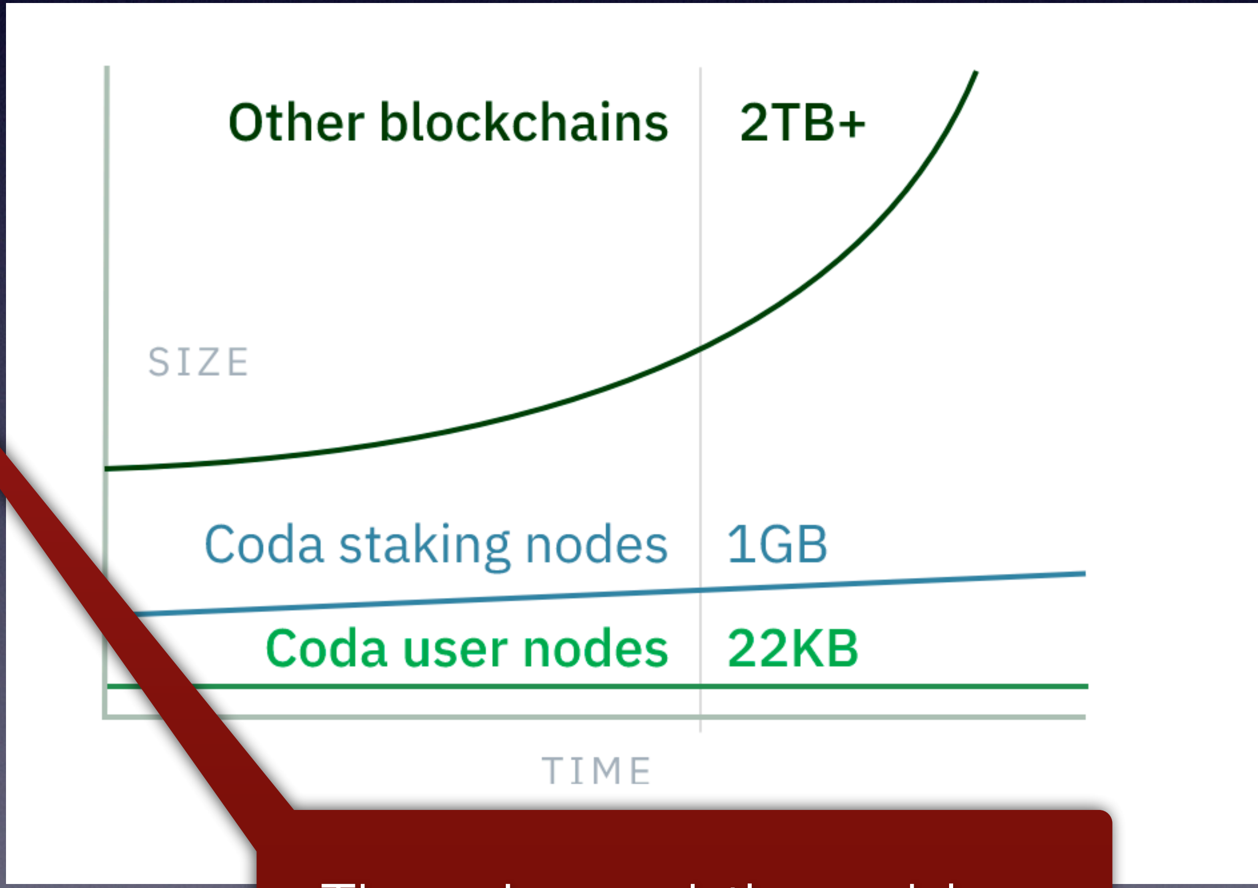
They could further magically trust their full node and skip some of this, with increased risk

This can take days, and uses gigabytes of bandwidth.

Lite Clients with SNARKs

- A lite client receives the latest block header and a proof that this block and all prior blocks are valid
- Valid means that both Consensus was valid, and all txs are valid.
- This solves syncing time, and the trust assumptions

Coda - A new blockchain that does this

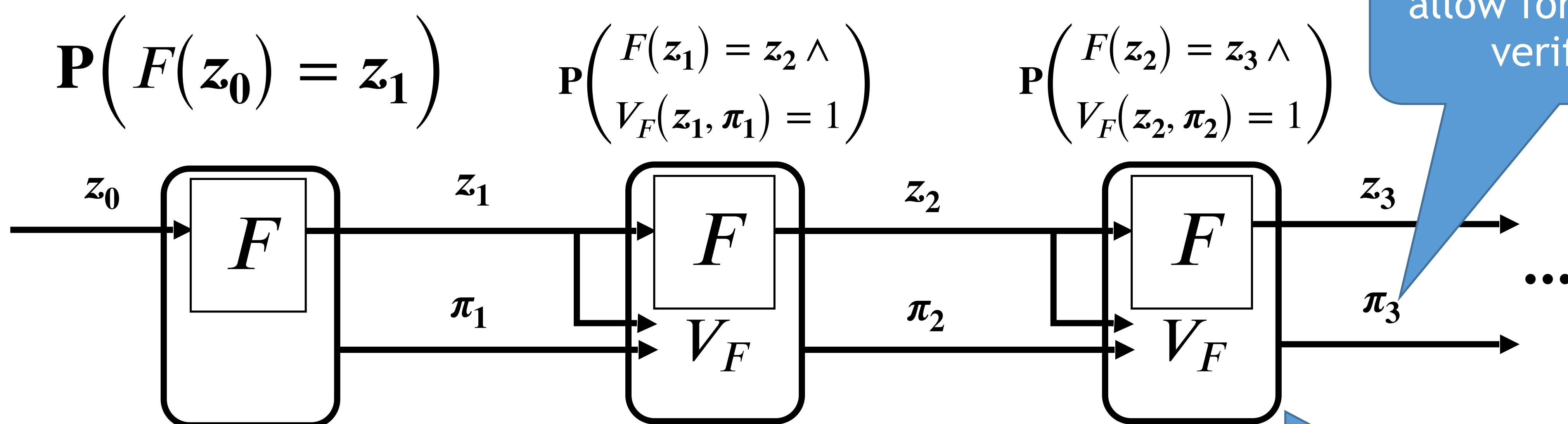


There is a subtle problem here, any guess at what it is?

Recursive composition of SNARKs

Recursion: verifying a SNARK proof *within* a SNARK

e.g.: verifying iterated function application



intermediate proofs allow for continuous verification

sizes of function and proof are *independent* of

Other Applications:

- Succinct Blockchains (e.g. Coda)
- Verifiable Delay Functions [BBBF19]
- SNARKs for MapReduce [CTV15]
- Internet with all computed results authenticated
- ...

Thank you