Secure and Fair Decentralized Computing: What's Next After Blockchain

OHAD BECK
ohad.z.beck@vanderbilt.edu
Vanderbilt University April 2020

Abstract

Decentralized computing has huge potential in the near future if implemented securely and fairly. Block-chain technology was the biggest leap so far in decentralized computing which transformed markets and how people looked at currencies with the introduction of Bitcoin and Ethereum. There is still unlocked potential in decentralized computing and much to be built after block-chain. Block-chain is still able to determine an order of transactions decentrally, but requires huge amounts of energy and computational resources, and is still prone to security threats.

The ability to compute without a central database securely and fairly is a need that can be applied to broad reaching applications such as trading, medical records, power grids, internet security, etc. Hedera Hashgraph, a type of distributed ledger that implements a consensus algorithm like block-chain, was created by Swirld (Swirld owns Hedera) to be the algorithm that fills the need of secure and fair decentralized computing. Swirld claims it is one of the most secure and fastest platforms available. There is a lot of doubt about their claims and Hashgraph will never truly build on where block-chain left off. There is still a lot of work to be done to create a fast and secure algorithm for decentralized computing.

I. INTRODUCTION

Data has been stored in databases since the dawn of computing, but what if the data centers cannot be trusted due to security threats? The need for a distributed ledger stems from the need for decentralizing computations and the need for security and defense against security threats. A distributed ledger is a database that is a consensus of shared data that is spread across multiple, geographical locations without a central data storage. While a central database is a data storage in a single location that does not have a consensus of shared data spread between multiple locations. The first famous distributed ledger came about with the introduction of Bitcoin. Bitcoin is a decentralized digital currency that implements a block-chain data structure. Block-chain allows for decentralized transactions and for there to be an agreement of the transaction order, or consensus, without all the data running through a database. There are hundreds of potential applications for such an algorithm that can determine order of transactions without a central computer. Applications include cryptocurrency, medical records, trading, etc. Some of the issues with block-chain are that it is outdated, it only can compute about 3-4 transactions per second (tps), and requires a huge amount of energy due to its consensus algorithm. A consensus algorithm is the algorithm that allows the block-chain to know the order of events and "who did what, and when".

Since the advent of Bitcoin, several other distributed ledger technologies (DLTs) began to emerge. DLTs such as Hashgraph, Ethereum, Tangle, Holochain, Golem, Iexec, Economic SPace Agency, and SONM. All of these are backed by their respective startup that is pushing the need for these decentralized computing platforms. Each has their own type of consensus algorithm, and we will go over on a high level how some of the more popular algorithms work. The core of this paper will be going over how the Swirld/Hedera Hashgraph algorithm works, what applications it is suitable for, and what Swirld is promising is actually true about Hashgraph's abilities.

Proof of Work (POW): establishes consensus through an election and chooses the leader by who solves a complex mathematical problem. This in nature requires a lot of energy and computational power and hence, discourages cheating since it costs the cheater energy and computational resources to cheat the system. Bitcoin and Ethereum use such consensus algorithm.

Proof of Stake (POS): in lieu of power and work, POS picks the leader in a deterministic way depending on their wealth, or stake in the system. This allows for less power and energy consumption to reach consensus in POS. A big issue in POS is the fact that there is not a big cost to cheat the system such as in POW. Due to this, there are hybrid consensus algorithms that use a POS based approach.

Delegated POS: a hybrid consensus algorithm based off of POS, but to minimize cheating in the system, delegates are chosen and they are the only ones to be a leader, or create the next block in the system, or chain.

Practical Byzantine Fault Tolerance (pBFT): a different approach to creating a byzantine fault tolerant system [1]. It reaches consensus by always delegating a

leader and everyone communicates with the leader. The leader can always be switched if there is an issue with the leader. This leads to a much cheaper alternative to POW. There is also a finality since it does not require multiple confirmations from members in the system and everyone are equally incentivized to help the system reach consensus. There are a few issues with its security as it is prone to attack since the leader can be singled out and acts in a sense like a central database which dictates consensus.

Directed Acyclic Graph (DAG): what Hashgraph is based off but there are several other algorithms that use this sort of data structure. Tangle is one such consensus algorithm that also uses a Hashgraph. DAG is a graph that has nodes and vertices. Each vertex points to another node in the graph, like a mesh, but does not point back to previous nodes, hence acyclic. Hashgraph achieves consensus by using a DAG and concepts such as gossip about gossip, virtual voting, witnesses, and fame. Each member in the DAG sends the other members everything they know so there is never a leader such as in pBFT. Since everyone knows everything else, the members can agree on a consensus fairly and securely. The Swirld/Hedera Hashgraph algorithm is described in detail below.

II. How Hashgraph Works

A. Hashgraph Characteristics

Swirld is the parent company of Hedera. Swirld is advertising Hashgraph as the "answer to blockchains faults". Swirld says it can compute up to 500,000 tps and is Asynchronous Byzantine Fault Tolerant(aBFT) [2] – the gold standard of cybersecurity. aBFT is coined from the computer science problem most often referred to as the Byzantine General Problem. In this problem, several different armies that cannot communicate directly are attacking a city. They must all determine when to attack as the city can defend a few of the armies, but will not be able to defend themselves if most or all of the armies attack at once. The only way to communicate between the armies is by sending a messenger through the city. The messenger might get captured while travelling through the city, which poses the question: how can we verify the message is secure and to trust the messenger? Bitcoin solved the issue of Byzantine Fault Tolerant(BFT) but was still not aBFT. Asynchronous Byzantine Fault Tolerant extends BFT by being time-independent and hence, it is more fair. Being fair is a huge advantage in an algorithm and has huge upside potential.

Some of Hashgraph's disadvantages include that it is a patented technology, it is a private ledger, its source code is not open source, and because of all of this it – is not entirely decentralized. It is also hard to determine if it is truly aBFT since not enough tests have been conducted.

The Hashgraph white paper can be found here [3] and a detailed walk-through here [4]. The Hashgraph consensus algorithm seems to have originated from Michael Ben-Or's protocol delineated in his paper [5]. Ben-or delineates the consensus problem as a "set of N asynchronous processed wish to agree about a binary value. Each process P starts with a binary input x_p, and they all must decide on a common value". Ben-Or explains a mathematical backed proof of a consensus protocol that is a solution to the Byzantine General Problem. These protocols are the backing of the Hashgraph algorithm that Swirld implements.

B. Building the Hashgraph

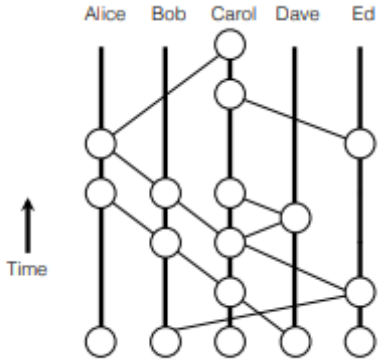

Fig. 1: [3] A Hashgraph with five members that have gossiped with each other.

The Hashgraph starts by every member creating an **event**. An **event** is a small data structure in memory and a **transaction** is a byte array. Each **event** can hold zero or more transactions. By **gossip**, each member of the community repeatedly calls others at random to sync with them to gossip everything they know with as many members as possible. A member **syncs** with other members by sending them all the **events** they know about and the other does not. By gossiping to other members about what they know, the algorithm shows a **gossip about gossip** protocol. After **syncing**, the receiving member creates a **new event** that the **sync** has happened. Then the receiver sends to the first sender all the **events** they know about but the other does not and the first sender now creates an event so they are now fully in **sync**. Then the first sender picks a new random member to **sync** up with and this continues forever, hence building a directed acyclic graph or the Hashgraph. The more members gossip, the bigger the Hashgraph becomes. This graph is connected by hashes so it is called a Hashgraph as can be seen in figure 1.

Consensus can only be achieved after the Hashgraph is built through gossip about gossip.

There is a concept of **rounds** that Ben-or created and Swirlds is using in the Hashgraph consensus algorithm. We will define this later as it can only be defined after understanding the following protocols. The first event that any member creates in a **round** is called a **witness**. For every **witness**, the community must determine if that **witness** is a **famous witness**. The famous witness is delegated to be trustworthy, but is not the leader such as in pBFT. The famous witness allows us to pick an event that is trustworthy and save computational resources so not all events need to be determined trustworthy. This can only be done in the following **rounds**. With this initial structure of the Hashgraph, consensus can be determined by the following protocols after the Hashgraph is built.

C. Achieving Consensus

For a **witness** to be famous there is an **election** and every **witness** in the **next round** must **vote** if it can **see** the **witness** that is being voted on. For a **witness** to **see** the **witness** in the **previous round**, both **witnesses** must be ancestors and descendants of each other. The **election** is not over until the **votes** are counted, which will only occur in the **next round**. So let's take a step back to see where we are in the rounds. The **original witness** that is being determined to be **famous** or not is in **round R**. Every **witness** in the **next round** that votes on their fame is in round R+1, and the witness that counts all the votes is in round R+2.

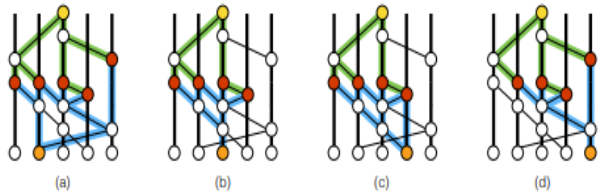

Fig. 2: [3] The yellow event can strongly see each orange event in all four scenarios, depicted a-d. The yellow event can see through a supermajority of the red events to see the orange event. In this case a supermajority is 4 since the whole integer up of 2/3 of 5 is 4.

To collect the votes, the witness in round R+2 must **strongly see**, which means the path from the witness in R+2 to the witness in R+1 must go through a **super majority** of the members. A **super majority** is more than of the members. So if there are four members, in order for the witness in R+2 to count the vote of the witness in R+1, the path from the two members must cross 3 of the member paths. Once the witness in R+2 receives a **supermajority**, or $2/3$ of yes, it decides that the witness in R is famous. A good example of strongly seeing is shown in figure 2.

Rounds are an important feature to the algorithm. They are created while the algorithm is running and not predetermined. A **round** is defined by this: an event's parents have a maximum round created M, then the event will usually be in round M as well. If that event can strongly see a supermajority of round M witnesses, then the event is defined to be round M+1, and is a witness in round M+1. In essence, an event is promoted to the next round when it can strongly see a supermajority of witnesses in the current round.

Once a round has the fame decided for all of its witnesses, the **round received** and **consensus timestamp** can be found for any of the events below. If an event can be seen (just needs to be an ancestor of the witnesses) by all the famous witnesses in a round C, that event is said to have a round received of C. The **consensus timestamp** is found after all **famous witnesses** can **see** the event. For each famous witness, find the earliest event that is an ancestor of the famous witness and a descendent of the event. Take the timestamps of all those events, sort them by their timestamp, and take the median. This median timestamp is the **consensus timestamp** for the event that all the famous witnesses can see. For every event that we find consensus for, we can sort the events into an order that all members agree on and that is the consensus order. This is also done by sorting them by round received. Ties are broken by first sorting by median timestamp, second sorting them by **extended median**, and lastly by sorting them by their **signatures** after all signatures have been XORed with a pseudorandom number (the pseudorandom number for a round received is found by XORing the signatures of all famous witnesses in that round).

D. Core Concepts Dictionary

Consensus: When most of the members agree on the order of transactions through gossiping, voting, and strongly seeing.

Gossip: repeatedly selecting another member randomly and sending them all the information they do not know.

Gossip about gossip: the Hashgraph is built from gossip from one member to the others, so gossip about gossip makes sure everyone in the system knows what everyone else knows.

Witness: the first event for a member in a specific round.

Famous witness: a witness that most of the system can see and rely on to be an accurate source of information.

Supermajority: 2/3 of the members and must always round up to determine the supermajority.

Strongly seeing: seeing through a supermajority (2/3) of the members to count the votes on determining a witness' fame.

Virtual voting: a way to determine if a witness is famous. The witness is the next round vote on if the witness in the current round is famous, and the witness two rounds from the current one collects all the votes. Everyone in the system know what the Hashgraph holds so people can determine what others should be voting making the voting virtual.

III. Swirld Hashgraph Demo Walk-Throughs

Swirld includes in their SDK a few demos showing how Hashgraph works. Below are two of the simpler demos and walk-throughs of each to help explain the algorithm.

A. Hello World Demo

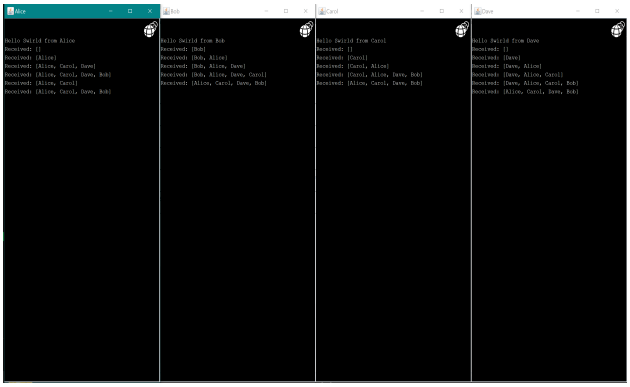

Fig. 3: The Hello World Hashgraph demo showing the order received by the corresponding members showing how they reach consensus at the end.

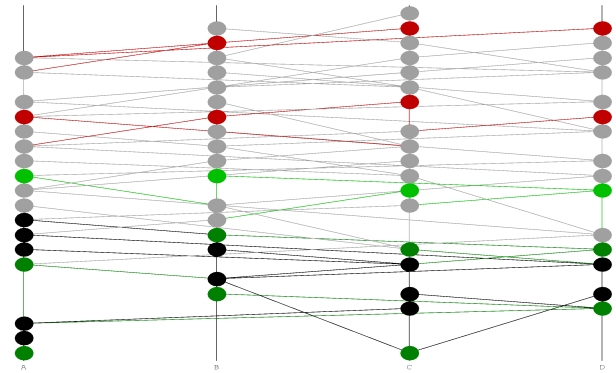

Fig. 4: The corresponding Hashgraph to the Hello World demo above.

In figures 3 and 4, the Hello World Demo is shown. Figure 3 shows the four terminals of four different members in the network. Each terminal shows line by line the current order of the messages received from other members. In the last line, they reach the same order of received. This means that they reached a consensus on the order of messages sent when the demo was started.

In figure 4 the corresponding Hashgraph is shown. The colored circles are witnesses and non-witnesses are gray or black. The darker events are part of the consensus and light are not. Famous witnesses are green, non-famous witnesses are blue (if present), and red for unknown fame.

It can be seen that it takes at least 3 rounds to determine the consensus of the first message of each member. Gossip about gossip can be seen in figure 4 as each member sends the other members at random everything they know about the order received to reach consensus between the members. A transaction in this demo is the order received on the first messages sent. Each event, or circle, show that the member has received all unknown transactions from the sending member. The light green events are the famous witnesses in round 3 that count all of the virtual votes. They can strongly see through a supermajority of the famous witnesses in round 2 to count the virtual votes. Gossip about gossip builds the Hashgraph with time. After at least 3 rounds, through the famous witnesses and an election with virtual votes, consensus can be determined on the initial order of transactions. This is the most simple graphical Hashgraph demo that Swirlds offers and shows very clearly how consensus is reached with their graphical interface options.

B. Game Demo

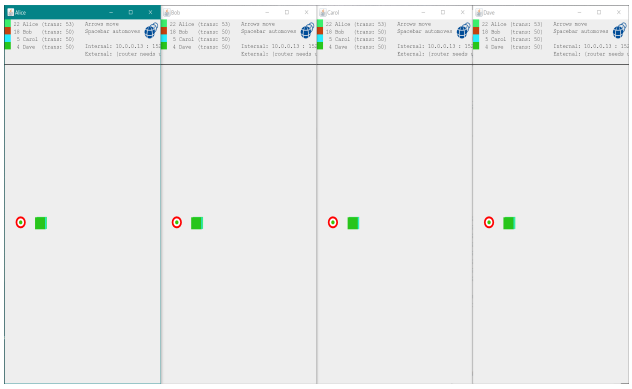

Fig. 5: The game demo graphical interface showing each member and how they continually reach consensus by reaching the target.

In figures 5 and 6, the Game Demo is shown. In this demo, instead of reaching a single consensus on the order of first messages sent by each member, it continually reaches a new consensus on the location of a goal. Each time one member moves to a goal, a new random goal is

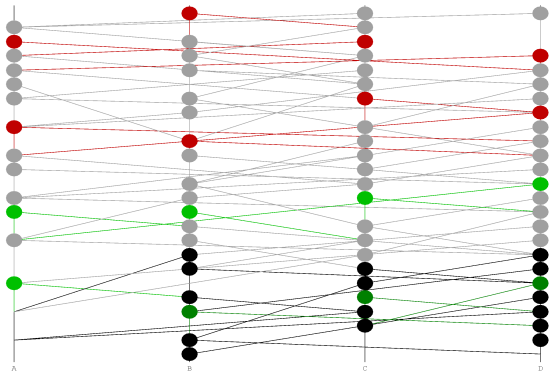

Fig. 6: The corresponding Hashgraph to the game demo above.

placed on the screen. By talking to each other and with consensus, they can know where the goal is.

The Hashgraph in this demo is a bit harder to read since there is continuous consensus happening. What is taken away from this graph is that the graph must keep growing and growing but you can only reach consensus two rounds away.

The Game Demo is more complex than Hello World but the same concepts are applied to reach consensus of the location of the goal. Gossip about gossip grows the Hashgraph with time when each members sends a random member all the events they know about, and then they sync. This allows each member to know about every other member even if they have not communicated with everyone. A transaction in this demo is the x and y coordinates of the goal location. Each event, or circle, shows that the member received all unknown coordinates from the member that sent the message. Famous witnesses are determined after two rounds succeed them. After there are at least three rounds, a consensus can be determined by an election with virtual votes. The member that reaches the goal first (random member at beginning) chooses the next goal and gossips to other members the location. Consensus takes longer in this demo since only one member knows the location and is gossiped through all members. Once they all agree on the direction they must go to reach the goal, they can move to attain the goal.

The takeaway from the game demo is how local and global lag affect the consensus. At first each member player, showed by a colorful square, are quite separate from each other. As the game goes on they are practically laid on top of each other. They are not perfectly on top of each other which accounts for the global lag. If there was no lag it would seem there would only be one square moving across the screen. This demo is a very useful tool for tracking consensus in a Hashgraph system when things are always changing and moving.

IV. Related Work

The Hashgraph SDK is available on the Hedera website: https://docs.hedera.com/guides/docs/sdks.

One sector Hashgraph could be utilized is the energy sector and securing smart grids across the country. The US Military Academy, Westpoint, and the Department of Defense have done some work in this field with their paper here [6].

There have been a few other papers explaining the trade-offs of Hashgraph [7] and [8] and another paper extending the Hashgraph algorithm [9].

There remains a lot of work to do in this space. It remains that the only proven tests of labeling Hashgraph as aBFT are the theorems created in the 1980's and the speed of Hashgraph has not been tested and verified by anyone but Swirlds.

V. Discussion

From my analysis, Hashgraph is not the solution to block-chain and does not improve all of block-chain's disadvantages. Hashgraph is not on any public ledgers and is not entirely decentralized, essentially acting as a database that does its computations decentrally, like block-chain, but is still owned like a central database like Amazon Web Services or Oracle. It does not fill the void of what block-chain is trying to fill as it is patented and is not open source. Swirld ends up owning all of the base code and gets paid when people implement and use the patented technology.

Swirld has proposed a public Hashgraph network [2] that will be a solution to a secure and fair public distributed ledger. Even though it is still public, it is overseen by for-profit corporations and due to this nature, is not an answer to a fair and public DLT. Facebook has also proposed a similar solution by introducing Libra. Both are overseen by several for-profit corporations and are just shifting the central bank figure from the Federal Reserve to a DLT owned by several corporations. None of these proposed solutions truly solve the problem of creating a fair and secure distributed ledger.

The need for Hashgraph is still there, just not the alternative to block-chain. By the nature of the data structure, it does have a very fast transaction speed so it could be used for private use for mission critical applications such as shown by Westpoint and the Department of Defense [6] for the power grid. In this regard, Hashgraph might be a good alternative, but more work needs to be done if it is really aBFT, which is one of the biggest claims Swirld is making. If it is truly aBFT, it could be used in mission critical applications and help secure private applications such as the power grid, money transactions, trading, etc. In these private applications, Hashgraph could excel due to its proposed qualities. It is not an alternative or better version of block-chain as it does not

allow fully decentralized computations that block-chain seeks to attain.

There is still a lot of work to verify this algorithm and test its proposal that it is aBFT and the true speed of the algorithm.

VI. Conclusion

Hashgraph is a step in the right direction after block-chain in some regards. It is much faster, computationally less intensive, and hopefully more secure. The fact that the algorithm is not implemented anywhere public, is still patented, and overseen by a for-profit corporation raises some doubts. The core algorithm has high potential in the realm of decentralized computing. If it were truly decentralized we might be in a new era of communication and internet security. For now, decentralized computing has much to improve.

References

[1] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[2] L. Baird, M. Harmon, and P. Madsen, "Hedera: A governing council & public hashgraph network," *The trust layer of the internet, whitepaper*, vol. 1, 2018.

[3] L. Baird, "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," *Swirlds, Inc. Technical Report SWIRLDS-TR-2016*, vol. 1, 2016.

[4] L. Baird, "Hashgraph consensus: Detailed examples," *Swirlds Tech Report. Swirlds*, 2016.

[5] M. Ben-Or, "Another advantage of free choice (extended abstract) completely asynchronous agreement protocols," in *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pp. 27–30, 1983.

[6] J. James, D. Hawthorne, K. Duncan, A. St. Leger, J. Sagisi, and M. Collins, "An experimental framework for investigating hashgraph algorithm transaction speed," in *Proceedings of the 2nd Workshop on Blockchain-enabled Networked Sensor*, pp. 15–21, 2019.

[7] P. Schueffel, "Alternative distributed ledger technologies blockchain vs. tangle vs. hashgraph-a high-level overview and comparison," *Tangle vs. Hashgraph-A High-Level Overview and Comparison (December 15, 2017)*, 2017.

[8] L. Hoxha *et al.*, "Hashgraph the future of decentralized technology and the end of blockchain," *European Journal of Formal Sciences and Engineering*, vol. 1, no. 2, pp. 29–32, 2018.

[9] M. Choe, "Open hashgraph: An ultimate blockchain engine," 2018.