Hashgraph

IBCC630E

Presented by-
IIT2015044
IIT2015086
IIT2015088

Under Supervision of -
Dr. S Venkatesan

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

.

1. Keywords and Definitions

ancestor **:** An event x is defined to be an ancestor of event y if x is y, or a parent of y, or a parent of a parent of y, and so on. It is also a self-ancestor of y if x is y, or a self-parent of y, or a self-parent of a self-parent of y and so on.

round **:** The round created number or a round of an event x is defined to be r + i, where r is the maximum round number of the parents of x (or 1 if it has no parents), and i is defined to be 1 if x can strongly see more than 2n/3 witnesses in round r (or 0 if it can't).

round received **:** the first round where all unique famous witnesses are descendants of
X.

fork **:** The pair of events (x, y) is a fork if x and y have the same creator, but neither is a self-ancestor of the other.

honest member **:** An honest member tries to sync infinitely often with every other member, creates a valid event after each sync (with hashes of the latest self-parent and other-parent), and never creates two events that are forks with each other.

to see an event **:** An event x can see event y if y is an ancestor of x, and the ancestors of x do not include a fork by the creator of y.

to strongly see an event **:** An event x can strongly see event y if x can see y and there is a set S of events by more than 2/3 of the members such that x can see every event
in S, and every event in S can see y.

witness **:** A witness is the first event created by a member in a round.

famous witness : A famous witness is a witness that has been decided to be famous by the community, using the algorithms described here. Informally, the community tends to decide that a witness is famous if many members see it by the start of the next round.

unique famous witness : It is a famous witness that does not have the same creator as any other famous witness created in the same round. In the absence of forking, each famous witness is also a unique famous witness.

consistent hashgraph : Hashgraph A and B are consistent iff for any event x contained in both hashgraphs, both contain the same set of ancestors for x, with the same parent and self-parent edges between those ancestors.

2. Methodology

The figure[1] shows a hashgraph which is growing upward over the time. Every participant in the hashgraph keeps the copy of it in memory.

In the given example, the four members Alice ,Bob, Carol and Dave are represented by the four lines labeled as A,B,C and D respectively.

Each member starts by creating an event(gray colored), which is small data structure in the memory that stores zero or more transactions. The hashgraph uses the gossip protocol ,which means each member repeatedly calls other at random to sync with them.

In the given case , Bob calls Dave by connecting over the internet and Bob sends Dave all events that Dave does not yet know. Refer figure[1], Dave creates new event , a new circle which has lines going straight down to his last event and diagonally down to Bob's last event. This shows how the members are communicating with each other.

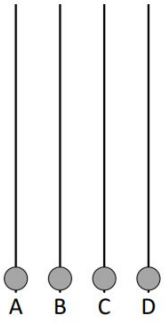

Figure 1

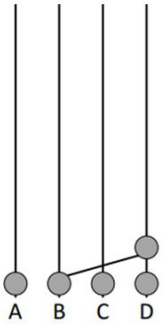

Figure 2

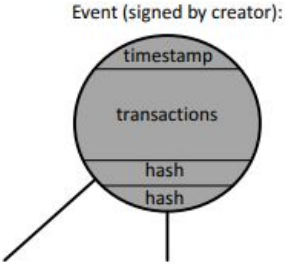

Figure 3

Refer figure[3], The event data structure contains on the top, timestamp i.e when he created the event, in middle zero or more transactions and at below two hashes first one of Bob's last event and second one of his last event.
At the time of gossiping of this event, it is sent with the digital signature. After this Dave Send his event to Bob and Bob creates same type of event and Bob further sends his events randomly to any one of the remaining members and this continues forever, growing a directed acyclic graph upwards.

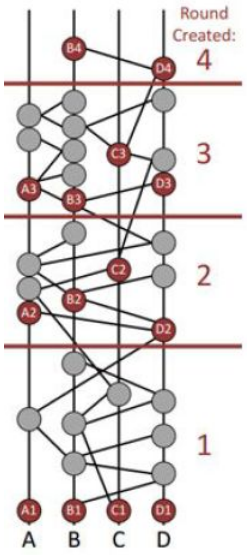

Figure 4

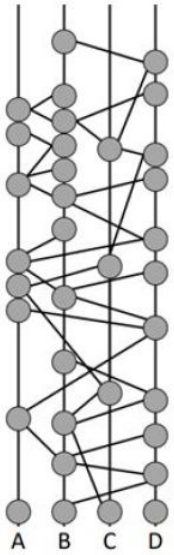

Figure 5

Refer figure[5] ,Here each event contains hashes and it is digitally signed by his creator so the entire hash values are cryptographically secure. It can always grow, but the older part are immutable, as strong as the cryptographic hashes and signature system used. This graph is connected by cryptographic hashes to each other, so it is called hashgraph.

Refer figure[4], Each event has rounds associated with it and it is created as we go upward in the graph. It is useful to define "round created' for each event. A child never has round created before its parent. So as the graph flows upwards round created can stay the same or increase. Every first event of each round of each member is witness (red colored) but it is possible that a member has no witness in the given round. Here the witnesses are labeled A2, B2, C2 and D2. For each of these witnesses we need to determine the famous witness and this is done by considering the next round witnesses A3, B3, C3 and D3. Now the election will hold for each witness to determine their fame. This event will vote for the witness event in the round below on the basis of whether they have seen them or not. To be seen, there should be downward path from the voting witness event to the round below witness event for which the election is held .

Refer figure[6]. Thus here for B2 , there is an entirely downward path from A3 to B2 means B2 is ancestor of A3 and A3 is the descendent of B2. Thus, A3 will vote "yes" for B2. Similarly B3, C3 and D3 for all four of these events will vote "yes" for B2 . Now all four events voted "yes" for B2 to be famous but the election is not ended until these votes are counted by some event. For this, the new event will be created in the next round of voters and this event will count the votes from the round below witness event . Thus, in next round B4 or D4 will count the number of votes. But for counting the votes B4 or D4 should strongly see the level below witness nodes. For strongly seeing there should be enough path through supermajority of the population which is the number more than the 2/3 rd of the replicated state machines connected to the network. Refer figure[8,9], In this case B4 strongly sees the all four witness event i.e A3,B3,C3 and D3 so he count the votes and declare node B2 as famous and color green to B2 to show that it is famous. If B4 had seen 3 "yes" and 1 "no" or 3 "yes" and no other vote , it would be still decides yes because that's super majority .

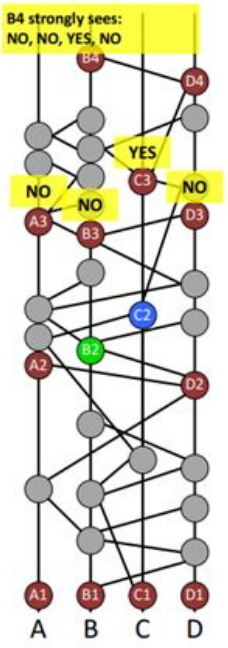

Figure 6

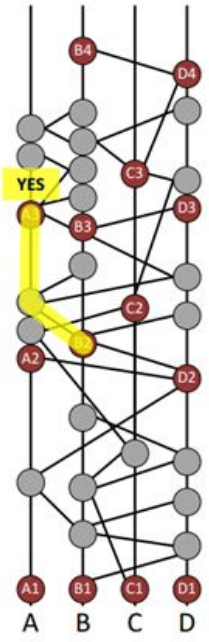

Figure 7

We need for B4 to strongly see a supermajority of witnesses, in order to even have a chance at

deciding. Therefore, we use this to define the "round created" . If an event x has parents with a maximum round created of R, then that event will usually be round R, too. But if that event can strongly see a supermajority of round R witnesses, then that event is defined to be round R+1, and so is a witness. In other words, an event is promoted to the next round when it can strongly see a supermajority of witnesses in the current round.

Refer figure[7], We can also run the election for the C2 node , but there are no downward paths from A3, B3, or D3 to C2, so they all vote "no" and C3 will vote "yes". as B4 strongly see A3,B3,C3 and D3 so it will collect the votes and these are "no" ,"no", "yes" and "no" so super majority is "no" . so C2 is not famous and color blue to C2 to show that it is not famous.

There is a theorem[] that if any witness is able to "decide" yes or no, then that is the result of the election, and it is guaranteed that all other witnesses that decide are going to decide the same way. In this example, B4 was able to decide the election. If it had collected votes that were more evenly split between YES and NO, then it would have failed to decide. In that case, we can consider D4. If D4 also fails to decide, then perhaps A4 or C4 might decide. If none of the round-4 witnesses can decide, then each of them will simply vote in accordance with the majority of the votes they collected (voting YES in case of a tie). In that case, it will be up to the round-5 witnesses to collect votes from the round-4 witnesses. Perhaps the round-5 witnesses will be able to decide. The voting continues until it eventually reaches a round where some witness can decide the election. There is a theorem[] saying that the election will eventually end (with probability one) as long as we add in a coin round every 10th round of voting. In a coin round, collecting a supermajority causes a witness to merely vote (not decide). And a non-supermajority causes it to vote pseudo randomly, by using the middle bit of its own signature as its vote.

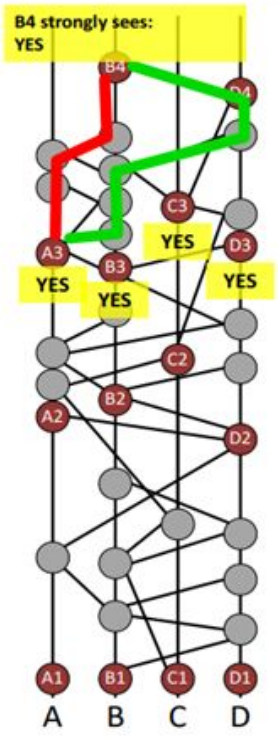

Figure 8

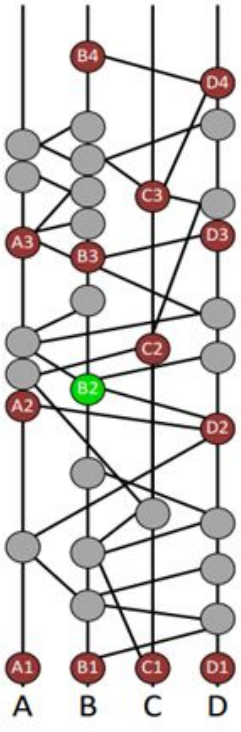

Figure 9

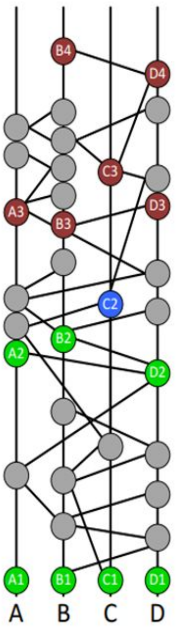

Figure 10

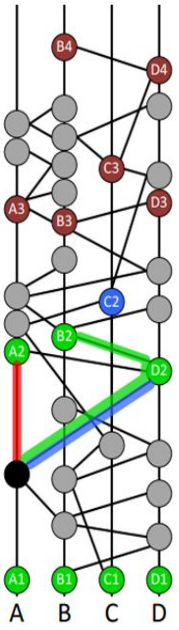

Figure 11

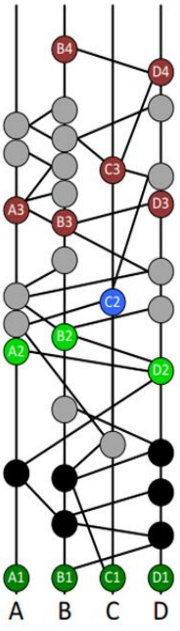

Figure 12

In normal operation, most events are not witnesses, so there is no election for most events. And most witnesses are declared famous with an almost-unanimous vote in the first round of voting. So most elections do not last very long. Refer figure[10] ,Notice that in this example, we have now decided the fame of every witness in round 2. Once a round has the fame decided for all of its witnesses, it is possible to find the round received and find the consensus timestamp for a new set of events. Start by considering the gray event immediately below A2 refer figure [11,12]. This event can be seen by every famous witness in round 2. The red, green, and blue paths show how A2, B2, and D2, respectively, can all see the black event. This merely requires seeing, not strongly seeing. This only requires seeing by the famous witnesses. It doesn't matter whether C2 can see the black event, because C2 is not famous. Since the black event is seen by all of the famous

witnesses in round 2 (but not in any earlier round), it is said to have a round received of 2.

Thus, after deciding the node is famous or not , consensus timestamp is calculated by taking the median of the timestamp of his very next event of each member but the events are sorted by their round received , if it is same for two events then consensus timestamp, extended consensus timestamp and their signature are considered for calculating the consensus timestamp.

3. Implementation

In the implementation of a similar algorithm called hashgraph, it is considered that there are *n* members, more than 2n/3 of which are honest which also implies that less than n/3 of them are dishonest. It is further assumed that the digital signatures and cryptographic hashes are secure, so signatures cannot be forged, signed messages cannot be changed without detection, and hash collisions can never be found.The syncing gossip protocol is assumed to ensure that when one node sends to another node all the events it knows, the other node accepts only those that have a valid signature and contain valid hashes corresponding to events that it has. The system is totally asynchronous. It is assumed that for any two honest members, each will eventually try to sync with the other and if one node repeatedly tries to send the other a message then it eventually succeeds.

The proposed algorithm's properties can be proven by the following lemmas and theorems.

Strongly Seeing Lemma **:** *If the pair of events (x, y) is a fork, and x is strongly seen by event z in hashgraph* A*, then y will not be strongly seen by any event in any hashgraph* B *that is consistent with* A*.*
Proof : The proof is by contradiction. Suppose event w in B can strongly see y. By the definition of strongly seeing, there must exist a set S_A of events in A that z can see, and that all can see x. There must be a set S_B of events in B that w can see, and which all see y. Then S_A must contain events created by more than 2n/3 members, and so must S_B, therefore there must be an overlap of more than n/3 members who created events in both sets. It is assumed that less than n/3 members are not honest, so there must be at least one honest member who created events in both S_A and S_B. Let m be such a member, and their events $q_A \varepsilon S_A$ and $q_B \varepsilon S_B$. Because m is honest, q_A and q_B cannot be forks with each other, so one must be the self-ancestor of the other. Without loss of generality, let q_A be the self-ancestor of q_B. The hashgraphs A and B are consistent, and qB is in B, so its ancestor q_A must also be in B. Then in B, x is an ancestor of q_A, which is an ancestor of q_B, so x is an ancestor of q_B. But y is also an ancestor of q_B. So both x and y are ancestors of q_B and are forks of each other, so qB cannot see either of them. But that contradicts the assumption that q_B can see y in B.That is a contradiction, so the lemma is proved.

Lemma : *If hashgraphs* A *and* B *are consistent and both contain event x, then both will assign the same round created number to x.*
Proof : If the consistent hashgraphs both contain x, then they both contain the same set of all its ancestors, including the first event in history. Then the proof is by induction: they agree on the round number of that first event, which is 1 by definition. And if they both contain an arbitrary state y, and agree on the round numbers of all its ancestors, then they will agree on the maximum round number r of the parents of y, and will agree on whether y can strongly see more than 2n/3 witnesses created in round r by different members, and therefore will agree on the round number of y. Therefore they will agree on the round number of all events they share, including x.

Lemma : *If hashgraphs* A *and* B *are consistent, and the algorithm running on* A *shows that a round r event by member m_0 sends a vote v_A to member m_1 in round r+1, and the algorithm running on* B *shows that a round r event by member m_0 sends a vote v_B to an event by member m1 in round r + 1, then $v_A = v_B$.*
Proof : The algorithm only sends a vote from event x to event y if y can strongly see x. It is not possible for consistent hashgraphs to have two events that are forks of each other and that are both strongly seen, by the Strongly Seeing lemma. Therefore, the two votes must be coming from the same event x in both hashgraphs. An event's vote is calculated purely as a function of its ancestors, so the two hashgraphs must agree on the vote, and $v_A = v_B$.

Lemma : *If hashgraphs A and B are consistent, and A decides a Byzantine agreement election with result v in round r and B has not decided prior to r, then B will decide v in round r + 2 or before.*

Proof : Decisions can't happen in coin rounds, so r must be a regular round. If A decides a vote v, that means some witness in round r received votes of v from a set of members S that contains more than 2n/3 members. Because voting is consistent (by the previous lemma), all other round r events in A and B will receive votes from more than 2n/3 members, a majority of whom will also be in S, because two subsets of size greater than 2n/3 drawn from a set of size n must each have a majority of their elements in common with the other. Therefore, every round r witness in both A and B will vote for v (and some may decide v). If round r +1 is a regular round, then every event in A and B in that round will receive unanimous votes of v and will decide v. If round r + 1 is a coin round, then all will receive unanimous votes of v, so none will flip coins, and all will vote v, and then all will decide v in round r + 2.

Theorem 1 **:** *For any single* YES/NO *question, consensus is achieved eventually with probability 1.*

Proof **:** If any member decides the question, then all members will decide the same way within 2 rounds, by the last lemma. So the only way consensus could fail is if no member ever decides, because no witness ever receives more than 2n/3 matching votes. However, in a coin round, if such a supermajority has not yet been achieved, then all the honest members randomly choose their vote, and will have a nonzero probability of all choosing the same vote. Coin rounds occur periodically forever, so eventually the honest members will become unanimous, with probability one, and then consensus will be reached within 2 rounds.

Lemma : *For any round number r, for any hashgraph that has at least one event in round r+3, there will be at least one witness in round r that will be decided to be famous by the consensus algorithm, and this decision will be made by every witness in round r + 3, or earlier.*

Proof : Let S_{r+3} be a set containing a single witness in round r +3, in a hashgraph that has at least one such witness. For each i < r + 3, let Si be the set of all witnesses in round i that are each strongly seen by at least one witness in S_{i+1}. It must be the case that $2n/3 < |S_i| \leq n$ for all $i \leq r + 2$, because the existence of an event in round i + 1 guarantees more than 2n/3 are strongly seen in round i, and none of the n members can create more than one witness in a given round that is strongly seen (by the Strongly Seeing lemma). Strongly seeing implies seeing, so each event in S_{r+1} sees more than two thirds of the events in S*r*. Therefore, on average, each event in Sr is seen by more than two thirds of the events in S_{r+1}. They can't all be below average, so there must be at least one event in S_r (call it x) that is seen by more than two thirds of the events in S_{r+1}.

So more than two thirds of S_{r+1} will vote YES in the election for x being famous. Therefore, every event in S_{r+2} will receive more YES votes than NO votes for the fame of x, and will therefore vote for x being famous (and may or may not decide that x is famous). Therefore, the event in S_{r+3} will receive unanimous votes for x being famous, which will cause it to decide that x is famous. Therefore, every member with an event in round r + 3 will first decide that x is famous in either round r + 2 or r + 3.

Byzantine Fault Tolerance Theorem **:** *Each event x created by an honest member will eventually be assigned a consensus position in the total order of events, with probability 1.*

Proof **:** All honest members will eventually learn of x, by the definition of honest and the assumptions that the attackers who control the internet must eventually allow any two honest members to communicate. Therefore, there will eventually be a round where all the unique famous witnesses are descendants of x. Therefore in that round, or possibly earlier, there will be a round r where all the famous witnesses are descendants of x. Then x is assigned a received round of r, and a consensus timestamp of the median of when those members first received it, and its consensus place in history will be fixed. Furthermore, it is not possible to later discover a new event y that will come before x in the consensus order. Because, to come earlier in the consensus history, y would have to have a received round less than or equal to r. That would mean that all the famous witnesses in round r must have received y. But once the set of famous witnesses is known for a round, all of their ancestors are also known, so there is no way to discover new ancestors for them in the future as the hashgraph grows. Furthermore, it isn't possible for a round to gain new famous witnesses in the future, once the famousness of all the known witnesses in that round are known. Any new round r witness that is discovered in the future will not be an ancestor of the known round r + 1 witnesses (of which there are more than

2n/3), and so the consensus will immediately be reached that it is not famous. Therefore, once an event is assigned a place in the total order, it will never change its position, neither by swapping with another known event, nor
by new events being discovered later and being inserted before it.

Given the above mentioned theorems/lemmas the following conclusions can be drawn:

The purpose of the concept of strongly seeing is to make the following lemma true.
The strongly seeing lemma is the foundation of the entire proof, because it allows for consistent voting, and for guarantees that different members will never calculate inconsistent results, even with purely virtual voting.

Different members may have slightly different hashgraphs, and so may have slightly different elections. However, all the votes will be consistent. Byzantine agreement on a particular YES/NO question is achieved by multiple rounds of virtual voting. A given member will end their election calculations in round r if it is a normal round (not a coin round) and some round r + 1 event strongly sees more than 2n/3 of the members voting the same way in round r. If that happens, then every active member will end their election in round r or r + 1 (or r + 2 if r + 1 is a coin round), and will decide the same way.

In the hashgraph consensus algorithm, Byzantine agreement is used to decide whether each witness in a given round is famous or not. Every round is guaranteed to have at least one witness that is famous.

Every round will eventually have all its witnesses classified as famous or not famous by universal consensus, with at least one of the witnesses being famous. After that, the set of famous witnesses for that round will never change, even if more events are added to the hashgraph. This set of famous witnesses can therefore act as a judge, to define a total order on all the events that have reached them, and a consensus timestamp on every event.