Hello

OK – I've spoken at TabConf every single time – -- and I went back and checked --

~Teaser~

And every time I talk at TabConf -- -- the topic usually becomes *something real* – that ends up being worth *100s of millions of dollars*. So -- for example in 2018, at the top center there, I spoke on prediction markets and Futarchy, in 2018 , that was building on earlier code I had written back in 2013 and technical blog posts I had written in 2015. Today this Futarchy idea is all the rage in some Altcoin communities – Augur/Gnosis , those were multimillion dollar projects back in 2014/2015 – (long before anyone had ever heard of Blockstream or Lightning Labs), now PolyMarket is huge – more than one-point-five billion dollars in volume last month), and apparently Solana is big into Futarchy now.

In 2019, I was here talking about what I called "sidechain leeching", today this is called cross-chain M-E-V (or just M-E-V)… and my solution, that I came up with back in May 2016/ Jan 2017, is now called producer-builder separation. These are – this is worth like millions of dollars per day in the Ethereum community, for example.

In 2021 I presented about drivechain, and then I did the debate with Peter last year. Now there are a few Altcoin versions of it – some were implemented very badly , and lost a lot of money, or were just part of some Altcoin scam or whatever. But there was one Ethereum-like Drivechain-thing , and it just raised one-hundred million dollars from a16z. And the merged mining revenues from Drivechain I think could eventually be in the hundreds of billions of dollars per year.

So, with all that in mind [*gesture*], I am now going to give a talk about soft forks – building on posts and talks in the past, name a talk I gave at MIT last year.

~Agenda~

Ok so here's the agenda. Intro – we're almost done with that – soft forks, history, governance , "soft forks over time", and lastly CUSF – the Core Untouched Soft Fork , c-u-s-f – then questions.

~About Me~

But before we do that, one slide about me – and where I work.

I am the author of BIPs 300/301 – collectively called "drivechain". I run a technical bitcoin blog called "truthcoin.info". I am the founder and CEO of "LayerTwo Labs" – "making every transaction a bitcoin txn" [**gesture**] -- which was created Dec 2022, but which has already produced the following pieces of software: we have our own testnet – we have a CUSF activator

for Bip300/301, and for OP CAT also BIp 347 version, we have a sidechain L2 that emulates zCash which is like a million times better than coinjoin, for privacy -- we have an L2 that emulates Ethereum called “EthSide” – basically just Ethereum on bitcoin. We have basically a custom version of Namecoin on Bitcoin – called “BitNames”. We have an L2 for tokens and token trading called BitAssets --- we have a set of largeblock sidechains that can scale to 8 billion users tomorrow, that I call “thunder”, and we have a new GUI for BitcoinCore that, I call “BitWindow”. Long ago I was a statistican in the Yale Econ department – I’m from academia, originally.

I also have a lot of strong opinions, my view is that – all these on the right – Lightning ARK CoinJoin DLCs BitcoinCore itself – are a waste of time, and we should just ignore them – no one will have heard of them in 5 years.

Ok enough about that.

~Review~ **(5 : 00)**

First, let us review the basics. Hopefully all of you already know this.

Now, actually, what's up here on this slide is not perfectly accurate. As we will see.

- but it is the conventional wisdom. So it is a good starting point.

The conventional wisdom --- is that soft forks "tighten" the rules, and hard forks "loosen" the rules.

A 2nd, competing definition, is that the hard fork requires the user to update their software, whereas the soft fork does not. That's a lot closer to the truth.

Anyway, here are some soft forks you should be familiar with:

August 2010, Satoshi took a bunch of messages that were previously broadcast-able in Bitcoin's blockchain, and he banned them. This tightened the rules ... and also regular users did not need to update immediately. Right? Satoshi banned a bunch of stuff, but no one was using it.

Sept 2010 -- Satoshi adds the blocksize limit, of 1 Megabyte. Shrinking the blocksize, tightens the rules. Soft fork. --- Expanding it, is loosening the rules. Thats the bottom.

Then we started adding new features. P2SH enabled multisig (among other things). Check Lock Time Verify allows you to .. post-date a check basically. SegWit helped lay the foundation for the Lightning Network.

~History~

Ok that was the basics.

Now we will discuss 4 things are NOT widely known:

1) Gavin called them "changes" – not “Forks”
2) Why is that important
3) How the Fork term arose
4) And why it actually makes sense after all.

Why it was used for upgrades…

~Gavin~

Ok, Gavin Andresen, who of course was the "leader" of the Bitcoin project after Satoshi left in 2010. He wrote this here.

This is very important: Note -- there are soft rule changes and there are hard rule

changes. ....Now what I want to point out here is.

~Forbidden~

...when Gavin says "at this point", he is referring to June 29, 2012! Quite a long time ago. Now "impossible to roll out 'hard changes'". Ironically, Gavin would later attempt the “hard fork blocksize increase” change, and fail. Exactly as he predicted he would, here. But I want to convey this idea that something is forbidden on grounds of impracticality. It is as if a law of physics prevents it. Money has network effects, the participants don't want a split, and so hardfork changes just can't be done.

~Fork~

Now a tiny detour. Why is it called "fork"?

I wrote a blog post long ago, with this image. You can see the culinary fork, the tuning fork, and the fork-in-the-road. They all have a split. Yet a soft fork has no split. And a network that upgrades via hard fork, doesn't have a split either! There s no permanent split into two equal branches.

So why call it a fork, if there's no *visual* fork.

~Definitions~

Believe it or not, there's a very satisfying answer to this question. We go back to the origin of the term itself. This post, from Nov 2012. Five months after Gavin's remark (that I showed you on Github moments ago).

~The 4 Definitions~

Now, let me read these at you.

~slide~

First though I want to point out that even Adam Back and Luke Dashjr disagree on what a hard vs soft fork is, which is why I'm boring you with this! I hope it helps you. These are the original, logically consistent definitions. – Top tier. In the Q&A we may hear some weird ones – but these are the only logically consistent and original definitions of these words – and these words describe *the problem being solved* and *in that way* they are *fundamental*.

~slide~

OK. "quote:"

When there are a sufficient number of bitcoin clients on the network that disagree on the rules about how blocks are created and recorded in the blockchain. It leads to a split in the chain, one set of bitcoin clients follow one branch and another set follows the other. To fix hard forks some action must be taken by us.

Orphaned Blocks -- whenever a soft fork or hard fork occurs, the blockchain is split into two paths. One of these chains will eventually be considered the valid one, and the other will be the invalid chain. Blocks that are in an invalid chain are called orphan blocks.

Soft Fork -- at the end, it says: these kinds of forks will solve themselves without any intervention from us.

So you see? Splitting is a *problem* --- --- it means that some blocks that are valid today, will later become invalidated, which is BAD! We want synchronization. Tight synchronization always.

Forking is a problem. The hard fork and the soft fork, are *problems* to be solved. Both. The hard fork is hard problem, because *we* have to *do something*. -- --- The soft fork is not as big of a problem, since it will *eventually* work itself out. Automatically.

That's why """fork""" is a good word, after all. – If you keep in mind that fork = problem. Problematic disagreement among nodes.

~slide~

Ok finally the logic.

A soft fork will resolve itself, as we just said.

[** read slide**]

Now this is a soft fork done by Peter Todd, and actually it is a great example because the English text in the BIP is very short and clear. I've got to hand it to Peter for this one. Very short. Possibly you can see for yourself the soft fork.

Governance **(16:00)**

~Definition~

Ok, what does governance mean? Merriam Webster, will give you this – but it doesn't help us in any way whatsoever.

"... overseeing the control and direction of ... something". That's what it says.

Bitcoin is "peer to peer" -- everyone an equal peer. So there is no separation of people into "governors" and "the governed".

~slide~

But this definition is bad in another way: it is not technical. We are here at TabConf - - which is very technical place. -- So it would be better to have a very cut-and-dry, black-and-white, *technical* definition of governance. -- How do we know that we are doing governance well? When is it going better , or worse?

No *objective function*, in the webster definition.

~slide~

So here it is, I'll give you a better definition.

~Technical Governance~

Governance equals finding today's node software. Governance is an answer to the question: where does the node software come from? What process? In that sense, it is more like an industrial process -- like how do we make steel. Or a recipe. How do we build this bridge?

Which code is fullnode-code? And which code is not.

How do we tell Bitcoin Nodes from non-nodes?

If there is a dispute, then who is correct? And who is incorrect?

We can restate this as the problem of *meta-consensus*: consensus about consensus. Your full node does consensus, yes... but only after you find that software and run it.

So, you can call it "pre-consensus" if you like. How to find the consensus software. If you didn't have a node, then *how would you get one*?

~NodeFinding Strategies~

There are three main philosophies for solving this problem.

Number one -- go to bitcoin.org , and run the latest version.

Luke Dashjr proudly advocates this strategy, as does Mike Hearn believe it or not.

~Hearn & op Ver~

Here he says "you're the decider". [**gesture**] in the left top middle there.

There was an old opcode call OP VER, that literally made every upgrade mandatory. If you just published some software with a new version number, it put you on a completely different network basically. Satoshi added this, then removed it, so he seems to have thought it was in the wrong direction. This op ver is the opposite of strategy three. Which we will get to.

~back~

The second strategy is characterized by extreme anxiety. The adherents really like bitcoin consensus, and so they are very annoyed that meta-consensus spoils the fun. It rains on their parade. So these people say: find an old version, run that software, and then try not to update. *Resist the updates*, as they are all ZERO benefit + potential problems. If it 'aint broke, don't fix it.

~green-bitcoin-foundation~

Mircea Popescu was a famous adherent of this point of view. These people ran a version of Bitcoin that was ... 0.5.4. And they have many followers today. Michael Saylor -- absolutely has this view, he believes that if we do nothing, Bitcoin will be worth a lot of money, every change is a risk, so resist the changes. Most of the "toxic" crowd, today, has this view.

~return~

The third strategy was the mainstream, I think, until the SegWit War of 2017, and the failed SegWit2x revolt. Those events made soft forks much more ... polarizing. You're either with us or your against us!

But this view originally meant ...

~blue arrow~

...that, we could all peacefully coexist. Since a line of soft forks can coexist, and it *kind of* doesn't matter. Run whatever software you like, as long as it is on the line.

~return~

The other thing that SegWit2x did was -- after it failed, Bitcoiners concluded that the reason for BTC's success was that it had successfully resisted change. Thus the second strategy rose in prominence.

...

~Governance Table~

Ok, now I will explain problems with each of those three strategies.

First, what is the problem with the strategy of always going to Bitcoin.org and downloading the software there? Bitcoin.org might be hacked, or compromised. But it is more fundamental than that – this “solution” just passes the buck to someone else. It’s an infinite regress, after all , how did Bitcoin.org get the right version. If we make a mistake, while grabbing the software from there – how would we even know. So it’s kind of a non-answer.

~fill 2~

So, here's the obvious stuff. Blindly outsourcing your thinking to bitcoin.org, isn't great. Sticking with one piece of software is a little better. Best of all is when you can choose from a bunch viable soft-wares. You can experiment with them – its very hard for someone to screw up one version that you are stuck with.

~add green square~

Ok, the left collum, the problem of expertise.

~expertise is mandatory~

But Luke-Jr's position, is that you must run the latest version of the software. And you learn about what that version is, by participating in the technical community. This has obvious problems: learning takes effort --- for some people prohibitive effort. (Not everyone can devote their life to the minutia of Bitcoin technical debate.). So laypeople are kicked out. And there will always be more laypeople than experts, by definition. So that's bad.

Another problem is that there is no accumulation of recognizability. You can't put out one thing, and allow the public to become more familiar with it over time. So like with cars we have turn signals, speed limits. Stop signs. It's a lot at first, but since it doesn't change, it is possible for each person to learn it -- in their own time.

~add left columns~

So a static protocol is easier for laypeople.

...

Why does soft fork coexistence only do OK? Why is it merely orange?

Well,

~fork diagram~

if you do two different soft forks, at the same time, then this actually equals a hard fork.

So... ~yellow square~

...the soft forks must be in a line. In order to co-exist.

~Bitcoiners Often Disagree~

Furthermore, Bitcoiners disagree about everything, so even getting people to agree on this line is difficult.

~circled innovation~

Now, the problem of Innovation. Very straight-forward. Well, there's no limit to how good of an idea people can come up with tomorrow. Innovation is inherently unpredictable – and it is inherently something that everyone doesn't know. Usually, the better an innovation is, the more people hate it at first! So we want those innovations to make it into our project. If we can't, then those innovations will launch as altcoins, and potentially destroy the project. It's innovation and adoption that will either make-or-break BTC. And honestly, adoption is just one kind of innovation.

~Fill 3rd column~

So this is the last column.

~slide to fill~

Now, actually, there is a way of getting all the things we want, at once…

~add 4th row~

… it is via sidechains. It is – different people running different software. – different people want different things, you have big L1 consensus, and optional L2s. Scaling in layers – where each upper is *optional* and also *functional*.

I will talk more about that in Part 4.

~slide~ **(26:00)**

But for now, Part 3.

Ok, what I've done here, is I've taken every fork every done, according to BitMEX. Not me. And the reason I did that, was so I could make this slide, with no accusations of bias, or whatever. So here it is...

~slide~

Now, this is the number of soft forks in a given year. – 8 years on the top row, then 8 years on the bottom row. Now from 2009-2016 , we did at least 1 soft fork per year, and in fact we averaged two per year.

In the latter period, we have only done 2 total.

Furthermore, I took the liberty of tracking down when *each* was first announced, then coded, and then activated. As you can see, SegWit took 20 months from when it was first announced as a concept, until when it activated. Taproot took 46 months.

So -- if we were to naively extrapolate, then the next soft fork would take like 9 years or something.

~Problems~

[**read slide**]

~CUSF~ **(31: 00)**

Ok Part 4 – my new .. invention, or idea.

~abstract~

Visit the website for this, bip300cusf.com . And the idea is to *soft fork via 2nd piece of software*.

[**read slide**]

~image~

So, here a diagram. Remember Peter Todd's CLTV. The red there, up top, is the script interpreter failing under new conditions. That was slide 16.

[**read remainder of slides**]