PoW results, forks part 2

2018-03-12
Tadge Dryja
today
pset02 recap
PoW analysis
more fork and non-fork types
tx replay and attacks
pset02 issues
looks like a (truncated, ahem) Pareto distribution
1 miner has >70% of the entire network power!
Exaggerated here, but these are real issues seen in PoW networks
pset02 work done
congrats to the workers
16 trillion hashes performed
prove it!
compact proof of work

Often heard, but incorrect:

"Proof of Work doesn't scale"

Actually couldn't scale better: prove $O(n)$ work in $O(1)$ time, space

Blockchains, and Bitcoin sure have scaling problems, but PoW doesn't
compact proof of work

How to prove all the work done throughout the entire pset in 1 line?
compact proof of work
How to prove all the work done throughout the entire pset in 1 line?
Show the luckiest block

hash(0000000065a211f01118fc6727661d71e6c6bf68d9f708c2116f6b1b72483675 turtle 1/654244/7105)

-> 00000000000c49a941d589d5e842032d221f9ba98a5a22f3ae13e25611f79f69
compact proof of work

How to prove all the work done throughout the entire pset in 1 line?

Show the luckiest block

hash(0000000065a211f01118fc6727661d71e6c6bf68d9f708c2116f6b1b72483675 turtle 1/654244/7105)

-> 00000000000c49a941d589d5e842032d22f9ba98a5a22f3ae13e25611f79f69

00 00 00 00 00 00 0c 49 a9 41 ...
compact proof of work
00 00 00 00 00 00 0c 49 a9 41 ...
that's 5 ½ bytes, or 44 bits
$2^{44}$ is ~17T, which is what we expect.
compact proof of work
00 00 00 00 00 00 0c 49 a9 41 ... 

Another way to look at it: 
need 33 bits, have 44, 11 bits 
of "excess" work, or 2048 
blocks. Close to the 1862 
observed.
Sending this over the wire, or storing on disk... what can we optimize here?
header optimization

First 8 chars always 0, so don't send them
header optimization

Entire prevhash can be removed, saves most of the space!
header optimization

This type of optimization is not done in Bitcoin; but would work!

If you want to, code up a PR!

(Nobody has bothered because headers are pretty quick and not a bottleneck)
forks and non-forks
continuing Neha's talk last week:

fork types: soft, hard, also, non-forks, where there is no change
non-forks
header optimization is not a fork
new nodes identify each other, omit the first 4 bytes of every block
old nodes see no change
example non-forks
internal only:
compressing blocks / utxo set on disk
faster signature verification
nobody else needs to know
example non-forks
peer non-forks: identify at connect time, default to old behavior
compact blocks
bloom filters
standardness

"non standard" txs will not be relayed, but will be accepted in a block

not-quite a soft fork, but close
standardness

"non standard" txs will not be relayed, but will be accepted in a block

not-quite a soft fork, but close
intermission
128 second break
<table>
<thead>
<tr>
<th>Hash rate</th>
<th>soft</th>
<th>hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>adopting: system halts</td>
<td>adopting: nothing changes</td>
</tr>
<tr>
<td></td>
<td>ignoring: nothing changes</td>
<td>ignoring: nothing changes</td>
</tr>
<tr>
<td>1% to 50%</td>
<td>adopting: nothing changes (orphans)</td>
<td>adopting: nothing changes</td>
</tr>
<tr>
<td></td>
<td>ignoring: nothing changes</td>
<td>ignoring: nothing changes</td>
</tr>
<tr>
<td>51% to 99%</td>
<td>adopting: nothing changes</td>
<td>adopting: nothing changes</td>
</tr>
<tr>
<td></td>
<td>ignoring: nothing changes</td>
<td>ignoring: nothing changes</td>
</tr>
<tr>
<td>100%</td>
<td>adopting: nothing changes</td>
<td>adopting: nothing changes</td>
</tr>
<tr>
<td></td>
<td>ignoring: nothing changes</td>
<td>ignoring: nothing changes</td>
</tr>
</tbody>
</table>
variant: soft & hard

example:
blocks CAN be 8MB (hard fork)
blocks MUST be 8MB (soft fork)
prevents reorgs, ensures split
heard described as "bilateral hard", "full"
# Soft & Hard Chart

<table>
<thead>
<tr>
<th>Hash rate</th>
<th>0%</th>
<th>1% to 50%</th>
<th>51% to 99%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Soft</strong></td>
<td>adopting: system halts</td>
<td>adopting: split off new rule</td>
<td>adopting: split off new rule</td>
<td>adopting: split off new rule</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>ignoring: nothing changes</td>
<td>ignoring: slow blocks</td>
<td>ignoring: slow blocks</td>
<td>ignoring: system halts</td>
</tr>
<tr>
<td><strong>Hard</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
variant: firm fork /evil fork
a hard (&soft) fork, that looks like a soft fork to non-adopting nodes
variant: firm fork / evil fork
a hard (& soft) fork, that looks like a soft fork to non-adopting nodes
PoW for new chain is an empty block in the old chain!
<table>
<thead>
<tr>
<th>Hash rate adopting / fork type</th>
<th>0%</th>
<th>1% to 50%</th>
<th>51% to 99%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evil fork</td>
<td>adopting: system halts</td>
<td>adopting: split off new rule</td>
<td>adopting: split off new rule</td>
<td>adopting: split off new rule</td>
</tr>
<tr>
<td></td>
<td>ignoring: nothing changes</td>
<td>ignoring: slow blocks</td>
<td>ignoring: system halts (empty blocks</td>
<td>ignoring: system halts (empty blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>forever)</td>
<td>forever)</td>
</tr>
</tbody>
</table>
evil fork
seen by some as the best way to hard fork
others don't want miners to know they can do this
seems coercive, thus "evil"
fork coordination

BIP9: miners signal soft fork adoption in header when 95% adopt, fork rule activates

probably deprecated. "governence"
transaction replay
split happens (minority soft fork, majority hard fork, or any full fork)
make tx on old chain
what happens on new chain?
transaction replay
in many cases, the tx happens on BOTH chains
if valid on both, someone will relay it
this can be messy!
transaction replay
split coins: merge with mined coins (diverges)
spam double spends
try exploiting locktime deltas
expensive, ugly, but possible
transaction replay problems
want to sell one, not the other
many users unaware of forks
may unknowingly send both
replay attack on exchange
network split to coinA, coinB
exchange only runs coinB
replay attack on exchange
network split to coinA, coinB
exchange only runs coinB
user: deposit coinB
exchange: you have coinB
user: withdraw coinB
replay attack on exchange
user: withdraw coinB
exchange: here's coinB (&coinA)
user: split utxos
user: deposit coinB
(GOTO top)
replay attack on exchange
this has happened
not saying this is obvious, but
there were warnings
consensus changes are hard integrated feature and bug you want coins to stay put you might not want new features but new features can help a lot miners have outsize influence?
consensus changes are hard small coins, changes are pretty easy: call up exchanges, miners
Bitcoin, very messy. Future fork methods unknown.
Stay tuned.