Changes without unanimous consent

Anthony Towns

Scaling Bitcoin, 2017

Outline

Chain splits

- What's this talk really about?
  - *Consensus* changes without unanimous consent
  - Consensus changes without consensus?
  - Really about: chain split

Chain splits

- What's this talk really about?
  - *Consensus* changes without unanimous consent
  - Consensus changes without consensus?
  - Really about: chain split

Chain splits

- What's this talk really about?
  - *Consensus* changes without unanimous consent
  - Consensus changes without consensus?
  - Really about: chain split

Chain splits

- What's this talk really about?
  - *Consensus* changes without unanimous consent
  - Consensus changes without consensus?
  - Really about: chain split

Unanimous consent

- If you do have unanimous consent, everything works great!
  - Developers are happy to update their software
    - Changes are clearly specified, and make sense
  - People running nodes are happy to deploy it
    - No security holes, upgrade challenges, extra costs
  - Miners are happy to deploy and signal
    - No hits to profit, no PoW on old chain, no split!
  - Economy is happy to maintain and increase value
    - Step 4: ... PROFIT

Unanimous consent

- If you do have unanimous consent, everything works great!
  - Developers are happy to update their software
    - Changes are clearly specified, and make sense
  - People running nodes are happy to deploy it
    - No security holes, upgrade challenges, extra costs
  - Miners are happy to deploy and signal
    - No hits to profit, no PoW on old chain, no split!
  - Economy is happy to maintain and increase value
    - Step 4: ... PROFIT

Unanimous consent

- If you do have unanimous consent, everything works great!
  - Developers are happy to update their software
    - Changes are clearly specified, and make sense
  - People running nodes are happy to deploy it
    - No security holes, upgrade challenges, extra costs
  - Miners are happy to deploy and signal
    - No hits to profit, no PoW on old chain, no split!
  - Economy is happy to maintain and increase value
    - Step 4: ... PROFIT

Unanimous consent

- If you do have unanimous consent, everything works great!
  - Developers are happy to update their software
    - Changes are clearly specified, and make sense
  - People running nodes are happy to deploy it
    - No security holes, upgrade challenges, extra costs
  - Miners are happy to deploy and signal
    - No hits to profit, no PoW on old chain, no split!
  - Economy is happy to maintain and increase value
    - Step 4: ... PROFIT

Unanimous consent

- If you do have unanimous consent, everything works great!
  - Developers are happy to update their software
    - Changes are clearly specified, and make sense
  - People running nodes are happy to deploy it
    - No security holes, upgrade challenges, extra costs
  - Miners are happy to deploy and signal
    - No hits to profit, no PoW on old chain, no split!
  - Economy is happy to maintain and increase value
    - Step 4: ... PROFIT

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Scaling and disagreements

- But that only works if everyone agrees, and growth makes agreement less likely
  - Disagreements over goals
    - eg, government buys into Bitcoin, then wants to make it hard for criminals to use by reducing anonymitiy
  - Unclear what the impact of a change will be
    - Perfect knowledge might imply consent, but what if some people just don't see it?
  - Non-Pareto improvements
    - What if someone is actually made worse off? Perhaps an upgrade makes some mining hardware less efficient...
  - Implementation bugs
    - More developers = more bugs? More developers = more bugs *found* ?
  - Stategic disagreements

Splits are cheap

- Splitting is cheap
  - change a few lines of code
  - change the proof of work rules
- No matter how undesirable it is, you can't stop it.
- So saying "Bitcoin is great as is – let's not change anything" isn't a solution either.

Splits are cheap

- Splitting is cheap
  - change a few lines of code
  - change the proof of work rules
- No matter how undesirable it is, you can't stop it.
- So saying "Bitcoin is great as is – let's not change anything" isn't a solution either.

Outline

Easy changes

- *Uncontroversial* soft-forks
- Simple, *uncontroversial*, emergency hard-fork
- Long-buried, *uncontroversial*, hard fork
- All of these work great!

Everything else

- Contentious hard-forks
  - People want to maintain unupgraded chain
  - → SPLIT
- Quick hard-forks
  - People don't have time to upgrade
  - Un-upgraded nodes run un-upgraded chain
  - → SPLIT
- Contentious User-activated Soft-forks
  - Un-upgraded chain remains viable
  - People still want it
  - → SPLIT

Everything else

- Contentious hard-forks
  - People want to maintain unupgraded chain
  - → SPLIT
- Quick hard-forks
  - People don't have time to upgrade
  - Un-upgraded nodes run un-upgraded chain
  - → SPLIT
- Contentious User-activated Soft-forks
  - Un-upgraded chain remains viable
  - People still want it
  - → SPLIT

Everything else

- Contentious hard-forks
  - People want to maintain unupgraded chain
  - → SPLIT
- Quick hard-forks
  - People don't have time to upgrade
  - Un-upgraded nodes run un-upgraded chain
  - → SPLIT
- Contentious User-activated Soft-forks
  - Un-upgraded chain remains viable
  - People still want it
  - → SPLIT

Who decides what we get

- Core developers?
  - If it's controversial, devs will disagree too
  - And they don't want to decide anyway
- Miners?
  - If everyone decides to defer to them, sure! (BIP9, etc)
  - If not, probably not
    - We'll get more data in a couple of weeks!
- Nodes?
  - Nope, way too easy to replace them

Who decides what we get

- Core developers?
  - If it's controversial, devs will disagree too
  - And they don't want to decide anyway
- Miners?
  - If everyone decides to defer to them, sure! (BIP9, etc)
  - If not, probably not
    - We'll get more data in a couple of weeks!
- Nodes?
  - Nope, way too easy to replace them

Who decides what we get

- Core developers?
  - If it's controversial, devs will disagree too
  - And they don't want to decide anyway
- Miners?
  - If everyone decides to defer to them, sure! (BIP9, etc)
  - If not, probably not
    - We'll get more data in a couple of weeks!
- Nodes?
  - Nope, way too easy to replace them

Who decides what we get

- Core developers?
  - If it's controversial, devs will disagree too
  - And they don't want to decide anyway
- Miners?
  - If everyone decides to defer to them, sure! (BIP9, etc)
  - If not, probably not
    - We'll get more data in a couple of weeks!
- Nodes?
  - Nope, way too easy to replace them

Who decides what we get

- The Economy
  - Provides the reason devs work
    - Either philosophically, or the paycheque
  - Pays miners
    - Block rewards only let you pay for electricity if Bitcoin has value

How does the economy decide

- The economy gives a value for Bitcoin
- That is, will trade Bitcoin for goods and services
  - Buy Bitcoin — give goods/services, get Bitcoin
  - Sell Bitcoin — give Bitcoin, get goods/services
- One way or another, the economy wants some sort of market in order to exercise its power

Outline

Change, or don't

- What are we talking about?
- Someone proposes a change
  - ...to consensus rules
  - Specific and explicit about what changes
- Everyone adopts the change
  - Release new versions of software
  - Update nodes
  - Care about who owns how much according to the new rules
- Or nobody adopts the change, and stick with the current rules
- Or some people do and some don't

Price formula

- Someone proposes a change, what's the expected value of the coin now?

$$c = p_{\mathcal{N}} \cdot a + p_{\mathcal{E}} \cdot b + p_{\mathcal{S}} (\alpha + \beta)$$

Trading coins

- Can gain pricing information by trading coins (atomic swaps, BitFinex markets, etc)
- Three types of markets:
  - Unconditional: someone gets coins on old chain, other person gets coins on changed chain
  - Split or refund: trade only takes place if a split occurs (refund if there's only one chain, whichever that is)
  - Activation or refund: trade only takes place if the changed rules activate (refund if only the old chain works)

Trading coins

- Can gain pricing information by trading coins (atomic swaps, BitFinex markets, etc)
- Three types of markets:
  - Unconditional: someone gets coins on old chain, other person gets coins on changed chain
  - Split or refund: trade only takes place if a split occurs (refund if there's only one chain, whichever that is)
  - Activation or refund: trade only takes place if the changed rules activate (refund if only the old chain works)

Trading coins

- Can gain pricing information by trading coins (atomic swaps, BitFinex markets, etc)
- Three types of markets:
  - Unconditional: someone gets coins on old chain, other person gets coins on changed chain
  - Split or refund: trade only takes place if a split occurs (refund if there's only one chain, whichever that is)
  - Activation or refund: trade only takes place if the changed rules activate (refund if only the old chain works)

Trading coins

- Can gain pricing information by trading coins (atomic swaps, BitFinex markets, etc)
- Three types of markets:
  - Unconditional: someone gets coins on old chain, other person gets coins on changed chain
  - Split or refund: trade only takes place if a split occurs (refund if there's only one chain, whichever that is)
  - Activation or refund: trade only takes place if the changed rules activate (refund if only the old chain works)

Trading coins

- Can gain pricing information by trading coins (atomic swaps, BitFinex markets, etc)
- Three types of markets:
  - Unconditional: someone gets coins on old chain, other person gets coins on changed chain
  - Split or refund: trade only takes place if a split occurs (refund if there's only one chain, whichever that is)
  - Activation or refund: trade only takes place if the changed rules activate (refund if only the old chain works)

But!

- But three markets and our expected value equation[1] gives us five equations...
- ...in seven unknowns
- So this is only enough to give us values for
  - $(p_{\mathscr{N}} \cdot a)$, $(p_{\mathscr{E}} \cdot b)$, $(p_{\mathscr{S}} \cdot \alpha)$ and $(p_{\mathscr{S}} \cdot \beta)$
- But is one of those figures low because:
  - that chain would not be very valuable?
  - or just because it's not likely to exist?

[1] and $p_{\mathscr{N}} + p_{\mathscr{E}} + p_{\mathscr{S}} = 1$

But!

- But three markets and our expected value equation[1] gives us five equations...
- ...in seven unknowns
- So this is only enough to give us values for
  - $(p_{\mathscr{N}} \cdot a)$, $(p_{\mathscr{E}} \cdot b)$, $(p_{\mathscr{S}} \cdot \alpha)$ and $(p_{\mathscr{S}} \cdot \beta)$
- But is one of those figures low because:
  - that chain would not be very valuable?
  - or just because it's not likely to exist?

[1] and $p_{\mathscr{N}} + p_{\mathscr{E}} + p_{\mathscr{S}} = 1$

But!

- But three markets and our expected value equation[1] gives us five equations...
- ...in seven unknowns
- So this is only enough to give us values for
  - $(p_{\mathscr{N}} \cdot a)$, $(p_{\mathscr{E}} \cdot b)$, $(p_{\mathscr{S}} \cdot \alpha)$ and $(p_{\mathscr{S}} \cdot \beta)$
- But is one of those figures low because:
  - that chain would not be very valuable?
  - or just because it's not likely to exist?

[1] and $p_{\mathscr{N}} + p_{\mathscr{E}} + p_{\mathscr{S}} = 1$

Prediction market

- A prediction market can solve this and give values for the probabilities
- But not if it's denominated in Bitcoin
  - ...and probably not if it's in any other cryptocurrency
  - (unless it's one that's not correlated with Bitcoin's value)

Lack of price discovery

- What happens if you don't have these sorts of markets?
- Suppose everyone knows that a split is coming: $p_{\mathscr{S}} = 1$ and $c = \alpha + \beta$
- But there isn't a good market price for α and β (or $p_{\mathscr{S}}\alpha$ and $p_{\mathscr{S}}\beta$)
- Then different people can have different values for α and β — $c = \alpha_1 + \beta_1 = \alpha_2 + \beta_2$ with $\alpha_1 > \alpha_2$ and $\beta_2 > \beta_1$
- And after the split, people with higher values for α will buy the old chain and vice-versa
  - Leading to a price rise — $c' = \alpha_1 + \beta_2 > c$
  - ... like we got when Bitcoin Cash forked

Lack of price discovery

- What happens if you don't have these sorts of markets?
- Suppose everyone knows that a split is coming: $p_{\mathscr{S}} = 1$ and $c = \alpha + \beta$
- But there isn't a good market price for α and β (or $p_{\mathscr{S}}\alpha$ and $p_{\mathscr{S}}\beta$)
- Then different people can have different values for α and β — $c = \alpha_1 + \beta_1 = \alpha_2 + \beta_2$ with $\alpha_1 > \alpha_2$ and $\beta_2 > \beta_1$
- And after the split, people with higher values for α will buy the old chain and vice-versa
  - Leading to a price rise — $c' = \alpha_1 + \beta_2 > c$
  - ... like we got when Bitcoin Cash forked

Lack of price discovery

- What happens if you don't have these sorts of markets?
- Suppose everyone knows that a split is coming: $p_{\mathscr{S}} = 1$ and $c = \alpha + \beta$
- But there isn't a good market price for α and β (or $p_{\mathscr{S}}\alpha$ and $p_{\mathscr{S}}\beta$)
- Then different people can have different values for α and β — $c = \alpha_1 + \beta_1 = \alpha_2 + \beta_2$ with $\alpha_1 > \alpha_2$ and $\beta_2 > \beta_1$
- And after the split, people with higher values for α will buy the old chain and vice-versa
  - Leading to a price rise — $c' = \alpha_1 + \beta_2 > c$
  - ... like we got when Bitcoin Cash forked

Lack of price discovery

- What happens if you don't have these sorts of markets?
- Suppose everyone knows that a split is coming: $p_{\mathscr{S}} = 1$ and $c = \alpha + \beta$
- But there isn't a good market price for α and β (or $p_{\mathscr{S}}\alpha$ and $p_{\mathscr{S}}\beta$)
- Then different people can have different values for α and β — $c = \alpha_1 + \beta_1 = \alpha_2 + \beta_2$ with $\alpha_1 > \alpha_2$ and $\beta_2 > \beta_1$
- And after the split, people with higher values for α will buy the old chain and vice-versa
  - Leading to a price rise — $c' = \alpha_1 + \beta_2 > c$
  - ... like we got when Bitcoin Cash forked

Outline

Splitting is expensive

- Splitting the chain has lots of negative externalities
  - Updating wallets, miners, node software
  - P2P confusion
  - Miners need to choose which chain to mine
  - Exchanges need to add new tokens, futures, ...
  - Dumb contracts have to be updated
  - People have to pay on-chain fees to rebalance
  - People get confused

Splitting is expensive

- Maintaining the PoW algorithm is expensive
  - and also somewhat quantifiable!
- The first blocks until retarget have to be mined at old difficulty on both chains
  - receiving about $2016 \times 12.5 \times (\alpha + \beta)$ in value for $2016 \times 2 \times d$ work
  - vs $2016 \times 12.5 \times a$ value for $2016 \times 1 \times d$ work
- For a given amount of hashpower, potentially a loss of an entire two week's mining revenue

Subsidising miners is expensive

- If the PoW rules aren't changed, miners will strongly prefer the higher valued chain
- So to sustain the lower value chain until the value vs difficulty ratios equalise, subsidies are needed
  - eg, transactions paying higher fees, off-book payments to miners, miners not optimising for short-term profit
- These aren't cheap — can cost over 20,000 BTC if one chain is worth less than a quarter of the other

Replay protection

- First step to making chain splits not horrible: prevent replay
- Ideally, do this generically, so that neither chain has to admit to "causing" the split by implementing replay protection
- Ideally, get it implemented in core, so that whenever someone random causes a chain split, everyone gets replay protection for free
- If selling coins is easy, coins causing split have low expected value, so splits aren't profitable, and happen less?

Replay protection

- First step to making chain splits not horrible: prevent replay
- Ideally, do this generically, so that neither chain has to admit to "causing" the split by implementing replay protection
- Ideally, get it implemented in core, so that whenever someone random causes a chain split, everyone gets replay protection for free
- If selling coins is easy, coins causing split have low expected value, so splits aren't profitable, and happen less?

Replay protection

- First step to making chain splits not horrible: prevent replay
- Ideally, do this generically, so that neither chain has to admit to "causing" the split by implementing replay protection
- Ideally, get it implemented in core, so that whenever someone random causes a chain split, everyone gets replay protection for free
- If selling coins is easy, coins causing split have low expected value, so splits aren't profitable, and happen less?

Replay protection

- First step to making chain splits not horrible: prevent replay
- Ideally, do this generically, so that neither chain has to admit to "causing" the split by implementing replay protection
- Ideally, get it implemented in core, so that whenever someone random causes a chain split, everyone gets replay protection for free
- If selling coins is easy, coins causing split have low expected value, so splits aren't profitable, and happen less?

Transactions commit to block history

- An obvious way of preventing replay is for transactions to commit to a particular block being in the history.
- BIP 115 proposes OP_CHECKBLOCKATHEIGHT
- Side benefit: makes recovering from some double spends easier, even without consensus changes
- Has the disadvantage that you need to explcitly specify the block hash (or at least the ending bytes thereof)
- Requires two transactions to actually split the coin

Signatures commit to block history

- Instead of having an opcode, have a SIGHASH_BLOCKCOMMIT flag.
- Allow specifying a block as part of the signature
  - 2 byte nHashOffset in the signature, nLockTime from the transaction,
  - block height is `nLockTime` − `nHashOffset`
- Add the given block height's hash when calculating the hash to sign (as well as nHashOffset)
- If nHashOffset is zero, use the genesis block to make locking a transaction to testnet or litecoin easy too

Signatures commit to block history

- Instead of having an opcode, have a SIGHASH_BLOCKCOMMIT flag.
- Allow specifying a block as part of the signature
  - 2 byte nHashOffset in the signature, nLockTime from the transaction,
  - block height is `nLockTime` − `nHashOffset`
- Add the given block height's hash when calculating the hash to sign (as well as nHashOffset)
- If nHashOffset is zero, use the genesis block to make locking a transaction to testnet or litecoin easy too

Signatures commit to block history

- Instead of having an opcode, have a SIGHASH_BLOCKCOMMIT flag.
- Allow specifying a block as part of the signature
  - 2 byte nHashOffset in the signature, nLockTime from the transaction,
  - block height is `nLockTime` − `nHashOffset`
- Add the given block height's hash when calculating the hash to sign (as well as nHashOffset)
- If nHashOffset is zero, use the genesis block to make locking a transaction to testnet or litecoin easy too

Signatures commit to block history

- Instead of having an opcode, have a SIGHASH_BLOCKCOMMIT flag.
- Allow specifying a block as part of the signature
  - 2 byte nHashOffset in the signature, nLockTime from the transaction,
  - block height is `nLockTime` − `nHashOffset`
- Add the given block height's hash when calculating the hash to sign (as well as nHashOffset)
- If nHashOffset is zero, use the genesis block to make locking a transaction to testnet or litecoin easy too

Benefits

- Can handle a chain split with just two pieces of information:
  - The height at which the chain split
  - The hash of the first block on your preferred chain
- Replay protection: just always commit to that block (or one after it) when signing transactions
- Wipeout protection: checkpoint that block, and don't it to be reorged
- Easy to do even with SPV/light clients

Additions

- BIP 115 proposes only verifying block history back to about 52000 blocks
- This way clients don't need to have even the complete set of block headers available to verify signatures
- Can more or less duplicate this by allowing the signature to specify the block hash explicitly:
  - Add the block hash to the signature, an extra 32 (or fewer) bytes of witness data
  - Require the specified block hash to match the actual block hash at the given height (if known)
  - If the block being referenced is 52000+ blocks deep require the signature to specify the block hash

Additions

- BIP 115 proposes only verifying block history back to about 52000 blocks
- This way clients don't need to have even the complete set of block headers available to verify signatures
- Can more or less duplicate this by allowing the signature to specify the block hash explicitly:
  - Add the block hash to the signature, an extra 32 (or fewer) bytes of witness data
  - Require the specified block hash to match the actual block hash at the given height (if known)
  - If the block being referenced is 52000+ blocks deep require the signature to specify the block hash

Additions

- What happens if there's an orphan block?
- Maybe some transactions were signed depending on the block and become invalid
  - What if those transactions paid you, and you already spent them? Argh.
- So perhaps add a rule:
  - $\texttt{nHashOffset} > 100$ — consensus rule, transactions are only invalidated if there's a huge reorg or there's a consensus split; **OR**
  - $\texttt{nLockTime} - \texttt{nHashOffset} + 100 < \texttt{tip}$ — standardness rule, transactions in mempool won't be invalidated but transactions in a block might be

Replay protection $\neq$ Price discovery

- That's great for replay protection
- But it doesn't really let you do price discovery in advance of a split.
- You can't commit to a trade until the first forking blocks are mined

Commit to a BIP

- Instead of committing to a block hash, commit to a BIP's activation status
- Same approach:
  - SIGHASH_BIPCOMMIT flag
  - Need a couple of bytes to specify a BIP
  - Also need a bit to specify whether the BIP should be active in inactive
- Does require implementations to have a BIP number assigned, and does require them to code that BIP number in their implementation.
  - But segwit2x doesn't have a BIP.
  - Well, they have BIP 102.

Commit to a BIP

- Instead of committing to a block hash, commit to a BIP's activation status
- Same approach:
  - SIGHASH_BIPCOMMIT flag
  - Need a couple of bytes to specify a BIP
  - Also need a bit to specify whether the BIP should be active in inactive
- Does require implementations to have a BIP number assigned, and does require them to code that BIP number in their implementation.
  - But segwit2x doesn't have a BIP.
  - Well, they have BIP 102.

Commit to a BIP

- Instead of committing to a block hash, commit to a BIP's activation status
- Same approach:
  - SIGHASH_BIPCOMMIT flag
  - Need a couple of bytes to specify a BIP
  - Also need a bit to specify whether the BIP should be active in inactive
- Does require implementations to have a BIP number assigned, and does require them to code that BIP number in their implementation.
  - But segwit2x doesn't have a BIP.
    - Well, they have BIP102...

Commit to a BIP

- Instead of committing to a block hash, commit to a BIP's activation status
- Same approach:
  - SIGHASH_BIPCOMMIT flag
  - Need a couple of bytes to specify a BIP
  - Also need a bit to specify whether the BIP should be active in inactive
- Does require implementations to have a BIP number assigned, and does require them to code that BIP number in their implementation.
  - But segwit2x doesn't have a BIP.
    - Well, they have BIP102...

Commit to a BIP

- Instead of committing to a block hash, commit to a BIP's activation status
- Same approach:
  - SIGHASH_BIPCOMMIT flag
  - Need a couple of bytes to specify a BIP
  - Also need a bit to specify whether the BIP should be active in inactive
- Does require implementations to have a BIP number assigned, and does require them to code that BIP number in their implementation.
  - But segwit2x doesn't have a BIP.
    - Well, they have BIP102...

Making this soft-fork compatible

- But what about soft-fork upgrades?
- Version 0.19 comes out with BIP365 via UASF.
- Everyone agrees that BIP365 support is essential.
  - Market valuation: cost of a pizza will be 20,000 non-BIP365 coins!
- BIP365 is activated.
- You make a transaction signed with SIGHASH_BIPCOMMIT 365 active.
- If someone is still running 0.18 do they see your transaction as valid?
  - No? Then it's not a soft-fork
  - Yes? How does 0.18 know BIP 365 is active when it wasn't even written yet?

Making this soft-fork compatible

- But what about soft-fork upgrades?
- Version 0.19 comes out with BIP365 via UASF.
- Everyone agrees that BIP365 support is essential.
  - Market valuation: cost of a pizza will be 20,000 non-BIP365 coins!
- BIP365 is activated.
- You make a transaction signed with SIGHASH_BIPCOMMIT 365 active.
- If someone is still running 0.18 do they see your transaction as valid?
  - No? Then it's not a soft-fork
  - Yes? How does 0.18 know BIP 365 is active when it wasn't even written yet?

Making this soft-fork compatible

- But what about soft-fork upgrades?
- Version 0.19 comes out with BIP365 via UASF.
- Everyone agrees that BIP365 support is essential.
  - Market valuation: cost of a pizza will be 20,000 non-BIP365 coins!
- BIP365 is activated.
- You make a transaction signed with SIGHASH_BIPCOMMIT 365 active.
- If someone is still running 0.18 do they see your transaction as valid?
  - No? Then it's not a soft-fork
  - Yes? How does 0.18 know BIP 365 is active when it wasn't even written yet?

Making this soft-fork compatible

- But what about soft-fork upgrades?
- Version 0.19 comes out with BIP365 via UASF.
- Everyone agrees that BIP365 support is essential.
  - Market valuation: cost of a pizza will be 20,000 non-BIP365 coins!
- BIP365 is activated.
- You make a transaction signed with SIGHASH_BIPCOMMIT 365 active.
- If someone is still running 0.18 do they see your transaction as valid?
  - No? Then it's not a soft-fork
  - Yes? How does 0.18 know BIP 365 is active when it wasn't even written yet?

Making this soft-fork compatible

- But what about soft-fork upgrades?
- Version 0.19 comes out with BIP365 via UASF.
- Everyone agrees that BIP365 support is essential.
  - Market valuation: cost of a pizza will be 20,000 non-BIP365 coins!
- BIP365 is activated.
- You make a transaction signed with SIGHASH_BIPCOMMIT 365 active.
- If someone is still running 0.18 do they see your transaction as valid?
  - No? Then it's not a soft-fork
  - Yes? How does 0.18 know BIP 365 is active when it wasn't even written yet?

Still able to be messed with

- If an implementation knowns (and implements) the BIP's rules, everything is fine.
- If it doesn't, it needs to track unknown BIPs by what signatures they see:
  - If a block includes a signature saying a BIP is activated, then
    - no other transaction in the block can assert it's inactive
    - no transaction in any later block can assert it's inactive
  - If a block includes a signature saying a BIP is inactive, then
    - no other transaction in the same block can want it to be active
- But this would let miners confuse things:
  - BIP 720 is written and sounds good to miners, but isn't implemented anywhere
  - Miners mine a few transactions with SIGHASH_BIPCOMMIT BIP720 active
  - When implementations come out, trying to avoid BIP 720 forces a huge reorg

Still able to be messed with

- If an implementation knowns (and implements) the BIP's rules, everything is fine.
- If it doesn't, it needs to track unknown BIPs by what signatures they see:
  - If a block includes a signature saying a BIP is activated, then
    - no other transaction in the block can assert it's inactive
    - no transaction in any later block can assert it's inactive
  - If a block includes a signature saying a BIP is inactive, then
    - no other transaction in the same block can want it to be active
- But this would let miners confuse things:
  - BIP 720 is written and sounds good to miners, but isn't implemented anywhere
  - Miners mine a few transactions with SIGHASH_BIPCOMMIT BIP720 active
  - When implementations come out, trying to avoid BIP 720 forces a huge reorg

Still able to be messed with

- If an implementation knowns (and implements) the BIP's rules, everything is fine.
- If it doesn't, it needs to track unknown BIPs by what signatures they see:
  - If a block includes a signature saying a BIP is activated, then
    - no other transaction in the block can assert it's inactive
    - no transaction in any later block can assert it's inactive
  - If a block includes a signature saying a BIP is inactive, then
    - no other transaction in the same block can want it to be active
- But this would let miners confuse things:
  - BIP 720 is written and sounds good to miners, but isn't implemented anywhere
  - Miners mine a few transactions with SIGHASH_BIPCOMMIT BIP720 active
  - When implementations come out, trying to avoid BIP 720 forces a huge reorg

Limited protection

- Can fix this by having implementations update regularly, and forbid activation of unknown BIPs while they're current.
- 0.18 comes out: for six months, it rejects any block that contains a transaction with a signature requiring any unknown BIP to be activated; but then relaxes this rule.
- Six months after 0.18 comes out, 0.19 comes out: for six months, it similarly rejects commitments to unknown BIPs being active
- And so on.

Limited protection

- Can fix this by having implementations update regularly, and forbid activation of unknown BIPs while they're current.
- 0.18 comes out: for six months, it rejects any block that contains a transaction with a signature requiring any unknown BIP to be activated; but then relaxes this rule.
- Six months after 0.18 comes out, 0.19 comes out: for six months, it similarly rejects commitments to unknown BIPs being active
- And so on.

Limited protection

- Can fix this by having implementations update regularly, and forbid activation of unknown BIPs while they're current.
- 0.18 comes out: for six months, it rejects any block that contains a transaction with a signature requiring any unknown BIP to be activated; but then relaxes this rule.
- Six months after 0.18 comes out, 0.19 comes out: for six months, it similarly rejects commitments to unknown BIPs being active
- And so on.

Limited protection

- Properties:
  - Unknown BIPs are rejected by current versions of node software at all times
  - New BIPs are allowed by the latest updates (which know about them) and older versions (because they've relaxed the rules)
  - Software releases must be somewhat regular
    - Though only need a minor release bumping the timeout
  - Soft-fork deployments must have at least a six month period between specification and explanation
    - Hard-forks can happen at any time, though

Limited protection

- Properties:
  - Unknown BIPs are rejected by current versions of node software at all times
  - New BIPs are allowed by the latest updates (which know about them) and older versions (because they've relaxed the rules)
  - Software releases must be somewhat regular
    - Though only need a minor release bumping the timeout
  - Soft-fork deployments must have at least a six month period between specification and explanation
    - Hard-forks can happen at any time, though

Limited protection

- Properties:
  - Unknown BIPs are rejected by current versions of node software at all times
  - New BIPs are allowed by the latest updates (which know about them) and older versions (because they've relaxed the rules)
  - Software releases must be somewhat regular
    - Though only need a minor release bumping the timeout
  - Soft-fork deployments must have at least a six month period between specification and explanation
    - Hard-forks can happen at any time, though

Limited protection

- Properties:
  - Unknown BIPs are rejected by current versions of node software at all times
  - New BIPs are allowed by the latest updates (which know about them) and older versions (because they've relaxed the rules)
  - Software releases must be somewhat regular
    - Though only need a minor release bumping the timeout
  - Soft-fork deployments must have at least a six month period between specification and explanation
    - Hard-forks can happen at any time, though

Price discovery via BIP commitment

- This is enough to establish Bitcoin-vs-Bitcoin markets
- Which is enough to establish conditional valuations (ie, $p_{\mathscr{N}}a$, $p_{\mathscr{E}}b$, etc).
- It can be done mostly trustlessly
- Markets offering a refund need some way to distinguish whether alternative consensus rules have activated or a chain split has occurred:
  - Trusted oracle
  - Crypto proof of split/activation
  - Economic incentive — each participant puts up a ransom, r, which they lose if they lie
    - Provided the other chain is worth $f(r)\%$ of this coin's value, cheating isn't profitable
- No need for a trusted exchange, however!

Conclusion

- Consensus on consensus is hard, and getting harder.
- Splits are easy.
- We can make splits hurt less.
- We can let the economy make better decisions on splits.
- Thanks for listening!