What's so Different about Blockchain? – Blockchain is a Probabilistic State Machine –

Kenji Saito and Hiroyuki Yamada
Orb, Inc.
Sumitomo Bldg. 25F, 2-6-1 Nishi-Shinjuku, Shinjuku, Tokyo 163-0225, Japan
Email: {kenji | hiro}@imagine-orb.com[1]

***Abstract*—Blockchain is a distributed timestamp server technology introduced for realization of *Bitcoin*, a digital cash system. It has been attracting much attention especially in the areas of financial and legal applications. But such applications would fail if they are designed without knowledge of the fundamental differences in blockchain from existing technology.**

We show that blockchain is a probabilistic state machine in which participants can never commit on decisions; we also show that this probabilistic nature is necessarily deduced from the condition where the number of participants remains unknown. This work provides useful abstractions to think about blockchain, and raises discussion for promoting the better use of the technology.

I. Introduction

Blockchain is a distributed timestamp server technology introduced for realization of *Bitcoin*[1], a P2P (peer-to-peer) digital cash system. Blockchain implements a *ledger* that validates existence of *digital assets* (e.g. coins, as in the case of Bitcoin), and tracks where the *control* exists for each of them in the network. Such control is totally distributed, and the one (or ones) who has the control can change the state of the asset (e.g. its controllership) without permissions from any central authority.

Blockchain has recently been attracting much attention from industries, especially in financial and legal sectors. For example, Nasdaq has introduced *Nasdaq Linq*[2], a blockchain technology to manage private securities transactions since 2015. A venture named R3CEV[3] has established a consortium of international banks since 2015 that has begun experimenting on blockchain to manage inter-bank transactions.

Meanwhile, fundamental properties of the blockchain technology have never really been questioned. To the best of our knowledge, this paper is the first work to ask and give answers to such questions.

Our contributions are summarized as follows:

1) We show that blockchain is a *probabilistic state machine*, and is abstracted as a *consensus machine with a cost register*. With a proof that there exists no protocol that can reach consensus if the number of participants n is unknown, we show that blockchain is necessarily probabilistic, and consensus is never actually reached under its assumed use conditions. We also show that blockchain does not tolerate network partitioning of unbounded duration.

[1]Kenji Saito is also reached by ks91@sfc.wide.ad.jp

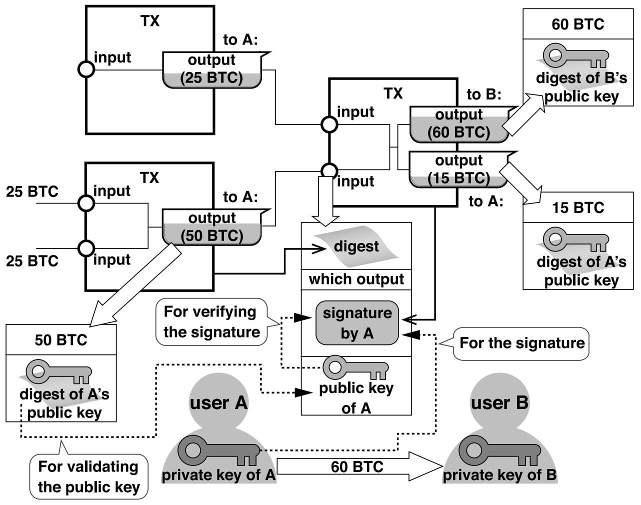


Fig. 1. TX Data Structure (of Bitcoin)

2) We discuss recommended use conditions for both blockchain and Byzantine Paxos[4].

II. Background and Definitions

A. Blockchain

In an abstracted protocol of blockchain, exchanged messages are either *TX*[2], *block* or *membership*. A TX is a command to change the state of an asset. A block is a collection of TXs. A membership is a collection of participants the sender of the message knows.

Any participant may construct and broadcast (by flooding) a TX on assets of which they have control. Fig.1 shows an example of TX data structure found in Bitcoin. A TX may have a list of inputs each referring to an existing asset, and a non-empty list of outputs each representing a new asset or existing asset with a new state. A TX is digitally signed in each input, which is verifiable with the public key also included in the input. The digest of the public key needs to match the digest to which the referred output is addressed. This structure is self-contained, and its validity can be verified by anyone.

[2]It stands for *transaction*. However, since its semantics is different from that of *transaction* in database[5] (described in section II-D), we use the term *TX* to avoid confusions.

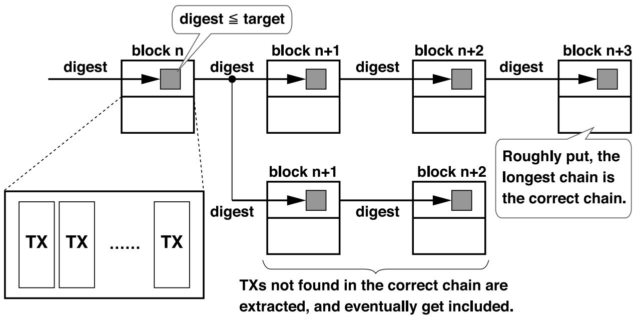

digest ≤ target
digest
block n
digest
block n+1
digest
block n+2
digest
block n+3
TX
TX
......
TX
block n+1
block n+2
digest
digest
Roughly put, the longest chain is the correct chain.
TXs not found in the correct chain are extracted, and eventually get included.

Fig. 2. Blockchain

The objective of the blockchain's consensus algorithm (*blockchain consensus* hereafter) is to maintain a single history of blocks such that it does not involve any invalid or contradicting TXs. An example of the latter is double spending of one digital coin, which cannot be prevented by the TX data structure alone. Presumably, the designer (or designers) of this technology first called it *distributed timestamp server* because it is intended to keep track of relative timing of events.

As Fig.2 shows, each block contains the cryptographic *digest* of the previous block, except the very first block sometimes called the *genesis block*. Such a digest must meet a certain criterion; it needs to be less than or equal to the pre-adjusted and agreed *target*[3] stored in or calculated from[4] the block (we call this structure *digest chain* hereafter).

Since the digest is calculated by a one-way function whose outputs are evenly distributed, no one can intentionally configure a block to satisfy the criterion. Instead, they need to partake repetitive trials to change the values of some *nonce* in the block they are creating until they get a right digest. Therefore, creation of a block is a probabilistic process.

Participants take part in maintaining the chain of blocks as follows:

1) A participant collects TXs as they receive them, and tries to create a valid block that contains them to add to the tail of the digest chain.
2) If a participant is successful, then it broadcasts the created block.
3) Other participants receive the block, and try to validate it. If it is successfully validated, they admit that it is the new last block in the digest chain, and continue to 1.

The necessity of repetitive trials functions as a *proof of work* mechanism to limit the number of proposed blocks at one time. But there still is a possibility of multiple participants each proposing a new block at roughly the same time, which may be accepted by different sets of participants. Then the digest chain may have multiple ends that are extended independently from one another, resulting in a *fork* of the blockchain with multiple (and possibly, contradicting) histories of blocks. If this happens, roughly speaking, the longest branch is considered to be correct. More precisely, to avoid the case of branches with artificially raised targets, the branch that is the most difficult to produce is chosen by all participants. This reflects the total cost cast in the creation of the digest chain.

The proof of work is also intended to be a protection against falsification. A TX itself cannot be falsified unless digital signatures are compromised. But it is conceivable to remove some TXs from a block. If one tries so, the digest of the block is changed and is typically greater than the target. Then they would have to retry the proof of work for the block. This changes the digest stored in the next block, which in turn means that the digest of the next block is also changed and is typically greater than the target, and so on. In short, ones with a malicious intention would have to redo the proof of work from where they want to change, and outdo the ongoing process of adding blocks eventually to make the change valid, which has generally been considered highly difficult[5]

B. Consensus Problem

The consensus problem is for a set of (not necessarily reliable) participants to choose a single value. The problem is expressed in terms of three different roles, where one participant may take multiple of these roles: *proposers* who can propose values, *acceptors* who choose a single value among the proposed ones, and *learners* who learn the chosen value. A participant is said to be *faulty* if it does not follow the protocol in some way. Otherwise, it is said to be *correct*. Then the consensus problem is formulated as a set of *safety*[6] and *liveness*[7] properties as quoted from [6] below (*CS* for Consensus Safety and *CL* for Consensus Liveness).

CS1 Only a value that has been proposed may be chosen.
CS2 Only a single value may be chosen.
CS3 Only a chosen value may be learned by a correct learner.
CL1 Some proposed value is eventually chosen.
CL2 Once a value is chosen, correct learners eventually learn it.

A faulty participant may be *benignly faulty*, in which case they just stop or omit messages, or *Byzantine* (see section II-C), in which case their actions may be deliberately harmful.

It has already been proven that it is impossible for any protocol to reach consensus in an asynchronous system (i.e. no synchronized clocks) with just one benignly faulty participant[7]. Therefore, any practical solutions for consensus need to introduce some level of synchrony in the system.

C. Byzantine Generals Problem and Its Extension with Unknown Participants

The *Byzantine Generals Problem*[8] is to reach consensus without any assumptions about failures; participants may stop,

[3]In Bitcoin, the target is global and stored in blocks, usually inherited from the previous block and updated at every 2016th block by the shared algorithm.

[4]For example, in *proof of stake*, the private target for each participant is calculated in proportion to the amount of digital assets of which they currently have control in the system.

[5]However, it is a matter of cost, and is physically possible.

[6]A safety property ensures that something bad never happens.

[7]A liveness property ensures that something good will eventually happen.

omit messages, deliver them in a wrong order, lie, or collude. But originally, the problem assumed a static network of known participants.

BFT-CUP[9] (Byzantine Fault Tolerance on Consensus with Unknown Participants) has been proposed as an extension of Byzantine Generals Problem to seek solutions in a network of unknown participants, such as ad-hoc, P2P or dynamic network. But [9] assumes the expected maximum number of faulty participants f, unique identifiers for participants, and secure direct connections among known participants. These assumptions may be unrealistic for real P2P applications.

D. Transaction

A transaction[5] is an atomic set of operations that is either committed or aborted. In a distributed system, the decision must be agreed by all participants. Therefore, a distributed transaction requires a consensus protocol to choose between two values: *commit* or *abort*. Once a choice is made on a transaction, the choice persists forever, and it may never be undone.

Since there will be one and only one history of all committed transactions, if the initial state is shared among all replicated services, the replicas must be maintained exactly the same among one another. In this case, replicas are said to be *consistent* with one another.

More precisely, *consistency* is defined as when an update is made, all observers would see that update[10] (from any of the replicas). [10] further defines the following two important concepts on consistency:

Strong consistency

After the update completes, any subsequent access will return the updated value.

Eventual consistency[8]

If no new updates are made to the object, eventually all accesses will return the last updated value.

III. Problem Statement

A. The Problem

We are to understand the essential properties of blockchain that make it different from existing technology for reaching consensus and managing ledgers, such as Paxos[11] or its *byzantized* versions[4][6][12][13].

B. Blockchain Consensus in Context of Consensus Problem

Under blockchain consensus, a proposed value is a block. Since a block contains a pointer to the previous block, the proposed value is effectively a history of blocks from the current one all the way back to the genesis block.

Participants who take part in validation of blocks are acceptors[9] under blockchain consensus. Any acceptors can be a proposer. The proof of work can be seen as a *leader election* process where a leader is a proposer, which may result in multiple leaders. In reality, acceptors may be just a subset of all participants, as participating in the leader election would involve intensive calculations. All participants are learners.

[8]Eventual consistency is a strengthened version of *weak consistency*. Weak consistency does not guarantee that subsequent accesses will return the updated value.

[9]They are called *miners* in the terminology of Bitcoin.

C. Preconditions

We presume that the designer (or designers) of blockchain has intended to solve the problem below. Whether this intention is successful or not is discussed in section IV. Note that the term *blockchain* in this work is inseparably associated with the conditions described below, under which, we believe, the technology is designed to work.

We formulate our version of BFT-CUP problem as follows, partially borrowing notations from [9]. Consensus is to be reached among correct participants $\Pi^c \subseteq \Pi$, where Π is a finite set of n participants. A participant $i \in \Pi$ is only aware of a subset $\Pi_i \subseteq \Pi$. Participants are subject to arbitrary failures, strategies or collusions that make them deviate from the protocol. The number of faulty participants f is unbounded, and unknown to all participants. The participants form a network $< \Pi, E >$ where E is the set of edges of the network such that $E \subseteq \Pi \times \Pi$. We define that E is a reachability relation, where iEj means that there is an underlying path between participants i and j so that a message can be reached from i to j. It is not assumed that the relation is either symmetric or transitive; iEj does not always imply jEi (because of unidirectional links or faulty participants), and $iEj \wedge jEk$ does not always imply iEk (because j may be faulty and does not forward messages to k; E *is* transitive among correct participants). A participant i may send a message to another participant j only if $j \in \Pi_i$ and iEj. A participant i occasionally receive messages from some $j \in \Pi_i$ that contain the membership of Π_j, so that Π_i may be updated during the execution of the protocol. Π_i *is* updated when i receives a message from j such that $j \notin \Pi_i$, so that $j \in \Pi_i$.

Time is imperfectly synchronized among participants. There is no assumption on the relative speed of participants' processing messages or on message transfer delays. A participant i joining the network is abstracted as Π_i containing just the initial contacts with which i is beginning to exchange messages. A participant i leaving the network is indistinguishable with i being temporarily unreachable from any other participants.

IV. What's so Different about Blockchain?

A. What to Think About

We see that blockchain consists of loosely-coupled three layers of different technology:

1) TX data structure
   This self-contained structure can be verified by anyone, and is useful for constructing a system maintained by unspecified participants. It can be used under other consensus mechanisms.
2) Digest chain
   This structure gives the basis of the ordering of events in the system. It also provides means for leader election, but the subsequent process can be of some other consensus mechanism.

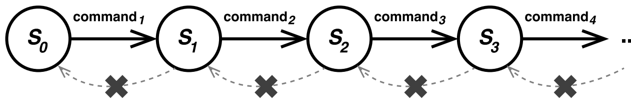


Fig. 3. Deterministic State Machine

3) Blockchain consensus
   This utilizes the digest chain, but can be discussed independently from other layers.

In this work, we focus on the blockchain consensus layer. In particular, we focus on the conditions under which it is supposed to work, and the extent to which it works in reality. There are some economic implications of blockchain consensus, which is out of scope of this work. Readers are referred to [14] if they are interested.

B. Deterministic State Machine

We apply state machine approach[15] to understand blockchain consensus. First, we observe the state machine of existing database replication technique for later comparison, as illustrated in Fig.3. Its deterministic state transition is stable once a command is committed in the system despite some faulty participants may exist. This is achieved by Byzantine Paxos for example, under the optimal condition of $n > 3f$ with Byzantine participants[4]. It is assumed that n is known and f can be bounded somehow to satisfy the condition.

C. Blockchain is a Probabilistic State Machine

We now show that blockchain consensus is a probabilistic state machine as illustrated in Fig.4.

Each participant has their private history of accepted blocks. By realizing that such a private history is different from a more probably converged history (up to block$_5$ in Fig.4), the effects of some recently accepted blocks (block$_{2'}$ and block$_{3'}$ in Fig.4) are reverted to follow the probably converged history. Such a realization is brought by receipt of a new block (block$_5$ for example, which in turn brings block$_{4,3,2}$, in Fig.4).

The converged history can only be finalized when the system stops completely, but is probabilistically stable. The converged history can eventually represent a run of a replicated database like Fig.3, at an unbounded future time. Fig.5 shows state transitions of an asset in blockchain, where all states could possibly be reverted.

D. Impossibility of Blockchain Consensus - Brief Explanation

We will give a proof sketch that a consensus is never really reached using blockchain. Since n is unknown, a value cannot be chosen by voting in our consensus mechanism. Instead, we utilize the cost for casting some amount of resources[10]. We abstract this consensus mechanism as a machine in Fig.6.

This machine is further abstracted as follows, without changing its semantics: it is a variable whose value can only

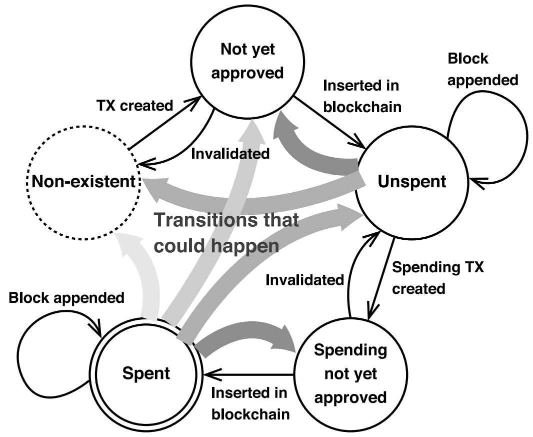
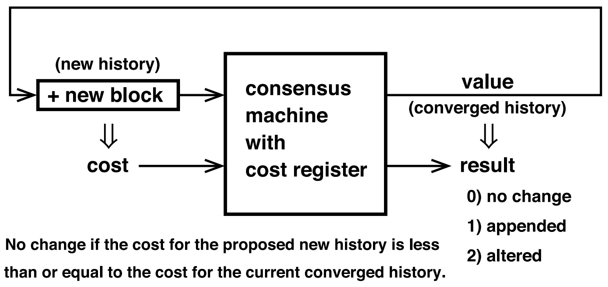


Fig. 5. State Transitions of an Asset in Blockchain (e.g. Coin)



(new history)
+ new block
cost
consensus machine with cost register
value
(converged history)
result
0) no change
1) appended
2) altered
- No change if the cost for the proposed new history is less than or equal to the cost for the current converged history.

Fig. 6. Abstracted Blockchain Consensus Machine

be changed by paying more cost than the cost c paid to decide the previous value v.

Now, suppose all known correct participants together change the value to v', collectively paying c' where $c' > c$. Even if all unknown participants are correct, any participants cannot throw away the possibility that there appears a set of unknown faulty participants who can collectively pay c'' such that $c'' > c'$ to change the value to v''.

Therefore value v' is never considered determined. Consensus is never reached using blockchain because a value is never actually *chosen*, and none of safety (CS1$\sim$3) or liveness (CL1$\sim$2) properties of the consensus problem is satisfied.

E. Impossibility of Blockchain Consensus

Now, we will clarify under which conditions blockchain consensus can work to more formally explain the impossibility.

Let R_i be the amount of resources cast by participant i for creating a block. R_i is 0 if i is benignly faulty (being stopped or omitting messages). We assume that correct participants will eventually participate as acceptors.

We define

$$R = \sum_{i=1}^{n} R_i, \quad F = \sum_{i=1}^{f} R_i$$

where R is the cast resources in total, and F is the sum of cast resources by faulty participants.

[10]Economic measures such as proof of work, proof of stake, etc. all ensure that ones who cast more resources are more likely to be able to create a block, and can be abstracted by the same model as proof of work.

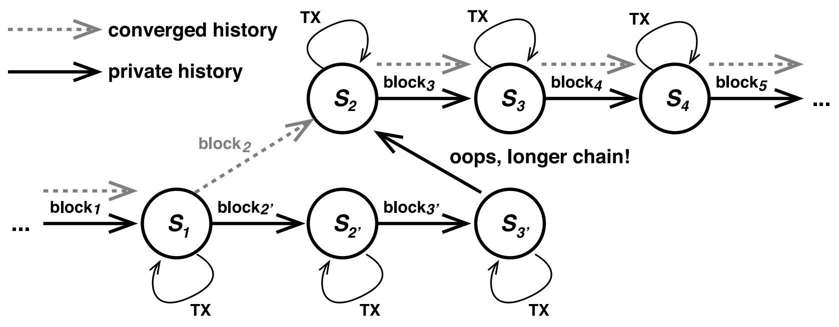


Fig. 4. Probabilistic State Machine of a Blockchain

Next, we define correct and malicious blocks. A correct block is a block created by a correct participant. A malicious block is a block created by a malicious participant, with some TX excluded intentionally. We assume that a correct participant cannot detect a malicious block.

We then define correct and malicious (converged) histories. A correct (converged) history is one that consists entirely of correct blocks. A malicious (converged) history contains at least one malicious block.

We give conditions for blockchain consensus to produce a single correct converged history by proving the following theorem.

Theorem 1: Blockchain consensus is not guaranteed to produce a single correct converged history unless the following conditions both hold:

1) $R > 2F$
This means that the sum of cast resources by correct participants is greater than that by faulty participants.
2) $\Pi^c \times \Pi^c \subseteq E$
This means that all correct participants are reachable from one another.

Proof: We consider each of the conditions above to show that production of a single correct converged history is not guaranteed if one of them is negated:

1) $R > 2F$
If this condition is negated, we get $R \leq 2F$ (the sum of cast resources by faulty participants is greater than or equal to that by correct participants). Then correct block-creation can be outpaced by collective block-creation by faulty participants.
2) $\Pi^c \times \Pi^c \subseteq E$
If this condition is negated, then there exist some $i, j \in \Pi^c$ such that $\neg\, iEj$ (j is not reachable from i). Because E is transitive for $\Pi^c \times \Pi^c$, this also implies that for all $k \in \Pi^c$ such that iEk, $\neg\, kEj$ (j is unreachable from all correct participants reachable from i). Fig.7 illustrates an example of such a situation. In this diagram, a faulty participant m denies forwarding messages from i to j by blocking all paths that may reach from i to j, separating the network into partitions 1 and 2. Then blocks created in partition 1 would never reach partition 2, resulting in two separate correct histories, one produced in partition 1 (excluding contradicting blocks from partition 2) and another in partition 2 (excluding any blocks from partition 1). Convergence would not correctly work as blocks are not fully propagated.

■

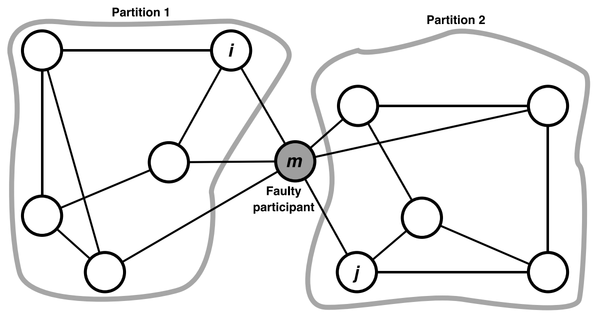


Fig. 7. Network Partitioning / Eclipse Attack

Theorem 2: The number of participants n is never determined under blockchain consensus.

Proof: For any $i \in \Pi$ to determine $n = |\Pi|$, i needs to know all $j \in \Pi$ so that $\Pi_i = \Pi$ and count $|\Pi_i|$. However, there may be some unknown number of participants who have not joined the network, whom i does not know, or the network may be partitioned, and some membership information is never conveyed to i. Therefore, i can never be certain that it knows all $j \in \Pi$. ■

We now show that consensus is never reached with blockchain consensus. We do it by showing that participants can never know whether a consensus can be reached or not.

Theorem 3: Participants cannot determine whether they can produce a single correct converged history or not under blockchain consensus.

Proof: A participant at least has to know whether both 1) and 2) in Theorem 1 hold or not to determine if they can produce a single correct converged history or not. We show that both conditions cannot be determined:

1) $R > 2F$
Neither R nor F can be calculated without known n or bounded f. R may be inferred unless the network is partitioned, using the average block creation interval and the target. But F cannot.

2) $\Pi^c \times \Pi^c \subseteq E$
A participant cannot test iEj for any $i, j \in \Pi$ where $i \neq j$. If j never receives a message from i, still iEj is not denied, because if $i \notin \Pi_j$, possibly i has not joined the network, and otherwise, possibly i is either stopped or left the network. A participant cannot test $i \in \Pi^c$ anyway, because one does not know whether participant i is correct or not.

■

F. Impossibility with Unknown Participants

To obtain more general result, we now show that the probabilistic nature is inherent with the conditions under which blockchain is designed to work.

Lemma 1: There exists no protocol that can reach consensus unless f is bounded.

Proof: This trivially holds because of the following optimal conditions for consensus protocols under more strengthened preconditions:

- $n > 3f$ for Byzantine participants.
- $n > 2f$ for benignly faulty participants.

■

Lemma 2: f cannot be bounded unless n is determined.

Proof: We represent f with the following expression:

$$f = g(n) + h(E, \ldots)$$

where g is a function of n that determines the probabilistic part of f, and h is a function of E, etc., which represents the part of f that is not probabilistically decided. Therefore, f is never obtained if n is not determined. ■

This reflects the reality that f is typically chosen based on statistical (thus probabilistic) measures of component reliability[15].

Theorem 4: There exists no protocol that can reach consensus if the number of participants n is unknown even when all participants are actually correct.

Proof: By applying lemma 2, f cannot be bounded under the given conditions. Since f cannot be bounded, by applying lemma 1, there exists no protocol that can reach consensus. All participants may still be correct, because f is just the maximum number of faulty participants. ■

This holds even if we exclude Byzantine participants from the preconditions. It is not contradicting with the result from the previous section, because participant m in Fig.7 can be benignly faulty, either being stopped or omitting all messages, and still can separate the network into partitions to prevent production of a single correct converged history.

Therefore, the probabilistic nature naturally follows from the condition under which blockchain is designed to work: the number of participants is unknown.

V. Discussion

A. Use Conditions : Comparison with Byzantine Paxos

Table I shows a comparison between blockchain and Byzantine Paxos. They use different methods to try to reach consensus, claim to work under different conditions, and have different applicabilities.

1) Blockchain: Note that $R > 2F$ is safe only if n is large. The reason why we consider that blockchain is useful if n is large *and* unknown, instead of n is large *or* unknown, is as follows: if n is small and unknown, the resources cast by one participant has a greater impact, and there is more risk of decisions getting reverted.

An eclipse attack is to cause network partitioning as illustrated in Fig.7 intentionally, so that messages from one partition do not reach others. It is a well-known idea in P2P security research, because in P2P, participants have control of which messages they actually forward to others, and can use such control with a rational or malicious intention to deviate from the correct protocol. Applying this attack against blockchain has been studied in [16] for the case of Bitcoin for example.

An eclipse attack can cause network partitioning of unbounded duration that cannot be tolerated by blockchain. Because n is not known and there is no complete membership information, partitioning cannot be detected, and a partitioned network does not stall, continuing to extend the chain independently from one another (maintains availability but loses consistency). Conversion starts only after the partitions are resolved, because it requires receipt of a block from an unknown chain. This would never happen if the partitioning is kept for unbounded duration.

A *softer* version of partitioning, or a fork in a chain, is often experienced in blockchain because of propagation delay of blocks, which is extensively studied in [17]. For the same reason above, consistency among replicas is not maintained with presence of a fork.

Blockchain is not suitable where finality of decisions is needed (e.g. financial infrastructures or applications to control physical processes), because no decision is committed or final.

Recommended use of blockchain is where n is large and unknown, the number of faulty participants f is unbounded, and state transitions are allowed to be probabilistic (e.g. financial applications in which the operators can take risks of infrequent inconsistencies).

2) Byzantine Paxos: We chose $n > 3f$ as the optimal condition with respect to the largest f, instead of to the smallest number of protocol steps, in which case f needs to be smaller. In any case, it is useful if f is bounded.

Deciding on the upper limit of f is relatively easy if failures are just handled as probabilistic events. But a malicious intention is not about probability, and if there is commitment, such an intention can harm the system with certainty.

Therefore, membership needs to be managed for two reasons under Byzantine Paxos:

1) To maintain the correct n.
This is usually done by applying the consensus protocol itself to update the membership.
2) To maintain the justifiedness of estimation of f.
Operators of the system may need to filter out malicious participants from joining the network.

TABLE I
USE CONDITIONS

	Blockchain	**Byzantine Paxos**
Method	Based on the sum of resources R	Based on the number of participants n
Conditions	$R > 2F$ where R is unknown	$n > 3f$ where n is known
Useful If	• n is large and unknown • f is unbounded • Allows probabilistic state transitions	• f is bounded
Note	Must tolerate eclipse attacks	Must manage membership

VI. Related Work

A. Abstraction of Blockchain

There are not many existing work that give formal representations of blockchain. The *Ethereum* yellow paper[18] is an attempt to represent Ethereum blockchain as a state machine, but it does not explicitly mention the probabilistic nature of the state transitions.

B. Consensus with Unknown Participants

Existing work on CUP[9][19] give consensus algorithms with unknown participants, but deal with the number of participants that can be determined in the first phases of the protocols. At their later phases, existing (Byzantine fault tolerant) consensus techniques are used. These work seem to be trials to solve the membership management problem rather than consensus itself.

C. Consensus with Known Participants

Since popularization of blockchain, existing membership-based consensus methodologies have been reprising in light of a new context of financial and legal applications. Here are just some examples.

Hyperledger Project[20] is a collaborative project at The Linux Foundation for a cross-industry open standard for distributed ledgers. At least its proposed contribution from Digital Asset Holdings seems to be a straightforward application of practical Byzantine fault tolerant protocols[12][13].

Ripple[21] is a cross-currency payment network, and its ledger system has also been proposed in Hyperledger Project. It utilizes a group of validators trusted not to collude with one another, chosen by each participant.

Effects of network partitioning for these work may need to be studied extensively, as existing work on consensus with known participants usually assume direct one-to-one connections among correct participants that may be lost by just a benign failure of a forwarding (underlying) component.

VII. Conclusions

Blockchain is different from other consensus technology because it works with unknown number of participants n without trying to determine it. But in return, its state transitions are necessarily probabilistic, because economic measures introduced to avert impossibility of voting allow faulty participants to alter decisions by paying greater cost. Blockchain is not suitable where finality of decisions is needed, because no decision is committed or final.

Recommended use of blockchain is where n is large and unknown, the number of faulty participants f is unbounded, and state transitions are allowed to be probabilistic. Eclipse attacks need to be handled, as blockchain does not tolerate network partitioning of unbounded duration.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008, http://bitcoin.org/bitcoin.pdf.

[2] Nasdaq, “Nasdaq Linq Enables First-Ever Private Securities Issuance Documented With Blockchain Technology,” December 2015, http://ir.nasdaq.com/releasedetail.cfm?ReleaseID=948326.

[3] R3 CEV, “R3 CEV,” 2015, http://r3cev.com.

[4] L. Lamport, “Byzantizing Paxos by Refinement,” in *Distributed Computing - Lecture Notes in Computer Science Volume 6950*. Springer, 2011, pp. 211–224.

[5] J. Gray and L. Lamport, “Consensus on Transaction Commit,” *ACM Transactions on Database Systems (TODS)*, vol. 31, no. 1, pp. 133–160, March 2006.

[6] J.-P. Martin and L. Alvisi, “Fast Byzantine Consensus,” *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 202–215, July 2006.

[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, April 1985.

[8] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, July 1982.

[9] E. A. Alchieri, A. N. Bessani, J. S. Fraga, and F. Greve, “Byzantine Consensus with Unknown Participants,” in *Proceedings of the 12th International Conference on Principles of Distributed Systems (OPODIS ’08)*, 2008, pp. 22–40.

[10] W. Vogels, “Eventually Consistent,” *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, January 2009.

[11] L. Lamport, “The Part-time Parliament,” *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, May 1998.

[12] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance and Proactive Recovery,” *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, November 2002.

[13] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, “Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults,” in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2009, pp. 153–168.

[14] M. Iwamura, Y. Kitamura, T. Matsumoto, and K. Saito, “Can We Stabilize the Price of a Cryptocurrency?: Understanding the Design of Bitcoin and Its Potential to Compete with Central Bank Money,” Institute of Economic Research, Hitotsubashi University, Discussion Paper Series A No.617, November 2014.

[15] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: a tutorial,” *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, December 1990.

[16] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse Attacks on Bitcoin’s Peer-to-peer Network,” in *Proceedings of the 24th USENIX Conference on Security Symposium*. USENIX Association, 2015, pp. 129–144.

[17] C. Decker and R. Wattenhofer, “Information Propagation in the Bitcoin Network,” in *13th International Conference on Peer-to-Peer Computing*. IEEE, 2013, pp. 1–10.
[18] G. Wood, “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER,” 2014, http://gavwood.com/paper.pdf.
[19] D. Cavin, Y. Sasson, and A. Schiper, “Consensus with Unknown Participants or Fundamental Self-Organization,” in *Ad-Hoc, Mobile, and Wireless Networks - Lecture Notes in Computer Science Volume 3158*. Springer, 2004, pp. 135–148.
[20] Linux Foundation, “The Hyperledger Project,” 2016, https://www.hyperledger.org.
[21] D. Schwartz, N. Youngs, and A. Britto, “The Ripple Protocol Consensus Algorithm,” 2014, https://ripple.com/files/ripple_consensus_whitepaper.pdf.