Adding Trust to CAP: Blockchain as a Strong Eventual Consistency Recovery Strategy

Keir Finlow-Bates

keir@chainfrog.com

September 27, 2017

Abstract

The CAP theorem [3] asserts that in any distributed data store only two out of three guarantees can be provided regarding consistency, availability, and partition tolerance. In commercial systems on the internet, partition tolerance can never be fully guaranteed in practice, limiting the choice to either consistency or availability. Furthermore, as availability has the most significant impact on revenue, ultimately the system design of distributed data stores for businesses reduces to a choice between different eventual consistency recovery strategies. Seen in this light, although blockchain is a strong eventual consistency recovery scheme with reconciliation through a consensus protocol, it is an inefficient and expensive method for achieving such consistency. However, when a fourth overlooked guarantee is added to the CAP mix — namely the question of trust — blockchain comes into its own. In this paper I argue that the significance of blockchain when considering distributed data lies in its provision of a new approach to an extended version of the CAP theorem: CAPT, which asks, "to what level can you guarantee consistency, availability, partition tolerance and *trust*?"

Keywords: blockchain, consensus, CAP, database, consistency, trust

1. Introduction

Historically, distributed data stores have been owned and operated by a single entity such as a corporation, often acting as a middleman in transacting between different external parties. In this situation the question of trust is reduced to a binary choice — either you trust the entity and therefore submit transactions to and retrieve data from their system, or you do not trust them and therefore select another service provider, "roll your own" solution, or

abstain completely. A result of this has been that the issue of trust does not figure in Brewer's CAP theorem. The CAP theorem has been refined over the years to consider consistency and availability as lying on a continuous spectrum [2], and has informed further research into ACID versus BASE [10], but apart from this it has basically remained unchanged.

In [2] Brewer further argues that because partitions in a network are rare, the aim is to ensure "perfect C[onsistency] and A[vailability] most of the time, but when partitions are present or perceived, a strategy that detects partitions and explicitly accounts for them is in order". In summary, partitions in traditional database network architecture are seen as very unusual events.

In a blockchain system the occurrence of partitions is significantly more likely. Blockchain attempts to handle a new emerging case in data storage systems whereby parties that are completely untrusted or only partially trusted are participating in the network on an equal footing. Not only are accidental partitions due to network or node failures a possibility, but a public blockchain in particular has to deal with maliciously generated partitions, chain forks due to competing nodes, and software forks due to community disagreement. Standard fault tolerance methods such as Praxos [7] and Raft [9] are no longer sufficient, and a new strategy is required to resolve these discrepancies.

Blockchain originated from an attempt to solve the "double-spending" problem in the context of a decentralized system [8], in which a digital token recorded on a ledger or in a database can be spent more than once unless suitable consistency approaches are taken. In a centralized system the double-spend issue is not a problem, because the central authority has the final say over the "single view of the truth".

In this paper I examine how blockchain resolves the fourth guarantee of trust and allows for an estimate of such trust on a spectrum, from completely untrusted or unknown and untraceable participants on the network, right up to a fully trusted network of participants. The analysis highlights the observation that ultimately the purpose of blockchain is enabling the sharing of data and transmittal of transactions or states between different stakeholders across low trust boundaries, and that this should inform the development of use cases for blockchain.

2. On trust

When you submit your personal data to a third party for storage, dissemination and retrieval, or delegate your financial transactions to a bank or credit card provider, you are implicitly making a trust decision. Yet hacks such as the Ashley-Madison and Equifax data leaks show that in many cases such trust is unfounded. Similarly, the emergence of Bitcoin as an alternative value transmission protocol emerged as a response to perceived malfeasance by the financial sector. And yet the CAP theorem is notably silent on the issue of trust. This may be in part due to the development and deployment of public, decentralized, and distributed systems being relatively new, and in part due to the fact that third party data storage and transaction systems are closed to inspection. Such "security through obscurity" makes a meaningful assessment of the risk associated with implicit trust impossible until such trust is revealed as unfounded after a breach. As with the initial formulation of the CAP theorem, trust is then a Boolean option – either you trust the system completely ($T = 1$), or you do not ($T = 0$).

Understanding how an estimate of trust can be calculated for a blockchain requires a review of the conditions under which a "single view of the truth" may be lost and subsequently regained, and is the topic of the next section.

3. Consistency, uncles/orphans and forks

There are three changes that affect blockchain consistency dramatically and which are not standard partitioning as experienced by conventional distributed databases: orphan or uncle blocks[1], soft forks, and hard forks. At their simplest, each is an event in which the data chain comprising the blockchain splits and grows along two separate paths, with subsets of the blockchain participants following each path. In a successful blockchain implementation the split should eventually be resolved, with one chain winning

[1]There is some confusion regarding the terminology for accidental chain forks based on duplicate "next block" candidates for extending the chain. The terms "stale blocks" and "orphan blocks" are used interchangeably by some authors. Others reserve orphan blocks to mean blocks for which the parent block has not yet been downloaded during a peer node sync operation. The confusion results from a misuse of the term "orphan" in the original Bitcoin code base. At the time of writing, although the term orphan is a misnomer, it is most commonly used to indicate valid yet non-included blocks. Ethereum uses the term "uncle block" to describe these.

out over the other deterministically.

3.1. Orphan blocks and orphan chains

The blockchain extends over time as successive blocks of data are added to the chain, forming a hash-linked list. A fork due to orphan blocks occurs when there are two equally or nearly equally valid candidates for the next block of data in the blockchain. This event can occur when the two blocks are found close in time, and are submitted to the network at different "ends", so that one subset of nodes receives the first candidate before the second, and the second subset of nodes receives the second candidate before the first. Each subset of nodes continues to attempt to grow the chain from their chosen candidate. Eventually one chain will grow faster than the other, and participants working on the shorter chain will switch, resulting in the more successful chain growing even faster, until all participants have switched to the winning chain. The blocks in the losing chain are then described as "orphaned".

Orphan blocks are generally a natural event in the blockchain ecosystem, and will always be resolved eventually. In Bitcoin, about 0.1% of blocks are orphans [6], and they typically resolve within one further block generation cycle. In Ethereum, orphan or uncle blocks are actually encouraged, and although their transaction content does not form part of the data set, the effort expended in creating them counts towards the proof-of-work hardening of the chain.

However, malicious blocks can also be created, for example in an attempt to attach to an earlier block in the chain. If a malicious block breaks the consensus protocol it will be rejected by the rest of the network, but a valid malicious block may intentionally be created earlier in the chain in order to create a new fork that benefits the attacker. In this circumstance the blockchain has to rely on the majority of the network being benign and ignoring the new chain in favor of the old.

3.2. Soft forks

Soft forks occur when part of the network agrees to alter the blockchain protocols in a backward incompatible manner that places further restrictions on what constitutes a valid block. Under a soft fork, participants who refuse or fail to upgrade to the new protocol will still accept blocks that are created by the new updated software. However, they may continue to submit blocks that are no longer accepted by the participants who have upgraded. If the

majority of the network has accepted the soft fork, the laggards may end up mining blocks that will not make it into the chain, which is a waste of their computing power and hence the financial rewards that public blockchains typically award to block-creators. This provides an incentive for the participants to upgrade. Soft forks allow for a gradual upgrade of the nodes operating on the network, provided a majority accepts the changes when they are released. They are the preferred method for improving the underlying blockchain mechanisms because they usually result in less chaos.

Technically the new software could reject the validity of the existing chain up to the fork point, if not for the fact that the software normally has a marker indicating when the fork occurred and uses the old protocols for validation up to that point.

3.3. Hard forks

The most difficult change to adopt is the hard fork, under which the majority of the nodes agree to alter the protocols in a backward compatible manner, by extending the protocol. Under a hard fork, future blocks will be rejected by nodes that do not upgrade. Hard forks are the most likely to generate problems in the blockchain ecosystem and have resulted in blockchain systems splitting in two permanently, for example Bitcoin Cash (BCH) versus Bitcoin Classic (BTC) [12].

An example of an unintended hard fork (which subsequently had to be reverted) was the update of Bitcoin from 0.7 to 0.8 in March 2013 [1], in which the underlying database was changed from Berkeley DB to LevelDB. Unbeknownst to the developers, version 0.8 with LevelDB was able to handle the inclusion of more transactions into a block than version 0.7 was. Unfortunately the split between 0.7 and 0.8 miners on the network was approximately 40/60, and as a result two chains developed. The final orphaned chain managed to reach a length of 24 blocks before the fork was resolved. This remains the longest orphan chain in Bitcoin history to date.

4. Calculating trust

Once the causes of orphan blocks and chains have been categorized, it becomes possible to derive a rule to estimate an actual probability of a transaction becoming part of the globally accepted truth on the blockchain. In Bitcoin this is simplified to a rule of thumb — 6 blocks further on (also known as confirmations) the transaction is effectively considered permanent.

As a result, software and users can make a determination whether to submit further transactions based on the number of confirmations and the importance assigned to the initial transaction. In Bitcoin's case, this boils down to "accept a coin transaction as confirmed after one or even no confirmations if it concerns a small sum, e.g. the price of a cup of coffee, but when buying a car wait for 6 blocks, and for buying a house wait for 20 or 30 blocks".

4.1. Empirical estimates

Calculating the probability of an orphan chain in Bitcoin is difficult, because different nodes keep different records of the blockchain, and when a new node joins, the old nodes only forward what they consider to be the blocks comprising the chain at that time. Furthermore, it is not possible to estimate what the odds of an unusual fork event will be, so an empirical estimate of the likelihood of orphan blocks can only be made for normal operation of the blockchain. However, some statistics are available from a variety of sources. With this data it is possible to parse back over forks to determine the probability that your transaction is included in what will ultimately be an orphan block. For example, Figure 1 below shows the number of orphan blocks found each month for a three-year period in Bitcoin's recent history.

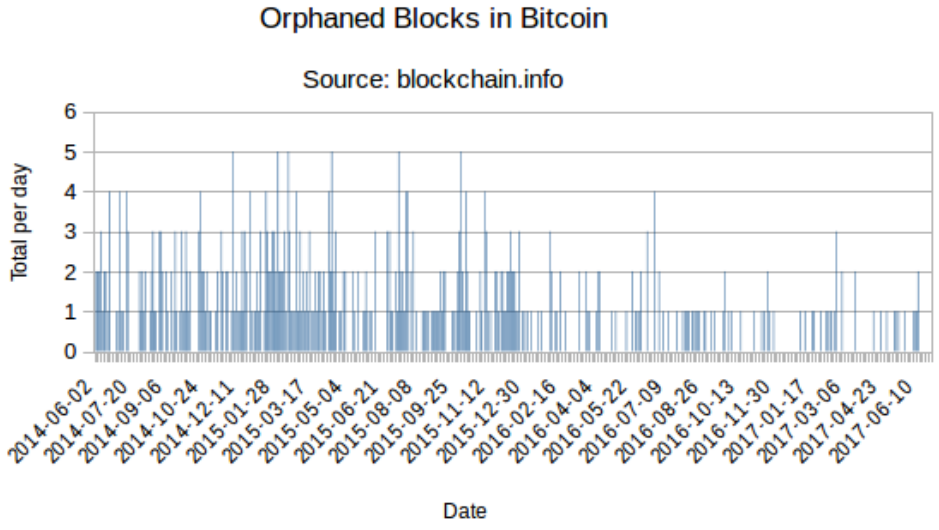

Figure 1: Recent orphan block counts in Bitcoin

Over this period of 1125 days, approximately 162000 blocks were added to the Bitcoin blockchain. 485 orphan blocks were mined, for an average of 0.431 orphans per day. The maximum number of orphans in one day was 5. Unfortunately this data does not tell us the length of orphan chains during the period. A rough estimate of the odds of your transaction ending up in an orphan block is:

$$\frac{485}{(162000 + 485)} \approx 0.00298$$

This is approximately 0.3%. The transaction could of course be included in both the orphan block and the successful block at the same time. Furthermore, technically Bitcoin transactions do not have an expiry date, so they should eventually be included in a mainline block.

Another approach is to use the Bitcoin core software command line option `printblocktree`, which produces a text file output that shows the main blockchain line and forks. This only works on a system which has been running without interruption over the period being examined. Records on the internet are patchy at best, but Decker [4] at StackExchange presents some data for blocks between 90392 and 189512, representing a period of 991200 blocks, or about 688 days of blockchain activity as observed by Michael Marquardt.

Fork length	Number of occurrences
1	87
2	4
3	1
4	1

Table 1: Forks observed on one Bitcoin node — 90392 to 189512

Using these figures, we end up with the odds of a transaction ending up in an orphan block at:

$$\frac{93}{(99120 + 93)} \approx 0.00094$$

At about 0.1% these odds match the figure quoted in Kernfeld [6]. Although lower than our first estimate, the order of magnitude is still the same, and the calculation reveals that blockchain nodes at the edge of the network

with fewer peer connections will return different results from central nodes with many peer connections. Unfortunately both Kernfeld and our estimate are an order of magnitude smaller than Decker *et al.* [5]. Why this should be is not clear.

4.2. Theoretical estimates

A more generalized approach for a proof-of-work backed blockchain such as Bitcoin can be taken by assuming a simple model of the network, represented by a regular isometric graph (except at the boundary), with an equal latency of l for each edge and n edges per node. We can propose a function such that after t seconds, miners controlling a proportion $\alpha(t)$ of the mining power will have received any new valid block, up to some point in time t_{max}, after which all miners will have received it. That is, $\alpha(t) = 1$ for all $t \geqslant t_{max}$. As the number of nodes that have received a block notification after k latency periods is $n \sum_{x=1}^{k} x = n\frac{k(k+1)}{2}$ we can see that the proportion should grow quadratically. The following is therefore an acceptable approximation for the block propagation function for a system with many nodes:

$$\alpha(t) = \begin{cases} \frac{t^2}{(t_{max})^2} & 0 \leqslant t < t_{max} \\ 1 & t \geqslant t_{max} \end{cases}$$

Proof-of-work based blockchains adjust the difficulty of finding a new block in proportion to the network hash rate, so that the blocks are found at a constant average rate. For example, mean time between blocks for Bitcoin is $\mu = 600s$.

As a result, finding blocks is a Poisson process, and the probability distribution that describes the time between events is an exponential distribution. In our case, given that a block has just been found, we need to calculate the probability that at least one other miner that has not received notification of the first block finds a block before time t_{max} (at which point all miners know of the first block). This means that the hashing power dedicated to finding a block that will end up being an orphan is in proportion to $1 - \alpha(t)$ during the time period $0 \leqslant t < t_{max}$.

The probability that a block will be found within t_{max} is given by the cumulative distribution function for the exponential distribution, $1 - e^{-\lambda x}$, where $x = t_{max}$ and $\lambda = \frac{1}{\mu} = \frac{1}{600}$.

To find out the proportion of hash power spent on finding an orphan, we evaluate the following:

$$\int_0^{t_{max}} 1 - \frac{t^2}{t_{max}^2} dt = \frac{2}{3} t_{max}$$

So the proportion is $\frac{2}{3}$ after the ratio is normalized.

Finally, from Decker *et al.* [5] we can obtain figures estimating t_{max} for the Bitcoin network. Decker *et al.* quote a mean block arrival time for a general node of 12.6s. For our hashing power proportion function, this means that:

$$t_{max} = \sqrt[3]{2 \times 12.6^3} \approx 15.875$$

Therefore, putting it all together, with our model the estimated percentage of orphan blocks is:

$$\frac{2}{3} \times (1 - e^{\frac{-15.875}{600}}) \approx 0.01741$$

At 1.741% our simple model produces a value very close to Decker *et al.*'s empirically observed value of 1.69%.

5. Summary

Although private and public blockchain systems face different issues in building consensus, they do have some features in common. Both public and private blockchains will face a consistency problem due to network latency, which needs to be overcome by the blockchain consensus protocol. As demonstrated above, the probability of a consensus discrepancy and the expected resolution time can be calculated empirically or theoretically with a reasonable level of accuracy.

Public blockchains also suffer from a significant level of risk due to malicious attackers. Although it is possible to calculate the chances of success given knowledge of an attackers resources [11], it is not possible to provide an estimate of the probability that someone will actually mount an attack against a given public blockchain.

Private blockchains are less likely to suffer an attack, as the participants can be clearly identified, and due to the audit trail left by the tamper proof record of the blockchain any damaging activity can be attributed to the responsible party, who can then be expelled from the blockchain system. The damage can subsequently be corrected.

Unlike traditional centralized data storage solutions, blockchain allows for the sharing, editing and creation of data between different entities across

lower trust boundaries, and under certain circumstances it is possible to produce a measure of the likelihood of and extent to which the system will be inconsistent, together with an estimated resolution time. Although no deterministic measure of the level of trust within the system can be provided, it should be noted that in the CAP theorem there is similarly no accurate determination of the level of consistency or availability provided; nor is the probability of partitioning within the system estimated.

6. Conclusion

I have proposed an extension to the traditional CAP theorem by adding an extra consideration, namely the issue of *trust*. When a decentralized data storage system is considered, in particular an open system where anyone may join as a node and participants are pseudo-anonymous, the standard CAP theorem model breaks down. A measure of the level of trust engendered is then required in order to provide a satisfactory balanced solution. Blockchain, through the use of consensus protocols, a time stamped hash-linked list of data blocks, and a consensus protocol such as "proof of work" with a history of transactions and blocks added to the chain provides such a quantifiable system with a probabilistic measure of the consistency of the system at any given point in time, both through empirical and theoretical methods.

Further issues remain, primarily due to the performance reduction caused by the increased workload on the system network due to block generation and validation. The provision of methods for the correction and rollback of databases attached to and reading from a blockchain data stream should therefore be of interest. As the trade-off in the original CAP theorem between consistency and availability and the subsequent emergence of NoSQL databases systems were due to significant increases in the quantity of data needing to be stored and processed, a secondary area for further research concerns the integration of blockchain with big data.

References

[1] Andresen, G., *March 2013 Chain Fork Post-Mortem*, 2013, retrieved on 23 Sept 2017 from https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki

[2] Brewer, E., IETF RFC5952, *CAP Twelve Years Later: How the "Rules" Have Changed*, 2012, retrieved on 21 Sept 2017 from https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

[3] Brewer, E., *Toward Robust Distributed Systems*, 2000, Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC 00), ACM, pp. 7-10

[4] Decker, C., *StackExchange Answer*, 2012, retrieved on 23 Sept 2017 from https://bitcoin.stackexchange.com/questions/3343/what-is-the-longest-blockchain-fork-that-has-been-orphaned-to-date

[5] Decker, C. and Wattenhofer, R., *Information Propagation in the Bitcoin Network*, 2013, Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing, retrieved on 26 Sept 2017 from http://www.tik.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf

[6] Kernfeld, P., *How Bitcoin Loses to the CAP Theorem*, 201, retrieved on 23 Sept 2017 from http://paulkernfeld.com/2016/01/15/bitcoin-cap-theorem.html

[7] Lamport, L., *Paxos Made Simple*, 2001, retrieved on 23 Sept 2016 from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/paxos-simple-Copy.pdf

[8] Nakamoto, S., *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008, retrieved on 12 Oct 2016 from https://bitcoin.org/bitcoin.pdf

[9] Ongaro, D. and Ousterhout, J., *In Search of an Understandable Consensus Algorithm*, 2014, Proceedings of USENIX ATC 14, retrieved on 23 Sept 2016 from https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14

[10] Roe, C., *ACID vs. BASE: The Shifting pH of Database Transaction Processing*, 2012, retrieved on 21 Sept 2017 from http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/

[11] Rosenfeld, M., *Analysis of hashrate-based double-spending*, 2012, retrieved on 25 Sept 2017 from https://arxiv.org/pdf/1402.2009.pdf

[12] Smith, J., *The Bitcoin Cash Hard Fork Will Show Us Which Coin Is Best*, 2017, in Fortune Magazine, retrieved on 23 Sept 2017 from http://fortune.com/2017/08/11/bitcoin-cash-hard-fork-price-date-why/