FORMALIZING THE HASHGRAPH GOSSIP PROTOCOL

Karl Crary
CyLab Partners Conference
September 24, 2019
Distributed shared ledger

- Consensus total order on “transactions”
  - No central authority
  - No reversal of transactions

- Applications:
  - Cryptocurrencies
  - Smart contracts
  - Digital property
  - Gaming
Scaling problem

- Bitcoin can handle ≈ 7 transactions/second.
- Ethereum can handle ≈ 15 tps.
- Paypal can handle ≈ 450 tps
- Visa can handle 56k tps (claimed)

- Proof-of-work is secure **because** it is slow.
  - To serve as a general currency, we need a completely different technology.
Byzantine consensus

- Get participants to agree on something
- Up to 1/3 can break the rules
  - Impractical: voting rounds require lots of messages
Hashgraph [Baird 2016]

- First practical algorithm
  - No proof-of-work
  - No voting messages

- Based on a gossip protocol:
  - Create event when receiving a message
    - Attach a payload of transactions
  - Track who talks to whom, in what order
  - Conduct a virtual election
    - Each event’s vote is determined by its ancestry
    - Gossip allows each participant can carry out the election independently
  - Agree on a total order on network events
Hedera Hashgraph

- A distributed ledger / cryptocurrency based on Hashgraph
- Gossip protocol’s overhead is very low:
  - Don’t send any messages you wouldn’t send anyway
  - Add a gossip payload to each message
- Throughput $\approx 10k-500k$ tps (in experiments)
Honest peers

• Never create a fork
  • X and Y are a fork if they have the same creator, and neither is an ancestor of the other

• An honest peer’s events are linearly ordered
• Over 2/3 are honest
Strongly seeing

• X sees Y if:
  • (1) X is a descendant of Y
  • (2) a technical condition holds, such that no event can see both sides of a fork

• X strongly sees Y if:
  • there exists a set of peers P such that:
    • (1) P contains over 2/3 of the peers
    • (2) for every Z in P
      • X is a descendant of Z
      • Z sees Y
Strongly-seeing lemma

• If X and X’ are a fork, they cannot both be strongly seen, even by different events

• Proof
  • Suppose Y strongly-sees X, and Z strongly-sees X’
  • Let P and P’ be the corresponding sets of peers
    • Then there exists at least one honest peer in \( P \cap P' \)
  • Let V and W be the mediating events on that honest peer
    • Then V and W are linearly ordered
    • Without loss of generality, assume V is an ancestor of W
  • Thus W sees X’ (directly) and X (through V)
  • That can’t happen
Witnesses

- Break events into *rounds*
- Each peer’s first inhabitant in a round is called a *witness*
- Advance to a new round when you can strongly-see 2/3 of the previous round’s witnesses

- Cheaters can have multiple witnesses per round
  - But – by the lemma – at most one can be strongly seen
  - The extra witnesses are irrelevant
Famous witnesses

- An event enters the consensus order when “most” peers can see it
  - Can’t say *all* peers, because some might not be talking
  - You can’t know if silent peers have seen it or not

- Use *famous witnesses*

- A witness is *famous* if most later witnesses can see it
  - Use virtual voting to determine

- Since it’s famous, nearly everyone knows about it
  - (In particular, what it can see)

- An event enters the consensus order the first round in which every famous witness is a descendent
Verification

• Formalized correctness proof using Coq.
  • (Popular proof assistant developed in France.)
• 13k lines of Coq
• Gives 100% confidence that the algorithm works, barring some flaw in the model or in Coq.
  • (Extremely unlikely.)

```plaintext
(* Lemma 5.12 *)
Lemma strongly_seeing :
  forall W x y v w, 
  member W v 
  -> member W w 
  -> fork x y 
  -> stsees x v 
  -> stsees y w 
  -> False.
Proof.
  intros W x y v w Wv Ww Hfork Hssx Hssy.
  destruct Hssx as (v' & Hv & Hmajx).
  destruct Hssy as (w' & Hv & Hmajy).
  so (supermajority_intersect_3 _#5 eq_peer_decide supermajority_honest Hmajx Hmajy) as (a & Hhonest & Hseesx & Hseesy),
  destruct Hseesx as (q & Hcrq & Hxq & Hqy'),
  destruct Hseesy as (r & Hcrn & Hyn & Hqy').
```
Ongoing and future work

- Verify the algorithm that is actually implemented.
- Verify the Hashgraph software.
  - First, develop the machinery to make it possible.
Thank you