CS425/CSE424/ECE428 — Distributed Systems — Fall 2011

BitCoin and Zooko's Triangle: Global, Distributed Time Stamping

Overview

- Two problems
  - Unforgeable electronic currency
  - Secure, globally unique names
- Same underlying principle
  - Decentralized global timestamping service

BitCoin

- Create a digital currency that is
  - Unforgeable
  - Transferrable
  - Secure
  - Decentralized
- `Traditional' e-cash:
  - Coin = Token + signature of bank
- BitCoin: eliminate the bank!

Proof of work

- Computational puzzle
  - Find x such that f(x) = y
  - f is easy to compute, hard to invert
  - f is many-to-one s.t. f(x) = y with probability p
- Find solution:
  - Try random choices for x
  - Expected running time – O(1/p)
- Verify solution
  - Compute f(x)
  - Expected running time – O(1)
- Example: f = cryptographic hash function H
  - Find x such that H(x) has k leading 0's
    - f(x) = first k bits of H [H_k], y = 0
  - Difficulty: 2^k

Scheme 1

- Coin: puzzle solution
- Forgeable, but only with computational effort
  - "Value" proportional to puzzle difficulty (2^k)
  - E.g., cost of electricity needed to "mint" new coin
- Payment protocol:
  - Alice->Bob: coin x
  - Bob: compute $H_k(x)$, verify = 0
  - Bob->Alice: goods or services

Double-spending

- Alice still has coin x after giving it to Bob!
  - Alice->Bob: coin x
  - Alice->Carol: coin x
  - Alice->David: coin x
- Traditional e-cash solution: detection after the fact
  - Bob, Carol, David deposit x into the bank
  - Bank realizes x has been double-spent, punishes Alice

BitCoin solution

- Transaction log
  - For each coin x, lists who has it
- When coin first "minted", claim it
  - Append: "Alice found x"
- During a transaction, log transfer
  - Bob verifies that Alice currently owns x
  - Appends "Alice transfers x to Bob"
    - (with proper signatures from Alice, Bob)
- Now Bob is owner of x

Global Transaction Ordering

LOG 1

1. Alice mints x
2. Alice transfers x to Bob
3. Bob transfer x to Carol
4. Alice transfer x to David — INVALID

Most recent owner: Carol

LOG 2

1. Alice mints x
2. Alice transfers x to David
3. Alice transfers x to Bob — INVALID
4. Bob transfers x to Carol — INVALID

Most recent owner: David

Solutions

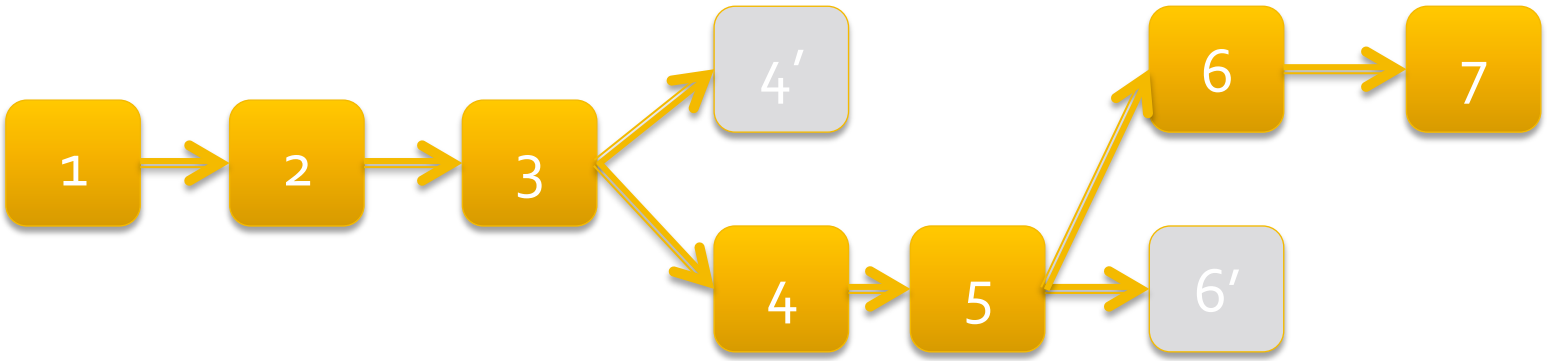
- Centralized: single log
  - Maintained by **trusted** bank
- Decentralized
  - Run Paxos on a global scale??
- Bitcoin
  - Proof of work, chains

Proof of work

- Can incorporate data (z) into puzzle
  - Find x such that H(x || z) has k o bits
- To append to log, must solve puzzle based on existing log
- Format of log "line" n: L_n = M, x, where
  - M: new message appended to log
  - x: number such that $H_k(x \| M \| L_{n-1}) = o$

Chaining

- Each line's puzzle depends on the previous one
  - L_n -> L_{n-1} -> … -> L_1 -> L_0
  - To add m lines, must solve m puzzles
- Longest chain wins



Chain Growth

- Suppose r people try to append to a log
  - Each person j has own message M_j
  - Each tries to solve $H_k(x \,||\, M_j \,||\, L_{n-1}) = 0$
- As soon a someone finds a solution, *broadcasts*† solution (L_n) to everyone
- Everyone else switches to searching for L_{n+1}
  - I.e., solve $H_k(x \,||\, M_j \,||\, L_n) = 0$
  - (why?)

† we'll return to this later

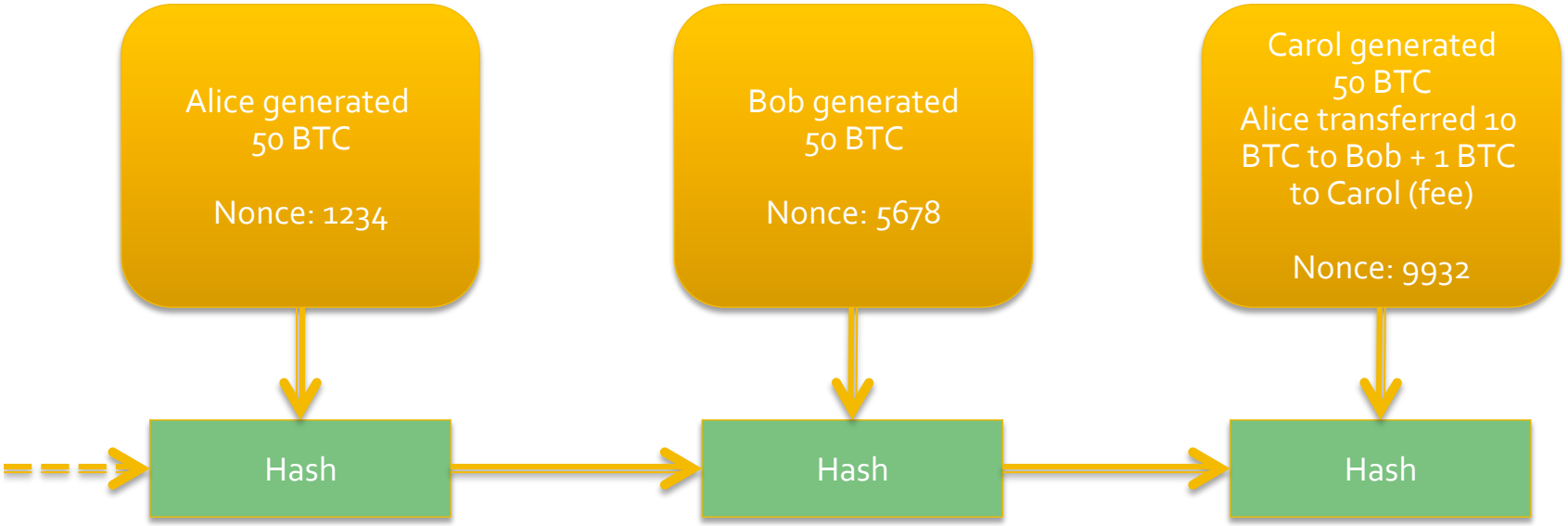
How fast does the chain grow?

- Each person expects to solve puzzle/generate new line in time t
- Among the r processes, log grows at the speed of t/r per line
  - Why?
- As more people participate
  - r grows
  - Log grows faster
  - More difficult to revise history!

Incentives for Logging

- Security better if more people participated in logging
- Incentivize users to log *others'* transactions
  - Transaction fees: pay me x% to log your data
  - Mining: each log line *creates* bitcoins
    - Replace "Alice minted x" entries with "Alice logged line L_n"
- Payment protocol:
  - Alice->Bob: here's coin x
  - *Broadcast* to everyone: Alice transfers x to Bob
  - Bob: wait until transfer appears in a new log line
    - Optionally wait until a few more lines follow it

Putting it all together



Account	Balance
Alice	39 BTC
Bob	60 BTC
Carol	51 BTC

Logging Speed

- How to set k?
  - Too short: wasted effort due to broadcast delays & chain splits
  - Too long: slows down transactions
- Periodically adjust difficulty k such that one line gets added every 10 minutes
  - Determined algorithmically based on timestamps of previous log entries
- Current difficulty
  - p = 0.000000000000002134626788616875506243708571219031000509
  - 4684659657288133 expected hash computations to win (4.7 **quadrillion!**)

Broadcast

- All-to-all broadcast
  - Every transaction (for logging)
  - Every block (for chain growth)
- How do you implement this?
  - DHT (e.g., Chord)
  - Gossip

Bandwidth

- Data volume
  - VISA network: 2000 tps
  - Transaction: 0.5 – 1KB
  - A single block (10 mins): 1.14 GB
  - Total volume ~160 GB / day
    - Or twice that if you include transaction broadcasts
- Bandwidth per node?
  - On average, each node downloads / uploads each block once
  - ~160 GB/day = 15 Mbps
  - (only ~$50/month at EC2 prices!)
  - Storage & CPU costs dominate

BitCoin Issues

- "Mining" not profitable
  - Unless you have some expensive, special-purpose rigs

  "the bulk of mining is now concentrated in a handful of huge mining pools, which theoretically could hijack the entire network if they worked in concert."
- Coins must be kept safe
  - From loss
  - From theft
- [Economics]

Naming

- Problem: How to assign names to processes?
- Solutions:
  - IP addresses
  - DNS
  - Web certificates
  - …

Desirable Features

- Secure
  - Can "claim" a name, and prove this claim to others
- Meaningful
  - Name needs to be "Nikita Borisov" or "Amazon," not "1Na7VPBzpwP5QNJk3zG9jMYXvaSzzGS3mn"
  - Sometimes called "memorable"
- Decentralized
  - Context-free: "Amazon" means the same to you as to me

Zooko's Triangle

- Conjecture: cannot have all three
  - But can have any two!
- E.g.:
  - Secure & decentralized: public keys
    - 1Na7VPBzpwP5QNJk3zG9jMYXvaSzzGS3mn uniquely, globally identifies someone, who can prove the right to have that name
  - Meaningful & decentralized: domain names
    - **Not** secure
  - Secure & meaningful: nicknames / petnames
    - *Local* mapping of names to identities
    - Can be translated to secure & decentralized names

Zooko's Tetrahedron?

- Two definitions of decentralized
  - Globally meaningful
  - No central authority
- Turns out you can have one of the two
  - E.g., X.509 certificates used for Web Browsing
  - E.g., DNSSEC

Distributed Timestamping Service

- Can solve triangle with distributed global timestamping
  - I "claim" a name by saying "Nikita Borisov" belongs to public key 1Na7VPBzpwP5QNJk3zG9jMYXvaSzzGS3mn
  - First claim wins
  - Sound familiar?

BitCoin names

- Mine names instead of coins!
- New block includes any names you want to register
  - Your own + any others you care about
- Name ownership verified by the longest chain
  - Can implement transfers, too

[originally proposed by Aaron Swartz]

Back to reality…

- WWW: use public key certificates
  - Issued by certificate authorities
  - Ensure uniqueness, trademarks, etc.
- Use *multiple* authorities
  - 100+ in existence!
  - Good: encourages competition, lowers prices
  - Bad: Any single authority can compromise security
- E.g.: "comodogate"
  - Comodo compromised issued new certificates for "mail.google.com", "login.skype.com", etc.
  - Other compromises have happened

Fixing Web Security

- Solution: add a new global ~~decentralized~~ distributed log
  - All new certificates added to log
  - Can be monitored by domain owners
- Log servers run an agreement algorithm
  - E.g., Paxos or BFT
  - Cryptographic evidence of correct behavior
  - Must compromise *all* log servers
- Proposals
  - Sovereign Keys (EFF)
  - Certificate Transparency (Google)

Summary

- Global, distributed consensus useful for many applications
  - Electronic cash
  - Naming
- Global, decentralized consensus *may* be possible
  - With some assumptions on computation, communication, …
- Centralized distributed consensus may be a good alternative