Note: This paper is undergoing a peer review, and there is a chance you might be reading an earlier version of it. To make sure you have the latest version available, visit https://zazzylabs.com/papers/superspace.pdf. Current revision: v0.1 (Draft).

superspace /ˈsuːpəspeɪs/ *noun*
1.1 a space of infinitely many dimensions postulated to contain actual space–time and all possible spaces.

— Oxford Dictionary

Superspace: Scaling Bitcoin Beyond SegWit

Steve Kallisteiros
kallisteiros@zazzylabs.com
zazzylabs.com

August 25, 2018

Abstract

SegWit, or Segregated Witness, a soft fork successfully activated mid-2017 on Bitcoin blockchain, provided among other benefits a backward-compatible increase to the block size limit, all while not introducing consensus breaking changes to Bitcoin protocol and not resulting in a hard fork network partitioning. The potential space increase is estimated at being close to 2 MB total for practical purposes. In this paper, the author argues that it is possible, using the same mechanism, increase the block size further up to any agreed-upon limit, while still providing the same security guarantees SegWit does.

1 Introduction

As Bitcoin [1] has been reaching wider adoption, it had to confront a problem of not having sufficient space to confirm all users transactions fast enough due to the initially imposed limit of 1 MB of data per block and the mechanisms keeping time between blocks at close to 10 minutes on average. The area of research dedicated to increasing the transaction throughput has become referred to as "scalability," or "scaling" problem, similarly to its use in computer science; several solutions to this problem have since been proposed.

While many seem to agree that indefinitely increasing the block size limit would ultimately result in the maximum centralization of control at best, and entire network's denial of service at worst, there has been an ongoing debate on whether it is still acceptable to allow for a reasonable increase of block size limit until off-chain solutions, referred to as "Layer 2 solutions", are developed and have become production-ready, such as Lightning Network and others. The debate seems to be centered around the cost of such an increase, and whether the benefits exceed the cost. [2]

2 To fork or not to fork

Due to a decentralized nature of Bitcoin, some changes to the protocol rules are breaking changes, incompatible with previous rules, and when imposed, unless there is 100% consensus, and everybody upgrades the clients to the new version, the network suffers partitioning, whereby nodes begin to disagree on the state of blockchain as soon as the block that is valid under the legacy rules and not valid under the new rules is introduced. The blockchain starts to diverge, dividing into two different blockchains, which are commonly called *hard forks*.

There are other changes to the protocol that do not break legacy consensus rules, but rather extend them, and do not force the network to partition. The result is a new backward-compatible protocol, commonly known as *a soft fork*. Despite its name, it only refers to the fork of only the consensus rules codified into Bitcoin clients, but it does not cause the blockchain to fork, i.e., to split.

The simplistic approach of increasing the block size limit by directly changing its value causes a hard fork to happen because it introduces breaking changes irreconcilable with legacy consensus rules. Ultimately, unless there is a 100% agreement, which is deemed a highly improbable event in practice, such change inevitably leads to the aforementioned network split. While some viewed this measure as worth the cost, others proposed indirect ways to introduce the change, through the soft fork. One such proposal is known as SegWit.

3 Segregated Witness

SegWit, or "Segregated Witness (Consensus layer)," as it is officially known, is a soft fork introducing certain changes to transaction validation rules in a way that does not break consensus with older Bitcoin software. For more information, see Bitcoin Improvement Proposals BIP141 [3], BIP143 [4], BIP144 [5], and BIP145 [6].

SegWit activation does not require a 100% consensus from the network *nodes*; it only requires consensus and further cooperation from the *majority of active miners* by hash rate (>50% would be sufficient for activation; the actual proposal chooses 95% as the activation threshold to reduce the amount of block orphanage events). SegWit has been successfully activated on the Bitcoin network on July 21, 2017. [7]

For older clients, a SegWit transaction output looks like one that anybody on the network can spend[1]. And yet, SegWit ensures only the intended recipient[2]can spend the output.

How does it work and what stops a malevolent party from spending someone else's SegWit output? After all, this would be considered a valid transaction for all legacy clients. However, SegWit is activated when the majority of miners agreed to do so, thus we expect miners to not include such a transaction because it is not valid under the SegWit rules. After SegWit is activated, along with a normal block having a legacy format, additional data begins circulating in the network, called *witness data*. Anyone spending a SegWit output must prove through the witness data that they are actually the intended recipient. This is checked against the public key hidden in the transaction output script (*redeemScript*), the part that is ignored by legacy clients but enforced by SegWit clients. Since SegWit requires a majority of miners to cooperate, they will only include the valid transactions in their blocks. Moreover, they will only build on top of blocks that follow the same strict rules, and miners that have not adopted SegWit will get their blocks that contain transactions breaking SegWit rules orphaned.

What stops the miners themselves from colluding to steal SegWit outputs? As discussed, if the colluding miners do not constitute the majority of miners by hash rate (>50% hash rate), their blocks will soon get orphaned, and the miners will not be rewarded for the energy spent mining them; thus, they are disincentivized from attacking SegWit in such a way. And if miners with more than 50% hash power were to collude with each other, or more than 50% hash power became concentrated in one's hands, Bitcoin decentralization would be under significant threat, and we would have to confront a much larger problem than the theft of SegWit outputs[3]. Thus, for the purposes of assessing the security of SegWit and the similar solution proposed below, we make the assumption that honest miners control more than 50% of network hash rate, and leave the subject of mitigating "51% attack" out of scope for this paper.

[1]Technically, in order to spend someone else's SegWit transaction *under legacy rules* one would have to provide the exact *redeemScript* that was hashed into the P2SH locking script, but security must not hinge upon this fact: at the moment the spending transaction is broadcast, *redeemScript* becomes publicly known, and if the transaction does not get into the blockchain quickly enough, someone could intercept it, use the revealed *redeemScript* for transaction input, and change the outputs to redirect the funds to themselves. For simplicity, we simply say "anybody can spend this output" in the main text.

[2]Once again, for simplicity, we omit the fact that a SegWit output can be intended to be unlocked by several recipients in a multisig, or by an arbitrary P2SH script as well.

[3]Contrary to the popular opinion, the danger of 51% attack lies not in the possibility of performing double-spends, but in the immediate expansion of 51% to 100% by ignoring other miners, disincentivizing them from participation, and taking significant control over the blockchain, and any subsequent community hard forks basing its mining algorithm on SHA-256.

4 SegWit Effect On The Block Size

SegWit has brought many benefits, including enabling Layer 2 solutions, fixing transaction malleability, improvements in multisig security and script versioning. However, what attracted the most spotlight is the increase in block capacity: transaction signatures that had previously been taking a significant part of block space can now be broadcast separately as witness data, which is not counted towards the 1 MB block size limit.

There is a question, however, of how much this increase amounts to. In theory, a contrived, specially crafted transaction could push the total block size (if we count the witness data as well) to 4 MB; however, in practice, real transactions are expected to be very different from those. A Bitcoin Core contributor Jimmy Song estimates[8] that the effective maximum block size increase with regular transactions could amount to almost 2 MB, i.e., twice the block size limit.

What puts the boundary on the upper limit is the fact that a SegWit transaction must partially exist in two worlds at once: its primary part on the blockchain, and its signature part in the "parallel world" that is witness data. This requirement of always having transactions occupy some part of the legacy block space is what restricts the maximum increase achievable by SegWit. In Proposal section, we discuss how to use the same mechanism to completely "snatch" the transaction output into such "parallel universe," after which the consecutive transactions do not have to occupy legacy block space at all, thus allowing for a theoretical increase of the total block size limit to any acceptable value, while doing so in a backward-compatible way.

5 Proposal

First and foremost, the author does not deny the validity and significance of proposed Layer 2 solutions in addressing the scalability problem (and in fact, working on implementing one for Ethereum blockchain). However, until these solutions are mature, well-tested, and adopted by many (e.g., Lightning Network hubs reaching sufficient liquidity and connectivity to prove usable), the search for ways to use the existing infrastructure to increase the block size limit in the meanwhile continues. SegWit constitutes one such approach, and as discussed, allows for almost a twofold increase. Building on the SegWit idea, author describes a proposal allowing for an even greater expansion of blocks, if considered necessary.

5.1 Superspace Blocks

Although SegWit blocks are, in a way, containers for normal blocks, we could view them as two distinct "worlds" in which transaction data resides: blocks, and witness data. Likewise, in Superspace, under the new rules, there are two worlds: normal blocks, and *superspace blocks*. Superspace block data propagates the same way as witness data in SegWit, and it is being considered only by the upgraded nodes.

A superspace block is a container for transactions, organized in a Merkle tree. Tables 1 and 2 define the structure of a superspace block and its header respectively.

Field	**Size, Data Type**	**Description**
Superspace block size	4 bytes unsigned int	Size of the superspace block, in bytes
Superspace block header	65 bytes, see Table 2	Fields from the block header
Transaction count	1-9 bytes VarInt	How many transactions will follow
Transactions	Variable	Transactions recorded in the superspace block

Table 1: Superspace block format.

Field	**Size, Data Type**	**Description**
Version	1 byte unsigned int	A version number of this protocol, initially `0x00`. Upgraded nodes must not reject Superspace blocks even if the version is higher than the one they run.
Previous normal block hash	32 bytes hash	Hash of the previous normal block, must match the `prevhash` in the paired normal block. It staples this superspace block to this exact chain and height in the normal space.
Superspace transactions Merkle root hash	32 bytes hash	Hash of the root of the Merkle tree containing transactions in this superspace block

Table 2: Superspace block header format.

5.2 Marker Transaction

To signal the existence of a superspace block, and also for Proof of Work to be able to secure data in both worlds, the two blocks must be paired together via a *marker transaction*. A marker transaction resides in the normal block and refers to its respective superspace block. Its amount can be of arbitrary length, its input (or inputs) can be any input(s) that the miner owns, but one of them must be signed by the same private key, respective public key of which is referenced by the coinbase transaction. This will provide proof to the rest of the network that the marker transaction came from the miner. It must also have exactly two outputs: one utilizing `OP_RETURN` followed by a `0x00` + 32 bytes SHA-256 hash of the paired superspace block header, and the second one can reference a change address, returning funds from inputs back to the miner. The inclusion of marker transaction does not cost the miner anything since the same miner takes the fee for it.

5.3 Block Assembly

For mining, this is the recommended order in which the block should be assembled:

1. Decide which valid transactions will be included in the superspace block
2. Calculate the Merkle tree root hash of superspace block transactions
3. Assemble the header of the superspace block
4. Generate the marker transaction
5. Decide which transactions will be included in the normal block (marker transaction included)
6. Calculate the Merkle tree root hash of normal block transactions
7. Assemble the header of the normal block
8. Validate both blocks to find any conflicts (see Validation subsection)
9. Start (or resume) mining by iterating on the nonce field in the normal block header and hashing the header data aiming to make the block valid for current difficulty

5.4 Address Format

A superspace address is generated as follows:

- Make sure the public key is in its compressed form (33 bytes in size) and starts with `0x02` or `0x03`.
- Calculate `hash160`[4] of the public key, let's call it *keyhash*.
- Assemble the P2SH *redeemScript*. Its format must be chosen as to not trigger SegWit execution, and be a valid *redeemScript* that evaluates to `true`. Author proposes: "`0x51 0x4c 0x22 0x00 <33-byte public key>`".
  - Legacy nodes that support P2SH but do not support Superspace will interpret such *redeemScript* this way: `0x51` is a `OP_1`, which pushes value "1" onto the stack. The next opcode is `0x4c`, which is an `OP_PUSHDATA1`, followed by the `0x22` which is the size of data to be pushed (34 bytes). It is followed by the 34 bytes which are pushed onto the stack. The script ends with leftover data on the stack. Since the script has successfully reached the end without termination, it evaluates to `true`.
  - Upgraded nodes running Superspace code would reinterpret any script that starts with the sequence "`0x51 0x4c 0x22`" followed by 34 bytes this way: ignore `0x51`, `0x4c` and `0x22`, treat the next byte as Superspace version (`0x00` initially, behavior for any other value is currently undefined and in this case the node must evaluate the *redeemScript* as a normal P2SH *redeemScript* until the behavior for this version is defined); treat the next 33 bytes as a public key for which the output is intended.
  - Note that `OP_1` (`0x51`) is different from `0x01` opcode; this format will not be interpreted as a SegWit witness program, since the first byte for SegWit *redeemScript* must be in a `0x00-0x0f` range[9].

[4] `hash160(x) = ripemd160(sha256(x))`

- Same as any other P2SH, the *scriptPubKey* is "`OP_HASH160` hash160(redeemScript) `OP_EQUAL`", and the resulting address is the corresponding P2SH address with prefix 3.
- For legacy wallets, this P2SH address must be provided. For upgraded wallets, a *superspace address* can be provided instead. It is generated using *bech32* address format [10], except:
  - The data type (initial letters before the first "1") must be:
    - "`bz`" instead of "`bc`" for mainnet
    - "`tz`" instead of "`tc`" for testnet
  - Instead of the *witness version*, use *Superspace version*;
  - Instead of the *witness program*, use the 33-byte public key.

5.5 Moving Bitcoin

5.5.1 From Normal Blocks To Superspace

When one intends to send bitcoins into superspace ("parallel universe"), they must set *scriptPubKey* according to the definitions in the "Address Format" Section. For upgraded wallets, a *superspace address* can be provided instead, and then the wallet software would infer from it the correspondent *scriptPubKey*.

When the transaction with this output gets included in the block, funds from this output are unlocked for use in superspace blocks, starting from the next block.

5.5.2 Within Superspace

Now that bitcoins went to superspace, one can spend them inside superspace. A radical difference of Superspace proposal from SegWit is in the fact that superspace transactions do not have to be represented in the normal block; thus they do not take any space in normal blocks at all and are not affected by the legacy block size limit in any way.

Superspace transaction format is mostly the same as for a normal transaction [11], and they are subject to the same validation rules. However, there are a few differences:

- In order to make the transaction format different and valid only in superspace, the 4 bytes version number must be `0xffffffff`, indicating it is a superspace transaction;
- A superspace transaction must reference as its inputs either:
  - another superspace transaction, in which case, its hash is referenced;
  - or superspace output from an initial transaction in a normal block, in which case, its *scriptSig* must consist of a transaction signature[5], public key of which must match the 33-byte compressed public key in the initial transaction, followed by a push of the serialized *redeemScript*.
- Although superspace transactions support P2SH, Superspace format is not recursive; transactions containing *redeemScript* that match SegWit and Superspace formats are treated like normal P2SH scripts, they must not trigger any additional execution/validation other than those defined in the P2SH *redeemScript*.

5.5.3 Back To Earth

Finally, bitcoins can be sent back from superspace to a normal block. Two scenarios are possible:

1. The output from the initial transaction has not been spent in superspace. In this case, the one for whom the output was intended can spend this output. She must already have a corresponding *redeemScript*, and her *scriptSig* must consist of a transaction signature[6], public key of which must match the 33-byte compressed public key in the initial transaction, followed by a push of this public key, followed by a push of the serialized *redeemScript*.

[5] Transaction signature must sign the transaction data, but the transaction data must include the signature. This paradox is resolved in Bitcoin by temporarily replacing *scriptSig* by the *redeemScript* itself, in the case of P2SH.

[6] See the previous footnote.

2. Or, alternatively, the output of the initial transaction has already been spent in superspace. In this case, *redeemScript* can be tracked down by following the superspace transaction history and extracting it from the first superspace transaction which referenced this output. The *scriptSig* must consist of: a push (`0x20` ...) of 32-byte superspace transaction hash, a push (`0x01` ...) of 1-byte index of the superspace transaction output to be spent, a push (`0x21`) of the compressed 33-byte public key, matching the P2PKH superspace transaction output referenced in its *scriptSig*[7], the transaction signature, and finally, the serialized *redeemScript*. If the amount of the initial transaction output differs from the superspace transaction output, one of the outputs must return the rest of the funds to the same exact *redeemScript*. The amount of this output must be exactly the fees spent on this transaction + the difference between the initial transaction output and the superspace transaction output. From now on, superspace transactions will not be able to reference this output, but they will be able to reference this newly created output instead.

5.6 Validation

- If a normal block contains a valid *marker transaction*, which signifies the existence of the paired superspace block, upgraded nodes must not accept the normal block as being valid until they have the superspace block, just as no full node would accept a block as being valid without being provided transactions data in it to verify validity of transactions.
- Superspace transactions are subject to the same validation rules as in legacy Bitcoin.
- To avoid the construction of elaborate validation rules spanning both worlds, one must be allowed to spend a normal transaction in superspace only starting from the next block. Likewise, one must be allowed to spend a superspace output in the normal block only starting from the next block.

6 Conclusion

Successful deployment of soft forks, such as Segregated Witness and many others [12], proved that it is possible to introduce major changes to the Bitcoin protocol without unnecessary network partitioning.

Superspace takes SegWit idea and expands it to allow for arbitrary block size limits, which would be ultimately decided by the Bitcoin community. The effect is such that, unlike in SegWit, superspace transactions do not have to take any space in normal blocks at all. To illustrate, it would now be possible, when the network is extremely congested, to have a full 1 MB normal block, with a paired superspace block of 9 MB, where transactions do not reference anything in the legacy space. After the "rush hour", the network can return back to normal, probably with superspace blocks not needed anymore; all while operating without disruptions.

Protocol activation, if accepted, would occur with a soft fork, in the same way as SegWit—with a 95% threshold of versionbits signaling in support of the protocol for 2016 consecutive blocks.

Although we say the Superspace protocol does not allow recursion, we could easily use the same mechanisms to extend the protocol further, if needed in the future, and have more "parallel universes". Extensions could not only allow for further alterations to Superspace block size limit, but also for new opcodes, experimenting with different consensus algorithms, all while not interfering with network operation of legacy Bitcoin blockchain.

One shouldn't mistakenly equate the Superspace protocol to the idea of sharding. In sharding, miners are allowed to choose the shard they operate on, and ignore other shards. They divide their hash power to dedicate it to the shards they choose, and as a result it becomes easier to perform a 51% attack. In Superspace, however, miners have to keep track of both normal and superspace blocks, and a superspace block is treated like an extension of a normal block, by the proxy of marker transaction, which includes the hash of the superspace block. Furthermore, to take bitcoins from superspace and back, only one transaction is required, not two as presented in many sharding concepts.

Of course, as was said, constant increase in the block size is not feasible in the long term, as only the well-funded parties would be able to keep up with ever-growing blocks, which would lead to extreme centralization [2]. As such, Superspace protocol is only a short-term proposal, intended to provide a temporary ease from scalability issues until Layer 2 protocols replace it as a permanent solution.

[7] Although a superspace transaction output can be as complex as in the normal block space, when it comes to spending a superspace output in a normal block, it must be in the P2PKH form.

References

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

[2] Block Limit Size Controversy - Bitcoin Wikipedia. https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[3] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Segregated Witness (Consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.

[4] Johnson Lau and Pieter Wuille. Transaction Signature Verification for Version 0 Witness Program. https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki.

[5] Eric Lombrozo and Pieter Wuille. Segregated Witness (Peer Services). https://github.com/bitcoin/bips/blob/master/bip-0144.mediawiki.

[6] Luke Dashjr. getblocktemplate Updates for Segregated Witness. https://github.com/bitcoin/bips/blob/master/bip-0145.mediawiki.

[7] SegWit Activation - Wikipedia. https://en.wikipedia.org/wiki/SegWit#Activation.

[8] Jimmy Song. Understanding SegWit Block Size. https://medium.com/@jimmysong/understanding-segwit-block-size-fd901b87c9d4.

[9] SegWit Witness Program Format (as defined in BIP141). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki#witness-program.

[10] Peter Wuille and Greg Maxwell. Base32 address format for native v0-16 witness outputs. https://github.com/bitcoin/bips/blob/master/bip-0173.mediawiki.

[11] Bitcoin transaction format. https://en.bitcoin.it/wiki/Transaction#General_format_of_a_Bitcoin_transaction_.28inside_a_block.29.

[12] A complete history of Bitcoin's consensus forks. https://blog.bitmex.com/bitcoins-consensus-forks/.