# Introduction to blockchain theory: CAP Theorem

The fundamental contribution to network communication that blockchain brings is to provide a method for trust and fact verification independent of a centralized authority. Blockchain provides a means for all network participants to agree on an immutable and enduring record of occurrences by means of essentially deterministic and unbreakable algorithms.

Blockchain is a complex subject combining deep and clever usage of cryptography, game theory and software. However it is not magic, and before discussing in detail the history or mathematical and programmatic mechanics of blockchain, it may be helpful to put in perspective the context in which blockchain theory operates in order to appreciate the difficulty of the problems it attempts to solve, and the necessary trade-offs which are inherent in the problem space - distributed network computing and data storage.

## CAP Theorem

In theoretical computer science, the CAP theorem, also named Brewer's theorem after computer scientist Eric Brewer, states that it is impossible for a distributed data store network to simultaneously provide more than two out of the following three guarantees:
[1] Consistency
[2] Availability
[3] Partition tolerance
(hence the name 'C-A-P' Theorem)

Since a network is trivial and useless if it contains a single point, all systems of interest are assumed to consist of multiple inter-connected nodes, and most likely massively multiple. These networks realistically are not perfectly robust so suffer occasional breaks in communication from one part of the network to another. Therefore it is essential that a non-trivial realistic network must satisfy [3]. The CAP theorem then states that any partition tolerant network cannot be both perfectly consistent and absolutely avaiable.

## CAP Theorem and blockchain

Leaving aside for now the specifics of the blockchain methodology, let's focus on how blockchain (or any distributed network data store) attempts to make trade-offs between [1] consistency, and [2] availability, of network data. Understanding the impossibility of the co-existence of both complete consistency and immediate availabilty of data will provide a backdrop for each of the various designs for blockchains - the varied mechanisms for consensus, and the scale and speed of transactions.

In order to understand the consistency-availability dilemma only a very simplified idea of the blockchain is required. At its most basic a block is a collection of transactions during a small period of time (we use the term transaction loosely since the data stored in the blockchain can be numeric, textual, media , code or anything at all referable by a digital record.) Blocks are created essentially asynchronously but must be 'verified' to be added to the chain. Every block in the chain has a single parent block, but a block may (temporarily) have more than one child. By some methodology the one or more contemporary 'candidate' blocks must be formed into a single growing linear chain which is then immutable, persistent and redundantly available in a distributed system. A final important point is that each block contains a crytogarphic record of its parent block so that all blocks are 'chained' together and each block contains a sort of fingerprint which points back into the complete history of the chain.

The CAP Theorem has a mathematical proof so it is absolute - no distributed partitioned data store network can communicate data which is both entirely consistent across the network and at the same time immediately available to all users. In addition to a non-trivial network needing to be partitionable, any practical network must store data which is at least eventually available (or it is useless) and eventually consistent (or the data is meaningless, being self-contradictory). Therefore the state of a distributed data store network is like a quantum 'superposition' or 'mix' of some portion of both eventual availability and eventual consistency.

Every distributed system chooses by its design which proportion of each that it permits, and blockchains are no different. Depending on the design of the particular blockchain's client, the system will favor either consistency or availability of its data.

## Example of consequences for blockchain of stressing either AP or CP

In order to show these design tradeoffs and consequences of the choices, let's consider a simple but somewhat unrealistic example. Suppose Jack wants to buy ten dollars of some cryptocurrency, say 'crypto', from a seller Jill. Leaving aside details, suppose they create a transaction which contracts the transfer of Jack's ten dollars to Jill's digital wallet, and Jill's equivalent crypto amount to Jack's wallet.

### [2][3] AP

Let's first illustrate a scenario in which the blockchain client stresses immediate availability of the data (funds) over consistency of the transaction record. Let's say the Jack-Jill transaction is collected into a block B1 by some blockchain 'miner' (more on mining and proof-of-work later), and another miner far away from the network location of Jack but perhaps near Jill, assembles a block B2 which does not contain the Jack-Jill transaction (due to network latency of Jack's actions, or maybe not sufficient transaction fees to attract the miner to include the transaction in his block (more on fees later also). In any case there are at least two different unverified blocks B1 and B2 vying to be verified and added to the blockchain record.

Suppose, since Jack is nearer B1, he sees that the transaction is in the block. Since the blockchain stresses immediate availability of the funds (AP) Jack says 'great, I'll send my dollars now and wait for Jill's crypto'. So Jack sends his ten dollars to Jill. Meanwhile Jill, nearer to B2, fails to see the transaction in B2 so does not release her crypto, but to her surprise finds ten dollars appearing in her wallet.

Further suppose, by whatever consensus mechanism implemented on the blockchain, that users agree that the blockchain should grow using B2 as the new 'head' or 'parent' block, and leave B1 and its packaged transactions 'orphaned', i.e. not on the blockchain. Then there is no record of the Jack-Jill transaction on the immutable blockchain ledger. Immediate availability and delyed consistency has left Jack poorer by ten dollars and Jill profiting by ten dollars at no cost in crypto. Obviously this is a problem for the AP blockchain.

**[1][3] CP**

Now suppose another blockchain implementation comes to market which, having learned from the unfortunate cases arising in the AP blockchain, implements a mainly CP design by declaring that a transaction not be considered by users to be valid until it has been recorded in blocks no fewer than N links in the past. As we shall see, the deeper a block is in the chain (the further in the past it was added to the chain) the smaller the chance of the block being invalidated (we will explain all this fairly soon) and all its transactions lost - allowing the Jack-Jill 'theft' described above.

Forcing consistency by needing to wait for an accumulation of N blocks before releasing funds solves the 'theft' problem of only one of an inconsistent pair of blocks entering the blockchain record (as described in AP above). However, many clients will be very angry since their transactions take a relatively long time to finalize. Let's give some concrete numbers to show the downside of the CP case. Let's say that it takes an average of one minute to create a block, tentatively verify it's correctness and add it to the chain, possibly alongside other forked subchains all vying for inclusion in the parent blockchain (more on forking chains later in the discussion of mining). If the blockchain requires a transaction to be at least five blocks deep in its chain for the contained transactions to be considered valid, then at least five minutes pass before funds are released in a transaction. This delay applies to any and all transactions, so fights break out in checkout lines across the landscape as frustrated customers wait one after the other to pay for material goods. Similarly, even a minute of delay is an eternity for a cryptocurrency trader since currency prices are extremely volatile and a good deal now may be a very bad deal in moments, if follow on trades cannot be consumated quickly.

Therefore assuring eventual consistency at the cost of delayed availability also has its problems.

## Scale

Blockchain is an ingenious technology since it essentially solves the problem of non-centralized network 'trust' and its mechanisms tend to emphasize consistency and security at the cost of availability (which as seen has problems when enforced in the extreme). However there is a further problem which is a disturbing reminder of the problems of previous data-storage and access systems.

Data storage matured rapidly with the development of relational database theory. These systems allow an algebraic data manipulation and access but require direct access to all data to perform the algebraic operations required by SQL. In terms of the CAP Theorem relational databases are essentially AC, and do not allow partitioning of the storage.

Although blockchains allow unlimited partitioning of data and maintain some mix of usually strong consistency/security and only eventual availability, blockchain shares a scaling problem (in a different form) with relational databases. It may seem that a blockchain can solve its scaling problem easily since it is distributed. This works well for systems such as distributed file systems, but these system have much weaker consistency and consensus constraints. Also, in the case of blockchain, the distributed aspect of the data is mainly massive redundancy of data, not true distribution.
As more and more transactions are made this storage problem increases possibly exponentially, whereas many solutions such as larger blocks, are simply linear stopgaps.

Perhaps the most critical (and interesting) present technical problem for blockchains is to solve the scaling problem while maintaining decentralized trust, and keeping transaction availability latency at an acceptable level.