Decentralized Consensus for P2P Network with Trust Relationships

Houwu Chen
Department of Computer Science and Technology
Tsinghua University
chenhw11@mails.tsinghua.edu.cn

Abstract—The decentralized nature of P2P network increases robustness because it removes the single point of failure, however, reaching consensus over an open P2P network while maintaining decentralization is still an open problem. We proposed a consensus algorithm for P2P network based on trust relationships and social consensus with emergent phenomena of complex system. Starting from synchronous system model, we model the algorithm with mean field equations and analyze the performance of convergence as well as fault tolerance. We then adapt the algorithm to asynchronous system model by incorporating a message filter and a Chandra & Toueg style failure detector. Simulations show that on the SNAP dataset of the Wikipedia who-votes-on-whom network, the algorithm can always converge within 40 rounds in synchronous system model and 70 seconds in synchronous system model under reasonable latency assumption if without failure. Simulations also show that the algorithm can tolerant collusion of 15% random nodes or 2% top influential nodes committing attack of the strongest type.

I. INTRODUCTION

P2P network is well known on its decentralized nature that increase robustness because it removes the single point of failure, however, no approach to reaching consensus over an open P2P network while still maintaining decentralization attracts public attention until the emergence of Bitcoin, which is a cryptocurrency, a form of money that uses cryptography to control its creation and management, rather than relying on central authorities [1]. Decentralization of consensus eliminates trust in central authorities while also bring great challenges for consensus over an open p2p network. One major challenge is sybil attack, wherein the attacker creates a large number of pseudonymous identities, and use them to gain a disproportionately large influence [2]. Because of decentralization, there is no logically central, trusted authority to vouch for a one-to-one correspondence between entity and identity, thus make it difficult to resist sybil attack. Another challenge is the byzantine failure, which encompasses both omission failures (e.g., crash failures, failing to receive a request, or failing to send a response) and commission failures (e.g., processing a request incorrectly, corrupting local state, and/or sending an incorrect or inconsistent response to a request) [3]. Because of decentralization, there is no central coordinator and it’s even impossible to know the exact number of participants taken part in the consensus, which make byzantine failure even more difficult to tolerant [4].

Bitcoin deals with those challenges by implicitly combining the generation of initial states as well as and consensus process in an ongoing chain of hash-based proof-of-work [5]. The majority decision of Bitcoin is represented by the longest chain, which has the greatest proof-of-work effort invested in it. As long as a majority of computing power is controlled by nodes that are not cooperating to attack the network, they’ll generate the longest chain and outpace attackers. However, the proliferation of ASIC miner and mining pools leads to the monopoly of computing power, enabling that one single principal with top influence can control the consensus and even subvert the consensus by withholding blocks [6], [7]. This dilemma is a violation of the original idea of bitcoin which tried to eliminate central control to achieve full decentralization.

Our idea to achieve decentralized consensus over an open p2p network consists of two key elements i.e. trust relationships and social consensus with emergent phenomena of complex system based on the trust relationships. Each node is identified by its public key, other nodes follow the node if they trust it, and thus the whole network can be abstracted to a directed graph representing the trust relationships between nodes. During a consensus process, each node broadcast its value signed with its private key to its followers, and each node decides its next value according to the latest values of its followees. After limited rounds of iteration, the whole network reach consensus that all honest nodes will have the same value. Since each node make decisions only according to its followers, no global view of the whole network is needed, and even the exact number of participants in the consensus process is not needed.

Starting from synchronous system model, we model the algorithm with mean field equations and analyze the performance of convergence as well as fault tolerance. The synchronous system model is impractical for P2P network, but the model enable us to get insight of the characteristics of our algorithm conveniently. We then adapt the algorithm to asynchronous system model by incorporating a message filter and a Chandra & Toueg style failure detector. Simulations show that on the SNAP dataset of the Wikipedia who-votes-on-whom network [8], the algorithm can always converge within 40 rounds in synchronous system model and 70 seconds in synchronous system model under reasonable latency.
assumption if without failure. Simulations also show that the algorithm can tolerate collusion of 15% random nodes or 2% top influential nodes committing attack of the strongest type.

II. RELATED WORK

Consensus fundamental Consensus is a fundamental problem for reliable distributed system to achieve agreement among distributed nodes on a value or an action, even when a number of those nodes are faulty [9]. The first instance of the consensus problem is presented in [10]. Examples of applications of consensus include whether to commit a transaction to a database, agreeing on the identity of a leader, state machine replication, and atomic broadcasts. In a crash failure model nodes may fail by stopping, while in a Byzantine failure model nodes may fail without stopping, Byzantine failures are more complicated than crash failures, because nodes may exhibit arbitrary, erratic and unexpected behavior which may even be malicious and disruptive [3], [11]. Consensus for asynchronous system which has no assumptions about the relative speeds of nodes nor about the delay time in delivering a message is also more generic than synchronous system. In fact, RLP impossibility even tells that strict consensus without relaxation on some criteria is impossible for asynchronous system [12]. Representative consensus algorithms including leader election based algorithms [13], [14], randomized algorithm [15], failure detectors based algorithm [16] etc.

Consensus with unknown participants Traditional consensus algorithms assume a static and known set of participants which stands in classic distributed system like a server clusters system. But for dynamic and self-organizing systems like P2P networks the participants are unknown, thus traditional consensus algorithms does not work in such scenarios. To deal with unknown participants, consensus algorithms based on graph theory are proposed [17], [4] where sink component is first discovered, then the consensus process is actually executed within the sink component and the decision is disseminated to other participants after that. However, the algorithm is sensitive to the topology of the graph, and it give no solution if there are several sink components discovered. Also the algorithm does not deal with Sybil attack problem. Another approach based on pseudo leader election is also proposed, but it can only deal with crash failure [18]. Consensus algorithm with Byzantine fault tolerance over wireless Ad Hoc networks without known participants is also introduced, but it relied on a strong assumption that the network is single-hop which does not stand in a typical P2P network [19].

Sybil attack Sybil attack is a challenging security problem in P2P systems, wherein the attacker creates a large number of pseudonymous identities, and use them to gain a disproportionately large influence [2]. One approach to resisting Sybil attack is relying on a certifying authority to perform admission control [20], however, decentralization is broken by the certifying authority. Another approach is remote issuing anonymous certification of identity by identifying distinct property of a node, for example utilizing geometric techniques to establish location information [21], but the algorithm can’t tolerance changes of the network environment which conflicts with the intrinsic characteristic of dynamic P2P networks. Puzzle computing is also introduced to increase the overhead of Sybil attack, such puzzles involve posing a challenge that requires a large amount of computation to solve but is easy to verify [22]. However, this type of algorithms forces honest nodes to continually spend computing resources solving puzzles, and also there’s no way to resist Sybil attack if the attacker has dominant computing resources. Sybil prevention techniques based on the connectivity characteristics of social graphs is another direction [23], [24]. Because of the difficulty to engineer social connections between attack nodes and honest nodes, this approach is considered to be more robust over other ones [25], [26], [27].

Social consensus Consensus over complex networks is intensively studied in the field of statistical physics where mean field equation is widely used to model and analyze the complex network [28], [29], [30], [31], [32], related topics includes opinion negotiation and epidemic spreading [33], [34], [35], [36]. The impact of committed individuals which are immune to influence during the consensus process is also identified that a small fraction of committed individuals can reverse the prevailing majority opinion rapidly [32]. Later studies further analysis the impact of committed individuals in different scenarios, e.g. different interaction rules, and even two groups committed to competing opinions [37], [38], [39], [40].

Consensus in crypto currencies Bitcoin lay its consensus mechanism on top of proof of work(PoW) and blockchain [5], but the monopoly of computing power enabling that one single principal with top influence can control the consensus and even subvert the consensus by withholding blocks [6], [7]. Peercoin proposed the proof of stake(PoS) algorithm which use staked as the vote power instead of computing resource [41], and derivatives of PoS are also developed in BitShare, Blackcoin and NXT [42], [43], [44]. PoS is blamed to enable attackers to vote on different branches losslessly thus it’s actually nothing at stake [45], [46], [47]. Ripple and Stellar use a different consensus algorithm which leverages trust relationships without blockchain and PoW/PoS involved [48]. The algorithm built on the implicit assumption that if a node has 80% followees agreed on something, then 80% of total nodes agree with the same thing, however, the assumption only stand when a node follows a large portion of total nodes. This assumption is not only hard to reserve in practical P2P network but also limit the scale of the network which leads to the reduction of decentralization, and at the time of writing, the ripple network only have 8 consensus servers [49]. After a consensus failure event and review of consensus algorithm from security researcher, Stellar admitted that the algorithm has innate weaknesses and decided to switch to a centralized solution [50]. TenderMint is a still in progress consensus algorithm without PoW/PoS involved too [51], but it require each node to collect votes from 2/3 of all the participants, thus it require global information of the whole network and also put a limitation on the scale of the network.
III. Model and Problem Formulation

A. System model

We focus on the system with the following properties.

P2P network We consider a peer-to-peer network where peers are equally privileged and equipotent participants in the network.

Trust relationships There are trust relationships among nodes, and when node $A$ trust node $B$, $B$ is called as followee while $A$ is called as follower of the relationship. The network can be abstracted to a directed graph where each peer is a node, and each trust relationship is a edge. To ensure connectivity and safety, each node is constrained to have at least a minimum number of followees, thus in the formed directed graph each node’s indegree must be equal or greater than a given number, meanwhile a node may have zero outdegree.

Local information There are no centralized global coordinators in the network, and each node have no global view of the whole network. A node can only have public information of its direct followees while know nothing about other nodes. A node also have no information about the whole network including the number of the participants, the structure of the network etc.

Public broadcast channel We define a new primitive named public broadcast channel (PBC) which has the following properties:

1) A followee broadcast messages to all its followers
2) The channel are unidirectional, i.e. a followee can’t receive messages from its followees through the channel.
3) All followers of a followee receive the same message, thus a followee can’t send one value to a followee whereas send another value to another followee.
4) If a message broadcasted by a followee is corrupted, it can be detected by the followees.
5) A message broadcasted by a followee can’t be forged or modified by other nodes.

Asynchronous A message from a followee to a follower may be delayed, duplicated or delivered to the followee out of order.

Byzantine failure A node is correct if it behave honestly and without error. Conversely, a node is faulty. A faulty node may have Byzantine failure exhibiting arbitrary, erratic and unexpected behavior which may even be malicious and disruptive. Example failures include crash, failing to receive a request, or failing to send a response, as well as processing a request incorrectly, corrupting local state, or sending an incorrect or inconsistent response to a request.

B. Problem statement

Goal of consensus In traditional definition of consensus, specifically binary consensus, each node has a initial value $v_i \in \{0, 1\}$. The consensus problem is to decide upon a common value among all nodes. A solution to the problem must fulfill the following three requirements:

1) Agreement: no two correct nodes decide differently.
2) Validity: if a correct node decides $v$, then $v$ was a proposed value. In the context of PBC, $v$ must be broadcasted by some nodes.
3) Termination: every correct node decides exactly once.

The first two are safety requirements which say that some bad thing cannot happen, while the last is a liveness requirement which states good things that must happen. We call a solution to the consensus problem with this agreement requirement $C1$.

Eclipse attack and weak consensus The three requirements can be possibly fulfilled in a traditional environment where each node communicates with all other nodes, but in the P2P network as we defined in the local information property of the system model, the connections to the other nodes are limited or each node will suffer a data bloating problem. Since each nodes only have informations of its followees, if it is under eclipse attack where the majority of its followees are malicious, there is no possible solution for it to survive the attack. Here the eclipse attack may be high order.

Definition 1 (High order eclipse attack). For node $A$ and $B$, node $A$ is the followee of node $B$, node $A$ is under order $i$ eclipse attack, if the conditions described later is fulfilled, then $B$ is called to be under order $i + 1$ attack. The condition are list in the following:

1) If node $A$ is not eclipsed attacked, then $B$ can survive the attack
2) If $A$ is eclipse attacked, then $B$ can’t survive the attack

The three properties of consensus can only stand when nodes of the network are in stop failures, whereas under malicious attacks, considering the presence of eclipse attacks, the agreement requirement is relaxed to a weaker one, i.e. no two correct nodes which are not eclipsed attacked decide differently. We call a solution to the consensus problem fulfills those weaker requirements $C2$.

Network as the subject of consensus We further consider the network as a living system, and the subject of consensus shift from individual nodes to the network as a whole. In this perspective, we don’t have to ensure every correct node without being eclipsed attacked to decide on the same value, instead, we just need to ensure that the overwhelming majority of the nodes decide on the same value. We call a solution to the consensus problem with this agreement requirement $C3$.

Our goal is to ensure that $C3$ must be retained in any case or the system is regarded as failed to reach consensus. However, we should try to reach $C2$ and even $C1$ as close as possible, and give node possibility to quit consensus when it find something abnormal. That’s the reason why confused decision and distributed oracle are presented in later sections.

IV. Experimental Dataset

Apparently, no consensus algorithm can work well on arbitrary network topology. For example, a network consists of two disjoint cliques have no chance to reach consensus. But for real network with trust relationships e.g. the SNAP dataset of the Wikipedia who-votes-on-whom network which is called
1) The time unit is round, and each message is delivered in exact one round.
2) When round \( i \) begin, each node receives all the messages broadcasted by its followees in round \( i-1 \).
3) Each node then finishes processing the received messages in no time, and broadcast its new value at the end of round \( i \).

According to the local information property of the system model, each node is nondistinctive for an algorithm, and a node can only receive messages from its followees. Thus an algorithm can be formalized as a function \( F \) at time \( t \) for each node \( i \), by designating the value of node \( i \) as \( v_i \), an algorithm can be written in the following general equation:

\[
v_i(t+1) = F(v_i(t), V_i(t))
\]

where \( V_i(t) = [v_{f_1}(t), v_{f_2}(t), \ldots, v_{f_n}(t)] \) and \( f_1, f_2, \ldots, f_n \) are followees of node \( i \).

To analyze properties of our algorithms, we build the theoretical models with the mean field method. We denote the densities of all nodes with value 0 and 1 to be \( p_0 \) and \( p_1 \) respectively, the densities of correct nodes with value 0 and 1 to be \( c_0 \) and \( c_1 \) while densities of faulty nodes with value 0 and 1 to be \( f_0 \) and \( f_1 \). Because the symmetrical property of the algorithm on binary value 0 and 1, and \( p_0 + p_1 = 1 \), it’s sufficient to only track \( p_0 \) and only considering \( p_0 > 0.5 \).

We also assume initially faulty nodes are evenly distributed among all the nodes with density of \( f \). Thus one can write the following two equations which will be used later to facilitate analyzing fault tolerance. Eq. (2) stands before the consensus process start, and at any time we have Eq. (3).

\[
\begin{align*}
 p_0 + p_1 &= 1 \\
c_0 &= p_0 \cdot (1 - f) \\
c_1 &= p_1 \cdot (1 - f) \\
f_0 &= p_0 \cdot f \\
f_1 &= p_1 \cdot f \\
f_0 + f_1 &= f \\
c_0 + c_1 &= 1 - f \\
c_0 + f_0 &= p_0 \\
c_1 + f_1 &= p_1
\end{align*}
\]

Also, according to \( f_0 + c_0 = p_0 \) in Eq. (3), we can also have:

\[
\frac{dp_0}{dt} = \frac{dc_0}{dt} + \frac{df_0}{dt}
\]

For system without failures, we have:

\[
\begin{align*}
f_0 &= 0 \\
f_1 &= 0 \\
c_0 &= p_0 \\
c_1 &= p_1 \\
c_0 + c_1 &= 1
\end{align*}
\]
from Eq. (6), the evolution of majority. Following the same denotation with Eq. (1), the value of its followees broadcasted in the last round following mean field equation:

and thus Eq. (4) can be written to the following equation:

We first deal with the convergence problem assuming that all nodes is correct, i.e \( f_0 = f_1 = 0 \), starting from a greedy algorithm which has best convergence speed but may stuck during the convergence process, then proposed a simulated annealing algorithm which won’t stuck but has low convergence speed, and at last we combine the two algorithms and presented a compound algorithm. Note here even we follow the same naming convention with traditional mathematical optimization techniques, the meaning is totally different, instead, it’s only the driving force to producing emergent phenomena of a complex network.

The performance of fault tolerance considering several types of attacks is analyzed in the next section.

A. Greedy

The greedy algorithm is quite straightforward, it simply count the value of its followees broadcasted in the last round as well as the value of its own and then choose the value of majority. Following the same denotation with Eq. (1), the algorithm is shown in Fig. 2.

Neglecting correlations between nodes, and fluctuations, from Eq. (6), the evolution of \( p_0 \) can be written in the following mean field equation:

\[
\frac{dp_0}{dt} = s_1 c_1 - s_0 c_0
\]

(7)

Where \( s_1 \) is the probability that a node switch from value 1 to value 0, and \( s_0 \) is the probability of that a node switch from value 0 to value 1.

A node flips from value 1 to value 0 only when the count of its followees with value of 1 in the last time is less than \( D/2 \), and vice versa. By specifying the mean indegree and outdegree of a node to be \( D \), the following equations can be established:

\[
\begin{align*}
\{ & s_1 = F(D/2 - 1; D, p_1) + \frac{1}{2}d(D/2; D, p_1) \\
& s_0 = F(D/2 - 1; D, p_0) + \frac{1}{2}d(D/2; D, p_0)
\}
\]

(8)

where \( F(k; n, p) \) is the cumulative distribution function and \( d(k; n, p) \) is the probability mass function for \( k \) successes in binomial distribution of \( n \) trials with probability \( p \).

The relationship between \( dp_0/dt \), \( D \) and \( p_0 \) can be plotted to Fig. 3a, and actually \( dp_0/dt \) is the convergence speed. The evolution of \( p_0 \) along with time \( t \), starting with \( p_0 = 0.501 \) at \( t = 0 \), and \( D \in [5, 300] \) is shown as Fig. 3b. We have the following observations:

1) \( \forall p_0 \in (0.5, 1) \) and \( \forall D > 0 \), the converge speed is positive, i.e. \( p_0 \) strictly strictly increases with time \( t \).
2) System with greater degree \( D \) will converge more quickly.
3) With a tiny deviation of \( p_0 \) from 0.5, even when \( D = 5 \), \( p_0 \) can still converge to 0 in about 10 rounds.
4) The convergence does not dependent on the number of nodes.

Simulations of the greedy algorithm is shown as Fig. 4, where convergence is defined as the following equation:

\[
cvg = abs(c_0 - c_1)/(c_0 + c_1)
\]

(9)

From the equation we can see that when \( c_0 = c_1 = 0.5 \),
convergence will be 0, while \( c_0 = 1 \) or \( c_1 = 1 \), convergence will be 1 which means consensus is reached. Fig. 4b shows the histogram of rounds to reach consensus for each run of 40 rounds. Note in Fig. 4b, consensus at round of 41 means the system failed to reach consensus in that run, and also each bin of the histogram is 2. From the figure we can see:

1) For each dataset, simulation result approximately fits theoretical analysis.
2) Network scale has no apparent impact on convergence speed.
3) For uniform network consensus can always be reached rapidly no matter what the network scale is.
4) Mean convergence can’t reach 1 for the wiki dataset, caused by some runs which can’t reach consensus as shown in Fig. 4b. The reason for this phenomenon we think is the whole network gradually splits into two cohesive subgraph because of the community structure.

B. Simulated Annealing

To overcome the hurdle that the greedy algorithm may stuck at some runs, we developed the simulated annealing (SA) algorithm shown in Fig. 5. It provide the ability for a node to escape from its current value at the probability proportional to the ratio of its followees with the other value to all its followees, while keep a node at a stable state while the ratio is great enough.

Following the same notations and method of deducing in the description of the greedy algorithm, the evolution of \( p_0 \) can be written in the following mean field equation:

\[
\frac{dp_0}{dt} = s_1 c_1 - s_0 c_0
\]

\[
s_1 = F(0.2D; D, p_0) + \sum_{i=0.2D}^{0.8D} f(i; D, p_0)(D - i) D + \frac{1}{2D} \]

\[
s_0 = F(0.2D; D, p_0) + \sum_{i=0.2D}^{0.8D} f(i; D, p_0)(D - i) D + \frac{1}{2D} \]

The relationship between \( dp_0/dt \), \( D \) and \( p_0 \) can be plotted to Fig. 6a while the evolution of \( p_0 \) along with time \( t \), starting with \( p_0 = 0.501 \) at \( t = 0 \), and \( D \in [5,300] \) is shown as Fig. 6b. From those figures we can see the following facts:

1) The convergence speed is never negative.
2) When \( p_0 \) is near 0.5, the converge speed is zero if the degree \( D \) is not too small.
3) When \( D \) is not too small, starting with \( p_0 = 0.501 \) at \( t = 0 \), the system makes no progress in consensus over time.

Following the notations and definitions same with simulation of the greedy algorithm, convergence of the SA algorithm is shown as Fig. 7a while Fig. 7b shows the histogram of rounds to reach consensus for each run of 40 rounds. From the two figures, we have the following conclusions and explanations:

1) For the uniform-more dataset with \( D = 33 \), none of the runs reach consensus.
2) For the uniform dataset with \( D = 33 \), there are tiny chances to reach consensus within 40 rounds.
3) For the uniform-less dataset with \( D = 33 \), about half runs reach consensus within 40 rounds.
4) The average convergence for uniform-more and uniform is nearly at the same level along with time. Note here positive convergence does not mean \( p_0 > p_1 \). Actually for each individual run, the ratio of \( p_0/p_1 \) always

```plaintext
1: function SIMULATED_ANNEALING(V(t), v(t))
2: \( n_0 \leftarrow \text{count 0 in } (V_f(t), v(t)) \)
3: \( n_1 \leftarrow \text{count 1 in } (V_f(t), v(t)) \)
4: \( n \leftarrow n_0 + n_1 \)
5: if \( n_0 > n \times 0.8 \) then
6: return 0
7: else if \( n_1 > n \times 0.8 \) then
8: return 1
9: else
10: test \leftarrow \text{random select from } [0, n]
11: if \( test < n \) then
12: return 0
13: else if \( test > n_0 \) then:
14: return 1
15: else
16: return \text{random select from } \{0, 1\}
```

Fig. 5. Simulated annealing algorithm
oscillate around 1.

5) The theoretical model simulation only conform when the network scale is big enough. The explanation to this phenomenon is that when a small portion of nodes escape from their existing values, the effect is non-negligible when the network scale is small, while in the big network scale, small disturbance can be absorbed. Simulations also show that for a run that finally reach consensus, it actually keeps oscillating until it suddenly escaped from oscillating at some time and then reach consensus within just several rounds.

6) For wiki dataset, the convergence steadily increases with time. The reason that they behave quite different to uniform dataset is that the node degrees are disequilibrium, actually, because of the scale-free property, a large proportion of node degrees are about 10. And as we already know from theoretical analysis, those nodes has a rapid convergence speed.

C. Compound

To leverage the ability of the greedy algorithm to make quick progress in consensus as well as the ability of the SA algorithm to escape from stuck, we proposed the compound algorithm by synthesizing the two algorithms as shown in Fig. 8.

For the compound algorithm, \( dp_0/\text{dt} \) is just a linear combination of that of the greedy and the SA algorithm as the following equation, where \( d_g p_0/\text{dt} \) and \( d_s p_0/\text{dt} \) are the derivatives of the greedy algorithm and the SA algorithm respectively:

\[
\frac{dp_0}{dt} = d_g p_0/\text{dt} \cdot ratio + d_s p_0/\text{dt} \cdot (1 - ratio) \tag{11}
\]

For ratio = 0.5, the relationship between \( dp_0/\text{dt} \), \( D \) and \( p_0 \) can be plotted to Fig. 9a and the evolution of \( p_0 \) along with time \( t \), starting with \( p_0 = 0.501 \) at \( t = 0 \), and \( D \in [5, 300] \) is shown as Fig. 9b.

We simulate the compound algorithm on the wiki dataset for 1000 runs with ratio = 0.5. Following the notations and definitions same with simulation of the greedy algorithm, the convergence and rounds to converge of all the algorithms on the wiki dataset are show in Fig. 10a and Fig. 10b, and the convergence and rounds to converge of the compound

---

**Fig. 6.** Theory analysis of the SA algorithm

**Fig. 7.** Simulation of the simulated annealing algorithm

**Fig. 8.** Compound algorithm

1: function SIMULATEDANNEALING(\( V_i(t) \), \( v_i(t) \), ratio = 0.5)
2: \( test \leftarrow \) random select from \{0, 1\}
3: if \( test < ratio \) then
4: return GREEDY(\( V_i(t) \), \( v_i(t) \))
5: else
6: return SIMULATEDANNEALING(\( V_i(t) \), \( v_i(t) \))
From the theoretical analysis and simulation of the above, we can conclude that the compound algorithm can reach consensus in probability 1, and the convergence performance increases with degree while has no obvious relationship with network scale.

VI. FAULT TOLERANCE ANALYSIS

According to the Byzantine failure property of the system model, nodes can behave arbitrarily. However, we focus on the following four representative failure types:

1) Faulty nodes crash and no longer broadcast their value in later rounds. This failure is named as crash failure.
2) Faulty nodes are malicious, immune to the messages...
from their followees and collude to mislead the system to reach consensus on one value (e.g., 1) even when actually the majority of the nodes initially have another value (e.g., 1). This failure is named as mislead attack.

3) Faulty nodes are malicious, immune to the messages from their followees and collude to prevent the system to reach consensus. This failure is named as consensus preventing attack.

4) Each faulty node generate random values in each round, meanwhile, all faulty nodes behaves independently to other faulty nodes. This failure is named as random-value failure.

To measure performance of faulty tolerance, we define critical point which will be used in later sections.

**Definition 2** (Critical point). For a given \( p_0 \) of value initial and target convergence of value \( cvg \), if \( f_{\text{critical}} \) fulfill the following two requirements, then \( f_{\text{critical}} \) is the critical point for \( p_0 = \text{initial} \) at target convergence \( cvg \):

1. As long as \( f < f_{\text{critical}} \), when \( t \to \infty \), \( p_0/(1 - f) > cvg \).
2. \( \beta f' \) such that \( f \) fulfill the previous requirement while \( f^{''} \geq f_{\text{critical}} \).

**A. Crash Failure**

In a synchronous model, when a node is in crash failure, it will not present in its follower’s received messages in later rounds. Since for a node the compound algorithm only count the values in its received messages, the node in crash failure has the effect equivalent to a system with a different topology that the faulty node does not exist in it. So as along as the crash failure does not lead to ill formed network topology like the example of two disjoint cliques we showed previously, the system will reach consensus in the manner same with the case where all the nodes are correct.

**B. Mislead Attack**

Under mislead attack, even the majority of correct nodes are with value of 0, faulty nodes are immune to messages from their followees and always broadcast 1 in each round. Thus we have \( f_0 = 0, f_1 = f, df_0/dt = 0 \), and Eq. (4) can be simplified to:

\[
\frac{dp_0}{dt} = \frac{dc_0}{dt}
\]  

where \( dc_0/dt \) is Eq. (1) with:

\[
\begin{align*}
    p_0 &= c_0 \\
    p_1 &= c_1 + f
\end{align*}
\]  

For \( D = 33 \) and \( \text{ratio} = 0.5 \), the relationship between \( p_0 \), \( p \) and \( dp_0/dt \) is exhibited in Fig. [12] where we have the follow observations:

1. \( \forall p_0 \in (0.5, 1) \), \( \exists p_{\text{thresh}} \), such that when \( p > p_{\text{thresh}} \), the convergence of correct nodes will decrease deterministically.
2. \( \exists p_{\text{max}} \in (0.5, 1) \) as \( p_0 \), where its corresponding \( p_{\text{thresh}} \) denoted as \( p_{\text{max}} \) is greater than other \( p_{\text{thresh}} \).

Following Definition 2, for target convergence \( cvg = 0.9 \), by iterating on the mean field equation, critical points can be plotted in Fig. [13] where solid lines are critical points. There are also dotted lines where each point \( (p_0(i), f(i)) \) means that for the \( p_0(i) \), at the current time, when \( f < f(i) \), \( p_0/(1 - f) \) will increase, while when \( f > f(i) \), \( p_0/(1 - f) \) will decrease.

From the figure we can have the following observations:

- \( \forall D \in [10, 400] \), critical point weakly increases with \( p_0 \), i.e. greater \( p_0 \) can tolerate heavier attack.
- \( \forall p_0 \in (0.5, 1) \), with greater \( D \) \( f \) can reach its maximal value with smaller \( p_0 \).
- \( \forall D \in [10, 400] \), as long as \( p_0 \geq 0.75 \), the system can tolerant attack with \( p \geq 0.15 \).
where $d_{c0}/dt$ is exactly same as Eq. (11), while $df_0/dt$ can be written in the following equation:

$$\frac{df_0}{dt} = F(D/2 - 1; D, p_0) \ast f_1 - F(D/2 - 1; D, p_1) \ast f_0 \quad (14)$$

Following the same approach, critical points of the random value failure are demonstrated in Fig. 14 (where solid lines are critical points of random value failure, and dotted lines are critical points of mislead attack). From the figure we can see:

- $\forall D \in [10, 400]$, critical point weakly increases with $p_0$, i.e. greater $p_0$ can tolerate heavier attack.
- For same $D$ an $p_0$, critical points of the random-value failure are never smaller than that of the mislead attack.

D. Random-Value Failure

Under the random value failure, faulty nodes are immune to messages from their followees and always broadcast a random value from $\{0, 1\}$ in each round. By assuming the probabilities of value 0 and 1 is equal, we have $f_0 = f_1 = f/2$ and further $df_0/dt = 0$, thus Eq. (14) can be simplified to:

$$\frac{dp_0}{dt} = \frac{dc_0}{dt} \quad (15)$$

where $dc_0/dt$ is Eq. (11) with:

$$\begin{cases} 
  p_0 = c_0 + \frac{f}{2} \\
  p_1 = c_1 + \frac{f}{2}
\end{cases} \quad (16)$$

Following the same approach, critical points of the random value failure are demonstrated in Fig. 15 (where solid lines are critical points of random value failure, and dotted lines are critical points of mislead attack). From the figure we can see:

- Critical points nearly have little dependency on $p_0$.
- $\forall D \in [10, 400]$, critical point weakly increases with $p_0$, i.e. greater $p_0$ can tolerate heavier attack.
- $\forall D \in [10, 400]$ and $\forall p_0 \in (0.5, 1)$, critical point are greater than 0.3
- For same $D$ an $p_0$, critical points of the random-value failure are always greater than that of the mislead attack.

E. Comparison

From the previous analytics, we can conclude that of the several types of failures, mislead attack is the strongest. The system has the lowest tolerance on mislead attack, and to compromise the system, one can commit mislead attack with the least resource. This conclusion also conforms intuition, that when majority correct nodes have value of 0, under mislead attack all of the faulty nodes have value of 1, while either under random-value failure or consensus preventing attack only part of the faulty nodes have value of 1.

F. Sybil Attack

We can prove that Sybil attack has no impact on mislead attack by mathematical induction. We first assume current system has $N$ Sybil nodes, thus $dp_0/dt$ can be expressed in Eq. (12) where $dc_0/dt$ is Eq. (11) with Eq. (13). One key factor of $f$ is that $f$ is not the density of faulty nodes among all nodes, instead, it’s the density of following relationships from correct nodes to faulty nodes. Thus when a new Sybil node joins, because it can’t engineer the following relationships from correct nodes to it by itself, the density $f$ still has no change, even actually there are now $N + 1$ Sybil nodes.

From the proof we can see that to compromise the system, creating new Sybil nodes is useless, instead, the attacker should increase the density of following relationships from correct nodes to faulty nodes, i.e. attract more correct nodes to following faulty nodes.

VII. ASYNCHRONOUS ALGORITHM

In the asynchronous time model, the algorithm introduced for synchronous time model does not work because of the following issues:

1) No global clock is available.
2) Messages may be transmitted to followees with different latencies.
3) Even all nodes are correct, a node still can’t wait messages from all its followees, because the messages broadcasted by its followees may reach the node in too long time, or even lost in the p2p network.
4) Nodes may enter a specific round in different time, e.g. when a node is in round 10, another node may be in round 20 or even 1.
1: function TransitState(node_i, V_i(t), v_i(t))
2: \( \text{THRESH} \leftarrow 2 \)
3: if node_i.round < MIN_ROUND then
4:     return deciding
5: \( n_0 \leftarrow \text{count 0 in } (V_i(t), v_i(t)) \)
6: \( n_1 \leftarrow \text{count 1 in } (V_i(t), v_i(t)) \)
7: if \( n_0 > n_1 \times \text{THRESH} \) or \( n_1 > n_0 \times \text{THRESH} \) then
8:     return decided
9: return confused

Fig. 16. State transition algorithm

5) As a consequence of the previous issues, nodes updates their value in different time.
6) When a node is in round \( r \), it may also receive messages from its followees of earlier or later rounds.

To deal with the issues described above, we adapt the synchronous algorithm by incorporating a message filter and a Chandra \& Toueg style failure detector [16]. We also introduce the implementation of PBC, and clarify the criteria for a node to make final decision.

A. Implementation of PBC

The public broadcast channel primitive is implemented by DHT and asymmetric cryptography. For a node as followee, all its followers and itself form a sharing group (known as a “swarm”) identified by the followee’s public key. Each broadcasted message is signed with the private key of the followee, and the followers can check the identity and integrity with the followee’s public key. Since the message is sharing by all nodes in the swarm without destination specified, and for each round there should be at most one broadcasted messages, if a follower received two different messages indicating the same round, it will broadcast the finding through peer exchange thus all the other followers will know it immediately.

B. Final Decision

A node decide to make its final decision when enough rounds passed, e.g. 40 rounds. We defined three types of state: \( \text{deciding}, \text{decided}, \text{and confused} \), each of them has a \( \text{stage} \) property while \( \text{deciding}.\text{stage} = 0 \), \( \text{decided}.\text{stage} = 1 \) and \( \text{confused}.\text{stage} = 1 \). A node is \( \text{deciding} \) before making final decision. If a node finally decide on 0 or 1, then it’s \( \text{decided} \). A node is \( \text{confused} \) if it’s considered to be safe on neither 0 nor 1. The state transition of a \( \text{node}_i \) is illustrated in Fig. [16] where the value of \( \text{node}_i \) is designated to be \( v_i(t) = [v_{f_1}(t), v_{f_2}(t), \ldots v_{f_n}(t)] \) and \( f_1, f_2, \ldots f_n \) are followees of \( \text{node}_i \). The \( \text{THRESH} \) constant controls the strategy to be aggressive or conservative. Greater \( \text{THRESH} \) leads to less nodes to decide at wrong value while also leads to more nodes to decide to be confused.

C. Message Filter

Each message \( \text{msg} \) broadcasted by \( \text{node}_i \) is a tuple of \( \text{(nodeid, round, value, state)} \), where \( \text{nodeid} \) is the id of \( \text{node}_i \), \( \text{round} \) and \( \text{value} \) is its current round and value, and \( \text{state} \in \{ \text{deciding, decided, confused} \} \). For a \( \text{node} \), the message filter will refuse to accept any new messages if it has already made its final decision, and it will always keep at most one message from a followee with the largest \( \text{round} \) denoted as \( \text{round}_{\text{max}} \) and \( \text{round}_{\text{max}} \geq \text{msg}.\text{round} \). The filter algorithm applied when a node receiving a new message is depicted in the MessageFilterAtRcv function of Fig. [17] while when a node finish a round after broadcasting to its followees, it will apply the post round message filter algorithm as depicted in the MessageFilterPostRound function of Fig. [17].

D. Failure Detector

A failure detector is designed to deal with issues related with the asynchronous time model, and note it does nothing related with attacks. The key idea of the failure detector is that each node maintains a followee nodes list as well as a suspect nodes list. Initially, all followees are in the followee nodes list, a node is moved to the suspect nodes list if no valid message received from it for a long time, while a node is moved from the suspect nodes list to the followee nodes list when a new valid message from it is received. A \( \text{node} \) will apply the compound algorithm when its message buffer has messages from all followee nodes. The failure detector algorithm is shown in Fig. [18] consists of three event handler function: a new message event handler OnMessage, a timer event handler OnTimer, a OnTry event handler triggered when try to finish the current round of a node.

VIII. Simulation

According to existing studies, latency between peers in DHT is mostly between 50 to 1000 ms [54], [55]. In our simulation, we employ a simply latency model that the time for each message to transmit conforms gauss distribution of \( (\mu = 500, \sigma = 500) \) with lower cutoff of 50, we also set \( \text{timeout} = 2000 \) for the failure detector.

Since for all of the failure types, the system has the lowest tolerance on mislead attack. Thus to keep the system safe,
we need only ensure the system survive mislead attack. As a result, we only run simulations for mislead attack.

A. Without Failure

Similar to Eq. (9), to meter the property of final decision, we also define an indicator of decision as the following equation:

\[
\text{decision} = \frac{\text{abs}(d_0 - d_1)}{(d_0 + d_1)}
\]  

(17)

where \(d_0\) and \(d_1\) is the count of correct nodes which have final decision on value 0 and 1 respectively.

Simulation results of system without failure are shown in Fig. [19] where solid lines exhibit convergence and dotted lines exhibit the decision. From the figure we can have the following conclusions:

1) For each dataset, the converge progress is similar with the synchronous scenario.

2) Convergence on all datasets can reach 1 in 60 seconds.

3) All of the nodes make the same final decision within 70 seconds.

B. Mislead Attack

Since for a system with the majority of the nodes are initially in value of 0, reach consensus at 0 or 1 is totally different for mislead attack, we define signed convergence as the Eq. (18), thus only if a system survive from an attack, signed convergence will equal to convergence defined in Eq. (9).

\[
cvg = p_0 - p_1/(p_0 + p_1)
\]  

(18)

From theoretical analysis, we can conclude that for \(D = 33\), \(p_0 = 0.75\), the system can tolerate attack of \(f = 0.18\) with final signed convergence of over 0.8 which means at least 90% correct nodes are finally in value 0 according to Eq. (18). To keep safe from fluctuation, we run the simulations on the datasets with relaxed condition of \(f = 0.15\), the signed convergence is illustrated in Fig. [20] and errors of the simulation result is exhibited in Table [11] where \(\mu\) is the mean value and \(\sigma\) is the standard deviation while correct is the ratio of correct nodes that decided on 0 to all the correct nodes, confused is the ratio of confused nodes to all the correct nodes, error is the ratio of correct nodes that decided on 1 to all the correct nodes, none-eclipsed error is the ratio of correct nodes that decided on 1 but not been eclipse attacked to all the correct nodes. We can see that in all decided correct nodes(decided on 1 or 1), for all dataset, correct deciding(decided on 0) is about 92%, we can also see that uniform datasets have almost the same performance regardless their network scale.

But for the wiki datasets, we also concern the tolerant of attack by collusion of top \(n\) influential nodes, defined as the first \(n\) nodes by sorting all nodes in descending order on the count of a node’s followees. Simulation shows that for the goal of 90% correct nodes agreeing on value of 0, the algorithm can tolerant attack by 2% top nodes on the wiki dataset, as shown in Fig. [21] where the red dotted lines are the case of failed to reach the goal under attack by 3% top...

---

```python
1: function ONMESSAGE(node, msg):
2:   if msg.nodeid not in node.suspects then
3:     return
4:   if msg.state.stage < node.state.stage then
5:     return
6:   if node.state = deciding and msg.state = deciding and msg.round < node.round then
7:     return
8:   delete msg.nodeid from node.suspects
9:   add msg.nodeid to node.followees
10: function ONTIMER(node, time):
11:   delta ← node.round_start_time − time
12:   if delta < TIMEOUT then
13:     return
14:   lives ← []
15:   for all msg in node.buffer do
16:     add msg.nodeid to lives
17:   for all followee in node.followees do
18:     if followee not in lives then
19:       add followee to node.suspects
20:       delete followee from node.followees
21: function ONTRY(node):
22:   if node.state ≠ deciding then
23:     return
24:   if node.buffer.length ≠ node.followees.length then
25:     return
26:   COMPOUND(node)
```

---

Fig. 18. Failure detector algorithm

Fig. 19. Convergence of the Asynchronous Algorithm

Fig. 20. Simulation results of mislead attack by top influential nodes
nodes. Errors of the simulation result of attack by 2% top nodes is exhibited in Table III, where we can see that in all decided correct nodes (decided on 1 or 1), for all dataset, correct deciding (decided on 0) is about 96.8%.

Comparing attacks committed by random nodes and top influential nodes, we also find that more centralized trust relationships leads to more powerful ability of attack even the total numbers of trust relationships participated in attacks are the same, even in theory analysis we have already known that ability of attack depends only on the density of trust relationships for correct nodes to faulty nodes. For the wiki dataset, the total trust relationships is 33256, and for 15% random nodes, the trust relationships involved is about 5000, while for top 3% nodes, the trust relationships involved is only 2155, in contrast that the system can survive in the former but no in the later. Even exclude the factor that in the 15% random node case, lots of trust relationships are between the faulty nodes which has no effect for correct nodes, the result also supports the finding.

IX. DISCUSSION

Faulty nodes have maximum impact when they behave abnormally in one single consensus process simultaneously, thus if a node enter faulty state in different consensus process and identified timely, then they can be unfollowed and has no impact in later consensus processes. A simple strategy is that if a followee has a final decision different to the majority of other followees, it’s moved from followee list to a suspect list until it behave good again.

If the decision process of a node is public auditable, then if it’s faulty or even malicious it can be identified. For example, if in each consensus round, the broadcasted message also include the values of its followees signed with the followees private key, then one can confirm whether the node make correct decision in that round. For example, if all its followees have values of 0 in that round, if the node decide the value of 1 then it must be faulty. Also, if most of a node’s followees are suspicious, then the node itself is highly suspicious too. With followees included in the broadcasted messages, one can even build the whole directed graph of trust relationships and conduct deep analysis on it. However, including followees in the broadcasted messages will introduce a significant overhead especially when the a node has many followees.

Top influential nodes generally are most trustworthy nodes, so the chance of collusion between a number of them are much smaller than ordinary nodes with same number. However, they introduces monopoly to some extent, even it’s already much better than the case of Bitcoin. It might be a good choice to suppress their impact. For a followee, one choice is unfollowing a node if the node already has too many followees, another choice is assigning a weight w if w < 1 for that node.

Even a consensus process succeed in the sense of C3, a tiny portion of correct nodes might still decide on the wrong value, e.g. a node is eclipse attacked. Distributed oracles can be employed to help those node to ensure safety. An oracle behaves exactly the same with a regular correct node, except it does not broadcast messages during consensus process, and it only broadcast its final decision for each consensus process. Thus an oracle is actually a sink in the directed graph of trust relationships, and has no impact on the consensus process. An oracle must follow many carefully selected nodes in a variety of communities to avoid deciding on the wrong value, and it should also publish its followees for public audit. In this way, an oracle acts as a non-intrusive sample of the whole network. A node can subscribe some oracles and compare its own decision against the decisions of the oracles to detect abnormal situation.

We did not study the general relationship between graph topology and the performance of convergence and fault tolerance such as the minimum requirements for the network to converge, the convergence speed on different topology, metrics of node influence for both correct and faulty nodes etc.

Even though we studied various failures theoretically and experimentally, we did not study the case where several types of failures coexist during a consensus process.

We use a fixed value 0.5 as the ratio parameter in the compound algorithm, however, it’s shown by simulations not presented in this paper that greater ratio will also decrease the performance of fault tolerance. As we introduced in previous sections that ratio also has impact on the performance of convergence, two extreme scenarios are that when ratio = 1, the compound algorithm is degraded to the greedy algorithm,
and when ratio = 0, the compound algorithm is degraded to the simulated annealing algorithm. Studying the impact of ratio may be beneficial to maximize the performance of convergence together with fault tolerance.

Security of the DHT is also a problem though its another topic. However, DHT is only used for a node to find the correct swarm for a new followee, once the swarm is found, nodes of the swarm can connect directly through peer exchange and informations of them can be reused later. Thus even DHT is compromised, each node will still work with existing followees. Also, the trust relationships itself can be utilized for securing DHT as per existing technologies like Whanau [24].

X. Conclusion

The consensus algorithm presented in this paper is far from perfect, but it can converge in reasonable time while preserve decentralization for P2P network, even it can only tolerant collusion of 2% top influential nodes, it’s still a huge progress comparing to Bitcoin which can’t survive attack by 1 top influential node in a network with about 8000 nodes.

XI. Acknowledgement and Future Work

This study is part of the skycoin project [56] with ideas shaped by members of the consensus team. We are now developing a belief tree based algorithm which can parallel multiple consensus processes simultaneously in the sense of eventual consensus, and the algorithm is expected to reduce the average time for each consensus process.

References

[1] J. Brio and A. Castillo, Bitcoin: A Primer for Policymakers, 1st ed. Mercatus Center at George Mason University, 8 2013.
[2] J. R. Douceur, “The sybil attack,” in Revised Papers from the First International Workshop on Peer-to-Peer Systems, ser. IPTPS ’01. London, UK, UK: Springer-Verlag, 2002, pp. 251–260.
[3] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” ACM Trans. Program. Lang. Syst., vol. 4, no. 3, pp. 382–401, 1982.
[4] E. A. Alchieri, A. N. Bessani, J. Silva Fraga, and F. Greve, “Byzantine consensus with unknown participants,” in Proceedings of the 12th International Conference on Principles of Distributed Systems, ser. OPODIS ’08. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 22–40.
[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” http://www.bitcoin.org/bitcoin.pdf, 2009.
[6] D. Cawrey, “Are 51% attacks a real threat to bitcoin?”, http://www.coindesk.com/51-attacks-real-threat-bitcoin/, June 20 2014.
[7] N. T. Courtois and L. Bahack, “On subversive miner strategies and block withholding attack in bitcoin digital currency,” CoRR, vol. abs/1402.1718, 2014.
[8] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” http://snap.stanford.edu/data, Jun. 2014.
[9] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” J. ACM, vol. 27, no. 2, pp. 228–234, 1980.
[10] L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” Commun. ACM, vol. 21, no. 7, pp. 558–565, 1978.
[11] C. Attiya, D. Dolev, and J. Gil, “Asynchronous byzantine consensus,” in Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, ser. PODC ’84. New York, NY, USA: ACM, 1984, pp. 119–133.
[12] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” J. ACM, vol. 32, no. 2, pp. 374–382, 1985.
[13] L. Lamport, “Paxos made simple,” SIGACT News, vol. 32, no. 4, pp. 51–58, 2001.
[14] M. Castro and B. Liskov, “Practical byzantine fault tolerance,” in Proceedings of the Third Symposium on Operating Systems Design and Implementation, ser. OSDI ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186.
[15] M. Ben-Or, “Another advantage of free choice (extended abstract): Complete asynchronous agreement protocols,” in Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing, ser. PODC ’83. New York, NY, USA: ACM, 1983, pp. 27–30.
[16] T. D. Chandra and S. Toueg, “Unreliable failure detectors for reliable distributed systems,” J. ACM, vol. 43, no. 2, pp. 225–267, 1996.
[17] F. Greve and S. Tixeuil, “Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks,” in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, ser. DSN ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 82–91.
[18] C. Delpero-Gallet, H. Fauconnier, and A. Tielmann, “Fault-tolerant consensus in unknown and anonymous networks,” in Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems, ser. ICDCS ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 368–375.
[19] H. Moniz, N. Neves, and M. Correia, “Byzantine fault-tolerant consensus in wireless ad hoc networks,” IEEE Transactions on Mobile Computing, vol. 12, no. 12, pp. 2441–2454, 2013.
[20] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, “Secure routing for structured peer-to-peer overlay networks,” SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 299–314, 2002.
[21] R. A. Bazzi and G. Konjevod, “On the establishment of distinct identities in overlay networks,” in Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing, ser. PODC ’05. New York, NY, USA: ACM, 2005, pp. 312–320.
[22] N. Borisov, “Computational puzzles as sybil defenses,” in Proceedings of the Sixth International Conference on Peer-to-Peer Computing, ser. P2P ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 171–176.
[23] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, “SybilGuard: Defending against sybil attacks via social networks,” in Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ser. SIGCOMM ’06. New York, NY, USA: ACM, 2006, pp. 267–278.
[24] C. Lesniewski-Laas and M. F. Kaashoek, “Whanau: A sybil-proof distributed hash table,” in Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI ’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 8–8.
[25] M. N. Al-Ameen and M. Wright, “Persea: A sybil-resistant social DHT,” in Proceedings of the Third ACM Conference on Data and Application Security and Privacy, ser. CODASPY ’13. New York, NY, USA: ACM, 2013, pp. 169–172.
[26] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi, “SoK: The evolution of sybil defense via social networks,” in Proceedings of the 2013 IEEE Symposium on Security and Privacy, ser. SP ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 382–396.
[27] W. Wei, F. Xu, C. C. Tan, and Q. Li, “SybilDefender: A defense mechanism for sybil attacks in large social networks,” IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 12, pp. 2492–2502, 2013.

[28] A.-L. Barabási, R. Albert, and H. Jeong, “Mean-field theory for scale-free random networks,” Physica A: Statistical Mechanics and its Applications, vol. 272, no. 1, pp. 173–187, 1999.

[29] E. Estrada and E. Vargas-Estrada, “How peer pressure shapes consensus, leadership, and innovations in social groups,” Sci. Rep., vol. 3, 2013.

[30] D. Vilone, J. J. Ramasco, A. Sánchez, and M. S. Miguel, “Social and strategic imitation: the way to consensus,” Sci. Rep., vol. 2, 2012.

[31] C. Doyle, S. Sreenivasan, B. K. Szymanski, and G. Korniss, “Social consensus and tipping points with opinion inertia,” arXiv:1411.1723 [cond-mat, physics:physics], 11 2014.

[32] J. Xie, S. Sreenivasan, G. Korniss, W. Zhang, C. Lím, and B. Szymanski, “Social consensus through the influence of committed minorities,” Physical Review E, vol. 84, no. 1, p. 011130, 2011.

[33] R. M. Anderson and R. M. May, Infectious diseases of humans. Oxford university press Oxford, 1991, vol. 1.

[34] N. Bailey, The Mathematical Theory of Infectious Diseases and its Applications. London: Griffin, 1975.

[35] X. Castelló, A. Baronchelli, and V. Loreto, “Consensus and ordering in language dynamics,” The European Physical Journal B, vol. 71, no. 4, pp. 557–564, 2009.

[36] A. Baronchelli, “Role of feedback and broadcasting in the naming game,” Phys. Rev. E, vol. 83, no. 4, p. 046103, 2011.

[37] P. Singh, S. Sreenivasan, B. K. Szymanski, and G. Korniss, “Accelerating consensus on coevolving networks: The effect of committed individuals,” Phys. Rev. E, vol. 85, no. 4, p. 046104, 2012.

[38] J. A. K. M. A. S. A. S. B. K. A. K. G. Xie, Jierui AND Emenheiser, “Evolution of opinions on social networks in the presence of competing committed groups;" PLoS ONE, vol. 7, no. 3, p. e33215, 2012.

[39] X.-T. Liu, Z.-X. Wu, and L. Zhang, “Impact of committed individuals on vaccination behavior,” Phys. Rev. E, vol. 86, no. 5, p. 051132, 2012.

[40] M. Turalska, B. J. West, and P. Grigolini, “Role of committed minorities in times of crisis,” Sci. Rep., vol. 3, 2013.

[41] S. N. Sunny King, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” http://www.peercoin.net/assets/paper/peercoin-paper.pdf, 2012.

[42] bitshares.com, “Bitshares toolkit: Delegated proof of stake,” http://bitshares.org/documentation/group_dpos.html, 2014.

[43] nxtcrypto.org, “Nxt whitepaper,” https://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt 2014.

[44] P. Vasin, “Blackcoin’s proof-of-stake protocol v2,” http://www.blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf, 2014.

[45] I. Bentov, A. Gabizon, and A. Mizrahi, “Cryptocurrencies without proof of work,” 06 2014.

[46] V. Buterin, “On stake,” https://blog.ethereum.org/2014/07/05/stake/ 7 2014.

[47] A. Poelstra, “A treatise on altcoins,” https://download.wpsoftware.net/bitcoin/altcoins.pdf 10 2014.

[48] D. Schwartz, N. Youngs, and A. Britto, “The Ripple protocol consensus algorithm;” https://ripple.com/files/ripple__consensus__whitepaper.pdf 2014.

[49] https://ripple.com/ripple.txt

[50] J. Kim, “Safety, liveness and fault tolerance—the consensus choices stellar,” https://www.stellar.org/blog/safety_liveness_and_fault_tolerance_consensus_choice/ 2014.

[51] J. Kwon, “TenderMint: Consensus without mining,” http://tendermint.com/docs/tendermint.pdf 2014.

[52] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” science, vol. 286, no. 5439, pp. 509–512, 1999.

[53] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks,” nature, vol. 393, no. 6684, pp. 440–442, 1998.

[54] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, “Designing a DHT for low latency and high throughput,” in Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1, ser. NSDI’04. Berkeley, CA, USA: USENIX Association, 2004, pp. 7–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251175.1251182

[55] J. Li, T. Stribling, W. Morris, M. Kaashoek, and T. Gil, “A performance vs. cost framework for evaluating DHT design tradeoffs under churn,” in Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1. Mar. 2005, pp. 225–236 vol. 1.