

mas.s62

# lecture 10

PoW results, forks part 2

2018-03-12

Tadge Dryja

schedule stuff

was at bitcoiny meetings (cool stuff)

pset03 out before midnight tonight

office hours tomorrow and next tues

pset03 due next wed, 21st

today

pset02 recap

PoW analysis

more fork and non-fork types

tx replay and attacks

pset02 recap  
seemed like fun!

from Feb 20 to Mar 01, 1862 blocks

mean 418 sec / block, about 7 min per  
block

15994458210304 work performed (16TH)

$1862 * 2^{33}$

# pset02 scores

#0	1337	turtle
#1	150	U+1F4A9
#2	138	bsnowden3
#3	45	iron
#4	35	yasushi
#5	21	anthony-3
#6	17	anthony-6
#7	17	anthony-5
#8	16	kezike17
#9	16	anthony-shedua
#10	14	fortenforge
#11	7	ihg
#12	6	ipHKS
#13	5	joechung
#14	5	Ihssanremote
#15	4	tsaengja
#16	4	jtol
#17	3	ShangyanLi
#18	2	tomriddle

#19	2	joechung2
#20	2	bfeeser
#21	2	Moin
#22	2	mamram
#23	2	AlanBidart
#24	1	joechung10
#25	1	BCHisBitcoin
#26	1	faraaaaaaz
#27	1	Thalita
#28	1	joechung0
#29	1	BMahon9
#30	1	MCCLM
#31	1	ekoto4
#32	1	averylwuzhere
#33	1	joechung1

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)
```

```
-> 000000000000c49a941d589d5e842032d221f9ba98a5a22f3ae13e25611f79f69
```

# 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
```

00 00 00 00 00 0c 49 a9 41 ...

compact proof of work

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 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.

# header optimization

```
000000007f1f75507526524972bd4a666ea2ee20899a1feb5bb5de095d687993 kezike17 1012907732
000000002fae9c508791febcbcc0e1e9daa5413ad1852dfe1a2b3da60695bba9 tomriddle LcGCkMKWys
00000000486dff5b17f76839e0e5073c1efccacf0db08a97af243bbb662c604b Thalita 6bkcnQAAAAA=
00000000057949f7b54e90dea28e32580b2d440f240be9e6258aeb81d7717cd3 AlanBidart 804030736
000000007dadcf0730175689c9d4bd2c389d08f99ebf8fad95332ad1f1a2eb1c ShangyanLi jaojqcgvMP
```

Sending this over the wire, or  
storing on disk... what can we  
optimize here?

# header optimization

```
000000007f1f75507526524972bd4a666ea2ee20899a1feb5bb5de095d687993 kezike17 1012907732
000000002fae9c508791febcbcc0e1e9daa5413ad1852dfe1a2b3da60695bba9 tomriddle LcGCKMKWys
00000000486dff5b17f76839e0e5073c1efccacf0db08a97af243bbb662c604b Thalita 6bkcnQAAAAA=
00000000057949f7b54e90dea28e32580b2d440f240be9e6258aeb81d7717cd3 AlanBidart 804030736
000000007dadcf0730175689c9d4bd2c389d08f99ebf8fad95332ad1f1a2eb1c ShangyanLi jaojqcgvMP
```

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

# header optimization

```
000000007f1f75507526524972bd4a666ea2ee20899a1feb5bb5de095d687993 kezike17 1012907732
000000002fae9c508791febcbcc0e1e9daa5413ad1852dfe1a2b3da60695bba9 tomriddle LcGCKMKWys
00000000486dff5b17f76839e0e5073c1efccacf0db08a97af243bbb662c604b Thalita 6bkcnQAAAAA=
00000000057949f7b54e90dea28e32580b2d440f240be9e6258aeb81d7717cd3 AlanBidart 804030736
000000007dadcf0730175689c9d4bd2c389d08f99ebf8fad95332ad1f1a2eb1c ShangyanLi jaojqcgvMP
```

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



# soft / hard chart

Hash rate adopting / fork type	0%	1% to 50%	51% to 99%	100%
Soft	adopting: system halts	adopting: split off new rule	adopting &	adopting &
	ignoring: nothing changes	ignoring: slow blocks	ignoring: new rule	ignoring: new rule
Hard	adopting: nothing changes	adopting: nothing changes (orphans)	adopting: split off new rule	adopting: new rule
	ignoring: nothing changes	ignoring: nothing changes	ignoring: slow blocks	ignoring: system halts

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

Hash rate adopting / fork type	0%	1% to 50%	51% to 99%	100%
Soft AND Hard	<p>adopting: system halts</p> <p>ignoring: nothing changes</p>	<p>adopting: split off new rule</p> <p>ignoring: slow blocks</p>	<p>adopting: split off new rule</p> <p>ignoring: slow blocks</p>	<p>adopting: split off new rule</p> <p>ignoring: system halts</p>

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!

# evil fork chart

Hash rate adopting / fork type	0%	1% to 50%	51% to 99%	100%
Evil fork	<p>adopting: system halts</p> <p>ignoring: nothing changes</p>	<p>adopting: <b>split off</b> new rule</p> <p>ignoring: slow blocks</p>	<p>adopting: <b>split off</b> new rule</p> <p>ignoring: system halts (empty blocks forever)</p>	<p>adopting: <b>split off</b> new rule</p> <p>ignoring: system halts (empty blocks forever)</p>

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.

"governance"



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.