BUIP151: Further Development of the Bobtail/Storm Protocol

Submitted: 31 August 2020
by George Bissias

Introduction

In November of last year, BU awarded roughly $34K for BUIP131: Bobtail Prototype Extending Storm on Bitcoin Unlimited. With the help of BU developer Griffith, we have developed a hybrid protocol that incorporates the desirable elements of both the Bobtail and Storm protocols. We believe that this new protocol also compensates for some deficiencies in those same protocols. The hybrid protocol name may change, but in this document, we will refer to it as Tailstorm. We have created a prototype full node implementation for Tailstorm, based on the BU codebase.

In this BUIP, I am requesting $34,580 in additional funds to push development, show larger-scale operation, and refine and communicate the theoretical foundations of Tailstorm. Specifically, I would like to develop zero-confirmation capabilities exposed for use in wallet software via BU's extension library (libbitcoincash), deploy wallet instances on the NextChain testnet alongside the existing Tailstorm full node software, and compile an academic paper that explores the security properties of this protocol and reports on the results of the deployment.

Motivation

Two of the most critical barriers to cryptocurrency adoption are transaction confirmation time and security. For the former, users require fast assurance of future transaction confirmation. And for the latter, they require reasonable assurance that a confirmed transaction will not be reversed.

The Tailstorm protocol does not decrease average transaction confirmation time, but using partial PoW, it does allow for users to calculate the probability that a given transaction will be included in the next block, the probability that a doublespend will be included in the next block, and the likelihood that some PoW has been diverted to "silently" mine a doublespend. This enables what is often called a zero-confirmation (or zero-conf) transaction. Such a feature has the potential to greatly improve network usability, particularly for lower value transactions that do not require multiple confirmations. However, despite the zero-conf capability of Tailstorm, there still remains work necessary to describe the probability of confirmation as well as implement a wallet capable of leveraging this feature.

The Tailstorm protocol also has the potential to dramatically improve transaction security because it minimizes variance in confirmation time just like Bobtail. However, Tailstorm introduces several significant changes to Bobtail (discussed below), which themselves have not yet been vetted from a security standpoint. Thus, in order to claim an overall improvement in security, Tailstorm requires the academic analysis we propose to conduct in this BUIP.

Beyond zero-conf capabilities, the protocol's minimization of block interval variance dramatically improves UX for certain types of transfers, notably any type of live exchange where two users are waiting for at least one confirmation. This type of exchange is important for many P2P use cases, which are a key focus of Bitcoin Cash in its goal as a worldwide currency for people. Today, with Bitcoin and Bitcoin Cash, although 10 minutes is the average block discovery time, approximately 5% of blocks arrive after 30 minutes or longer --- this is a painful risk of delay for what should have been a quick transfer. For Tailstorm, assuming a reasonable choice of parameters, 95% of blocks will arrive within 12.5 minutes. Note that these results also apply to the multiple confirmations needed for exchanges. Although, requiring multiple confirmations tends to "smooth" out block variation, Tailstorm will produce even more regular block inter-arrival times.

Results from BUIP131

The first task was to develop a hybrid protocol that combined elements from Bobtail and Storm, which we label Tailstorm in this document. New nomenclature is introduced. For example, the blocks containing partial PoW called proofs in Bobtail and delta blocks in Storm, are renamed subblocks in Tailstorm. Like Bobtail, Tailstorm continues to form strong blocks, which we call Bobtail blocks, from the PoW represented by the k subblocks with lowest hash value (for tunable k).

A major early challenge arose from the realization that Bobtail, as described in the latest academic paper, forms the canonical transaction set from only a single subblock. This means that zero-conf transactions will be delayed by half a block interval on average, which is unacceptable. Our solution is to instead use transactions from the union of subblocks. But because subblocks build upon transactions in earlier subblocks having the same Bobtail block parent, their dependencies form a directed acyclic graph (DAG). Access to the DAG allows for block reconstruction and validation (both sub and bobtail blocks) and it also provides assurance that subblocks contain only compatible transactions (no doublespends). For this reason, another major change is that Bobtail blocks now include the entire DAG of subblocks connecting the k subblocks having lowest hash value (where k is a tunable security parameter), as well as the transactions included in each.

Our full node Tailstorm prototype implementation is available on gitlab as a fork of the official BU codebase. Griffith and I continue to refine the code, however the basic protocol has been implemented: subblocks are mined, Bobtail blocks are assembled from those subblocks, and the blockchain is extended accordingly. We have developed several QA tests that run on the python RPC framework for verifying block production. Recently we completed a Graphene-based subblock relay, which allows for block compression using the Graphene protocol. The following tasks will be completed before the end of work on BUIP131: 1) implementation of compact Bobtail blocks (that send only subblock hashes), 2) persistence for Bobtail blocks and subblocks to the block database, and 3) deployment on a 3-node testnet.

Objectives

The objectives for this BUIP are three-fold: zero-confirmation transactions for a BU wallet implementation, a deployment of both wallet and full nodes to the NextChain testnet, and submission of the Tailstorm protocol description, deployment details, and security analysis to a peer-reviewed conference.

1. Zero-confirmation transactions. Because subblocks are incentivized with a fraction of the overall block reward, the Tailstorm protocol enables higher security for zero-confirmation transactions than what is provided by Storm alone. Tailstorm DAGs are required to contain subblocks with compatible transaction sets, which makes a transaction included in a large DAG more likely to be confirmed than one included only in a small DAG. In fact, it is possible to assign a probability of confirmation to each transaction based on the DAG it occupies. We will leverage this property in designing a zero-confirmation facility.
    1. Work out statistics of zero-confirmation transactions in Tailstorm.

    2. Expose DAG-work to wallets through BU's libbitcoincash.so extension library. This will require implementing the statistical tests developed in part and introducing hooks for accessing them.
    3. Implement the zero-confirmation feature in a wallet compatible with BU's libbitcoincash.so extension library.
2. NextChain deployment. Our goal for the Tailstorm protocol is to demonstrate its viability as a future technology for Bitcoin Cash. Therefore, a major output from this BUIP will be demonstration and confirmation of interoperability and some theoretical properties of Tailstorm by means of a testnet deployment.
    1. Deploy and synchronize both Tailstorm full node as well as zero-confirmation-enabled wallet nodes on NextChain.
    2. Implement and run doublespend experiments to determine efficacy of zero-confirmation protection.
3. Analysis and conference submission. Another important aspect of protocol design is engagement with the academic community for the purpose of vetting statistical and game theoretical concepts. The final major contribution to this BUIP will be preparation of a submission to a peer-reviewed conference.
    1. Compile and analyze results from experiments conducted in the completion of objective 2b.
    2. Conduct a thorough literature search and review.
    3. Perform simulation of key statistical processes.
    4. Seek to develop security analysis of the Tailstorm protocol using standard techniques.

Budget and Project Duration

The total project duration will be 36 weeks from December 20, 2020 until August 28, 2021 at a cost of $34,580 with effort based on part-time FTE. The timeline is as follows with the budget allocated proportionally.

1. Zero-confirmation transactions [18 weeks]
    1. Work out statistics [2 weeks]
    2. Expose DAG-work to wallets [2 week]
    3. Develop wallet implementation [14 weeks]
2. NextChain deployment [6 weeks]
    1. Deploy full and wallet nodes [2 weeks]
    2. Implement and run experiments [4 weeks]
3. Analysis and conference submission [12 weeks]
    1. Compile and analyze experimental results [3 weeks]
    2. Conduct literature review [1 week]
    3. Simulate statistical processes [4 weeks]
    4. Security analysis [4 weeks]