Consensus Among Computer Networks

Darren Tapp

Arizona State University

November 18, 2020

Consensus

Academic organization:

- Byzantine Fault Tolerance
- Nakamoto Consensus
- Dash Consensus

Citations

- BFT: Several peer reviewed articles, e.g. "Practical Byzantine Fault Tolerance"
- Nakamoto Consensus: "Bitcoin: A Peer-to-Peer Electronic Cash System"
- Dash Consensus: Dash Improvement Proposals 2,3,6-8

Byzantine Fault Tolerance

For an open network

Byzantine Fault Tolerance

- Is vulnerable to Sybil attacks
- Fails (generally) if over one third of nodes are malicious
- Requires multiple communication rounds

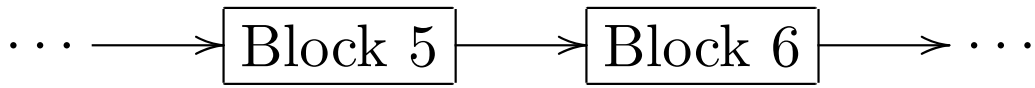
Nakamoto Consensus

For an open network

Nakamoto Consensus

- Mitigates Sybil attacks with proof of work
- Can be verified by a passive node
- Generally runs at a stable Nash equilibrium

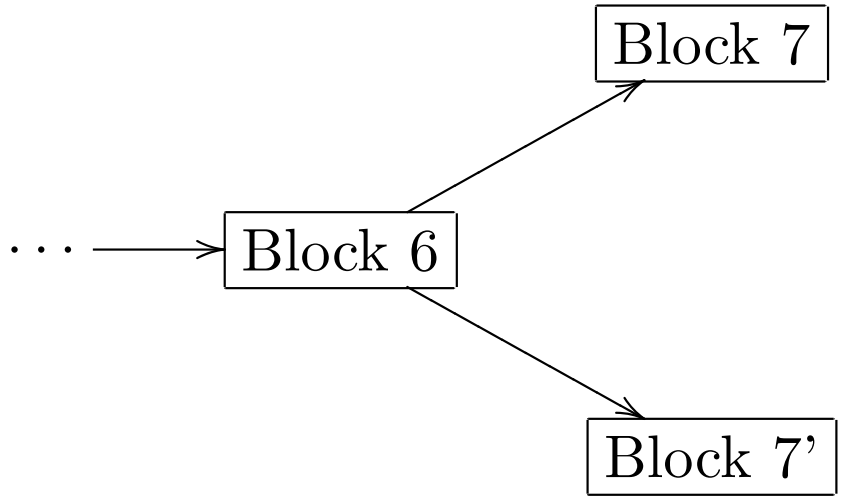
Nakamoto Consensus



The consensus is about a global state (UTXO) and the blocks are instructions for changing the global state (transactions.)

Nakamoto Consensus

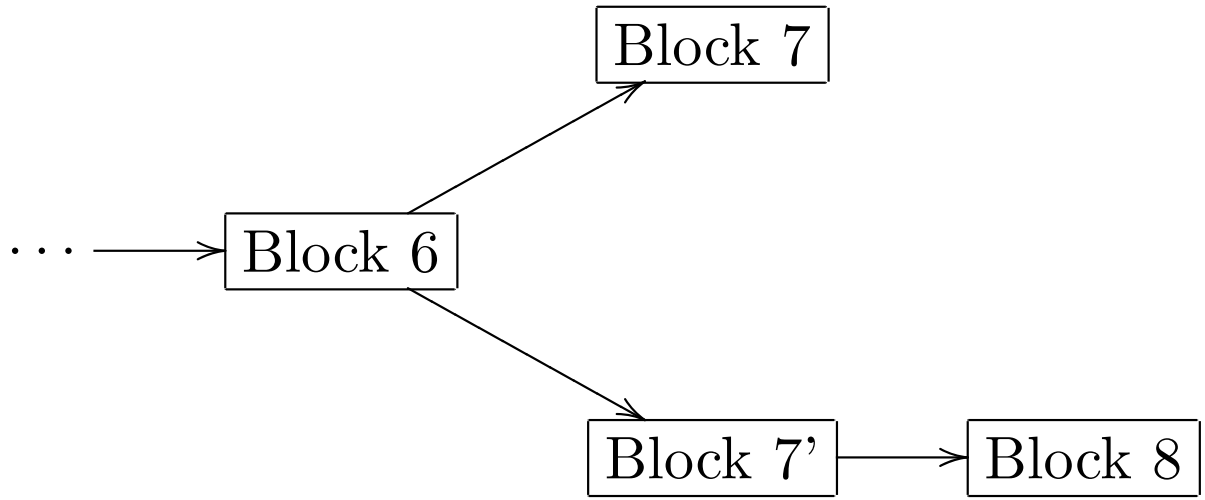
A Nakamoto network could be out of consensus



This diagram shows a blockchain where the most recent changes could be in dispute.
Consensus is still maintained over old changes.

Nakamoto Consensus

However, the network will collapse into a consensus by following the longest chain



Dash Consensus

Outline:
The remaining portion of this seminar will be for explaining existing Dash consensus implementations

- Quick overview of what a Dash consensus is
- Choice of threshold signature scheme
- Byzantine Resistance; "Quorum Math"
- Applications of ChainLocks and InstantSend

What is Dash Consensus?

A Dash consensus requires a blockchain to provide Nakamoto consensus.

1. Register actors on chain with public key
2. Select subset, called a quorum, of registered actors
3. Produce threshold signature of statement/datum that is in consensus

Nomenclature

If M actors are registered and a subset on n actors is selected and a threshold of t out of the n actors is required for signature.

We call this a:

$$\mathrm{Dash}(M, n, t)\ \mathrm{Consensus}$$

for security reasons we assume $t > \frac{n}{2}$.
We also assume that the quorum of n nodes is a simple random sample of the M registered.
In practice, a simple random selection is simulated using hash functions.

Register on Chain

Details for Transaction

Hash	000386691c7330f4cee32ef149f56cebb91f2de9ff819ee2411c69c68196fedd
Block Height	1176917 :52 (194686 confirmations)
Block Date/Time	2019-11-25 12:07:26
Total Output	0.09866102 DASH
Fees	4.79E-6 DASH

Inputs / Outputs | Raw Transaction

Inputs

Index	Previous output	Address	Amount
0	7931be8acdded98d...:0 in 1176917	XqfLWBNn4cFnBtJbhxF6bgkKMsdJKqtdYf	0.09866581 DASH

Outputs

Index	Redeemed in	Address	Amount
0	05d337a78d08bfba... in 1176982	XqfLWBNn4cFnBtJbhxF6bgkKMsdJKqtdYf	0.09866102 DASH

Register on Chain


```
"extraPayload": "010000000000b316b1e28c958b1e2f08f8bf37297e269d24938f87ad55ef2974dc8803ec968b0100000000000000000000000000000000ffff8b3
"proRegTx": {
   "version": 1,
   "collateralHash": "8b96ec0388dc7429ef55ad878f93249d267e2937bff8082f1e8b958ce2b116b3",
   "collateralIndex": 1,
   "service": "139.59.91.7:9999",
   "ownerAddress": "Xb2ZLakfoSRiJzhr8f71ueZfg8uj8dFwzb",
   "votingAddress": "XtMJytB4mTtwusXGBosZ5HP588moxFhJnP",
   "payoutAddress": "Xc6tLizDnn1YgnuTuGAH2BfvRQrPuk1Bfk",
   "pubKeyOperator": "87ff1c096a79b20ca7e921491f99a652dac04cb1f9c18d0c0e98f88132f8910494f4078cd2ce28baf5ab60bf904a15cd",
   "operatorReward": 0,
   "inputsHash": "a00b9780d03a2b6dcf6d7822a1a976b5133b5efde9caff3455c110efb72faa6a"
},
```


collateralHash and collateralIndex point out 1000 dash on chain.
This 1000 dash requirement mitigates a Sybil attack.
ownerAddress, voterAddress, and payoutAddress are base 58 encoding of ECDSA keys.
pubKeyOperator is a hexadecimal encoding of a BLS public key.
This key is what's used for Dash consensus.

Choice of Threshold Signature

We'll compare two threshold signature schemes:

- Schnorr signatures
- BLS signatures

Naïve Multisignature

For Schnorr and BLS signature schemes,
public keys are points on elliptic curves.

If three actors publish their public keys P, Q and R
a multisignature is a valid signature for $P + Q + R$.

Related-key Attack

Alice Bob and Malory want to construct an multisignature.

- Alice obtains private key a with public key A and publishes A
- Bob obtains private key b with public key B and publishes B
- Malory obtains private key c with public key C
  but publishes $C - A - B$

Then an aggregate signature will be for the key

$$A + B + C - A - B = C.$$

Since Malory knows the corresponding private key of C
he can construct an aggregate signature without Alice or Bob.

Advantages

Schnorr signatures can work with ECDSA keys that are currently used for Dash payments.

BLS signatures are easier to construct on the fly and they have a smaller attack surface.

Long Story Short

Schnorr signatures require each party to provide entropy.
This entropy is combined before a signature can be constructed.
If something goes wrong in the entropy stage
then the process must start over.
Reusing entropy could allow private keys to be solved for.

BLS signatures do not require entropy.

Citation

Signing. Let X_1 and x_1 be the public and private key of a specific signer, let m be the message to sign, let $X_2, \ldots, X_n$ be the public keys of other cosigners, and let $L = \{X_1, \ldots, X_n\}$ be the multiset of all public keys involved in the signing process.[7] For $i \in \{1, \ldots, n\}$, the signer computes

$$a_i = H_{\text{agg}}(L, X_i) \tag{1}$$

and then the "aggregated" public key $\widetilde{X} = \prod_{i=1}^{n} X_i^{a_i}$. Then, the signer generates a random $r_1 \leftarrow_{\$} \mathbb{Z}_p$, computes $R_1 = g^{r_1}$, $t_1 = H_{\text{com}}(R_1)$, and sends t_1 to all other cosigners. Upon reception of commitments $t_2, \ldots, t_n$ from other cosigners, it sends R_1. Upon reception of $R_2, \ldots, R_n$ from other cosigners, it checks that $t_i = H_{\text{com}}(R_i)$ for all $i \in \{2, \ldots, n\}$ and aborts the protocol if this is not the case; otherwise, it computes

$$R = \prod_{i=1}^{n} R_i,$$
$$c = H_{\text{sig}}(\widetilde{X}, R, m),$$
$$s_1 = r_1 + c a_1 x_1 \bmod p,$$

and sends s_1 to all other cosigners. Finally, upon reception of $s_2, \ldots, s_n$ from

Cite page 11 of:
"Simple Schnorr Multi-Signatures with Applications to Bitcoin"

Citation

it does not have the restriction that the subset of signers should be known in advance. We then propose the new GDH multisignature scheme MGS. It works in any GDH group. Our MGS scheme solves the open problem stated in [33]: it does not require a priori knowledge of a subgroup of signers and is provably secure. We state the security result and provide a proof in [5] . Moreover, MGS is more efficient than the one of [33] which requires three rounds of communication for the multisignature generation protocol, where MGS requires only one, it is basically non-interactive. Similarly to their scheme, the signature length and verification time for MGS is independent of the size of the subgroup and is almost the same as for the base signature scheme. In fact each signature share of our multisignature scheme is the standard GDH signature. In the scheme of [33] a signer is not allowed to begin a new signing protocol until the previous one has completed. This is because their proof of security uses rewinding which is incompatible with concurrency. Our scheme does not have such restriction not only because our proof does not use rewinding but mostly because the signing protocol is non-interactive.

We note that the approach underlying the construction of the multisigna-

Cite page 35 of:
"Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme"

Clarification

A BLS threshold signature, has an interactive
Distributed Key Generation phase.

After that, all threshold signatures are non-interactive.
For a t out of n threshold signature. Any party with $> t$ signature shares can create a valid threshold signature.

Byzantine Resistance

A Byzantine actor attempts to disrupt or alter $\mathrm{DASH}(M, n, t)$ consensus by registering or controlling nodes.
There are two types of attacks that a Byzantine actor can employ.

Byzantine DOS

The actor has control of and/or has disabled $> n - t$ nodes to prevent a signature from being formed

Byzantine Hijack

The actor controls $\geq t$ nodes and can produce any signature.

Byzantine Resistance

When considering a $\mathrm{DASH}(M, n, t)$ consensus. There are M choose n possible quorums that can be selected. Recall that M choose n is a binomial coefficient and written

$$\binom{M}{n}$$

If b out of the M nodes are Byzantine, and $b > t$ then there are

$$\binom{M-b}{n-t}\binom{b}{t}$$

quorums that contain exactly t Byzantine nodes. So the probability that a quorum has more than t Byzantine nodes is

$$\frac{\sum_{j=t}^{\max(n,b)} \binom{M-b}{n-j}\binom{b}{j}}{\binom{M}{n}}$$

Byzantine Resistance

For a $\mathrm{DASH}(5000, 400, 240)$ consensus.

Byzantine Proportion	Byzantine Nodes	DOS probability	Hijack probability
0.05	250	4.19e-125	1.27e-284
0.10	500	3.31e-65	7.11e-157
0.20	1000	1.68e-22	2.89e-76
0.30	1500	3.37e-06	1.29e-38
0.34	1700	3.81e-3	1.23e-28
0.40	2000	0.478	3.21e-17
0.50	2500	0.99999	1.82e-05

Source: Python2 Calculation

Byzantine Resistance

For a $\mathrm{DASH}(5000, 50, 30)$ consensus.

Byzantine Proportion	Byzantine Nodes	DOS probability	Hijack probability
0.05	250	3.83e-15	3.25e-27
0.10	500	2.81e-09	3.10e-18
0.20	1000	2.98e-4	5.42e-10
0.30	1500	4.69e-2	9.53e-06
0.34	1700	0.147	1.39e-4
0.40	2000	0.439	3.22e-3
0.50	2500	.900	0.10

Source: Python2 Calculation

Applications

With M active masternodes on the Dash network

InstantSend

A $\mathrm{DASH}(M, 50, 30)$ consensus is used to identify first seen transactions. Once a transaction has a InstantSend lock network (Nakamoto) consensus rules will not accept a conflicting transaction.

ChainLocks

A $\mathrm{DASH}(M, 400, 240)$ consensus is used to identify a first seen block. Once a mined block is signed then the network will not reorganize before the signed block.