Reducing the Number of Transaction Messages in Bitcoin

Article · December 2021

CITATIONS
0

READS
68

3 authors:

Vojislav Misic
Ryerson University
407 PUBLICATIONS **3,733** CITATIONS
SEE PROFILE

Jelena Mišić
Ryerson University
452 PUBLICATIONS **4,585** CITATIONS
SEE PROFILE

Xiaolin Chang
Beijing Jiaotong University
148 PUBLICATIONS **975** CITATIONS
SEE PROFILE

Peer-to-Peer Networking and Applications manuscript No.
(will be inserted by the editor)

Reducing the Number of Transaction Messages in Bitcoin

Vojislav B. Mišić · Jelena Mišić · Xiaolin Chang

December 13, 2021

Abstract Data propagation in the Bitcoin network is inefficient due to its permissionless nature and the lack of multicast/broadcast features. In particular, the number of messages needed to propagate a single transaction is very high which is rather wasteful in terms of bandwidth utilization. In this work we propose two simple modifications of Bitcoin software that allows a vast reduction of the number of messages needed for propagating a transaction over the network. The first modification consists of deferring the transaction announcements until a certain predefined number of new transactions is collected. The second modification consists of a new message format that allows several transactions to be sent within a single message. We show that the number of messages can be substantially reduced in this manner, at the expense of slight increase in transaction propagation delay. However, the latency of block propagation and of transaction acceptance is virtually unaffected by the proposed changes. Moreover, the tradeoff between the reduction of traffic and transaction delay can be minimized by judicious choice of the threshold number of deferred transactions.

1 Introduction

Bitcoin [19] is perhaps the most popular cryptocurrency, and likely the most famous (and the first widely known) application of blockchain technology [7,25]. Bitcoin provides verification and storage of financial transactions using a replicated blockchain ledger. Single transactions are eventually packaged into blocks, and both transactions and blocks are disseminated in a peer-to-peer (P2P) network using a set of simple protocols designed to ensure consistency of the ledger. As the network is permissionless, any node can join or leave at will, and most, if not all, of Bitcoin message protocols utilize a high degree of redundancy to compensate for the inability to use multicast and/or broadcast features.

Among the downsides of this redundancy is the high number of inventory (INV) messages which announce newly acquired data items (blocks and transactions) to the network. The inefficiency is particularly pronounced for transactions which are short, only 100 to about 500Bytes, yet require an entire Bitcoin message which is furthermore packaged into a TCP message, only to be replicated many times at each node to propagate a single transaction.

This inefficiency has been noted in the past and some solutions have been proposed, but the improvements obtained thereby are not very efficient overall; in addition, they require substantial changes to the Bitcoin protocols [20,6].

In this paper we adopt a different approach which relies on deferring the new transaction announcement until a certain predefined number of new transactions has been received by a node. Optionally, a timeout can be imposed as well to limit the period that deferred transactions wait until they are announced. **The contributions of the paper may be summarized as follows:**

- **The deferred transaction relay allows for substantial reduction in the number of messages needed to propagate a single transaction through the network, a the expense of a slight increase in transaction propagation delay. However, the increase is not too high and, in fact, the tradeoff between the reduction of the number of messages and the increase in transaction**

This paper is a revised and extended version of the paper 'Making transaction propagation more efficient: Deferred transaction relay in Bitcoin' by V. B. Mišić, J. Mišić, and X. Chang, presented at *IEEE Globecom 2020* in Taipei, Taiwan, in December 2020.

V. B. Mišić
Ryerson University, Toronto, ON, Canada

J. Mišić
Ryerson University, Toronto, ON, Canada

X. Chang
Beijing Jiaotong University, Beijing, China

propagation delay can be adjusted by choosing the threshold for sending the announcement about new transactions.
- **A new message format is described that accommodates multiple transactions, similar to the BLOCKTXN message originally introduced as part of the compact block protocol [4].**
- **We show that those improvements lead to a reduction of the number of messages and the associated overhead with only a small change in Bitcoin message propagation protocols, without significantly affecting the performance of the Bitcoin cryptocurrency system itself.**

The paper is organized as follows. Section 2 presents the sources of inefficiency in Bitcoin message propagation. Section 3 introduces the proposed changes, while Section 4 discusses the changes that need to be made to the Bitcoin message propagation protocols. Section 5 presents the results of performance evaluation. Section 6 presents and discusses related work. Section 7 concludes the paper.

2 Standard transaction relay is not efficient

Let us assume that the network is comprised of N nodes, each of which has $c_i, i = 1 \ldots N$, connections to other nodes or peers. Mean number of connections per node is, then, $\overline{c} = \frac{1}{N}\sum_{i=1}^{N} c_i$.

We also assume that new transactions arrive to the network following a Poisson process with rate of λ_t, and that a newly arrived transaction can be injected into the network at any of its N nodes with equal probability. As the result, externally injected transactions arrive to a node according to a Poisson process but with arrival rate of $\lambda_{tn} = \frac{\lambda_t}{N}$. The total arrival rate of transactions to any node is still λ_t, as all transactions eventually reach every node regardless of the node where they were originally injected.

Now, when a new transaction has arrived, the recipient node verifies it by checking the participant signatures and the amounts and, if it is found to be valid, announces it to its peers by broadcasting an INV message. Note that Bitcoin documentation refers to the announcement as broadcast, but in practice it is a sequence of unicast messages, one to each peer, as multicast is hard to achieve due to the geographical distribution of nodes and its volatility [18].

The INV message contains the 32-byte hash of the transaction which is deemed sufficient to reduce the likelihood of collisions to a sufficiently low value.

If the peer has not seen the transaction with the given hash value before, it replies with a GETDATA message asking for the actual transaction which is then sent in a TX message. Pseudocode of the standard transaction relay protocol is given as Algorithm 1.

```
Data: incoming TX message with new transactions
Data: set of Peers
Data: list of deferred transactions
receive TX message from peer p_s;
if transaction known then
    drop transaction
else
    verify transaction;
    if transaction valid? then
        for p ∈ Peers do
            if p ≠ p_s then
                prepare INV message with new transaction;
                send the INV message to peer p
            end
        end
    else
        drop transaction
    end
end
```

Algorithm 1: Standard transaction relay.

The other peers follow the same procedure: once they receive and verify the transaction, they send it to their peers using the protocol shown above. The original sender may not be included in this, as the peer knows which node did the transaction come from and omit that node from the 'broadcast.'

The procedure is repeated until all the nodes in the network have learned about the transaction. Since the nodes are connected into a graph, rather than a tree, some nodes may get duplicate INV announcements which they will simply ignore.

Let us assume, without loss of generality, that the transaction is injected into the network at node n_1. Then, the total number of INV messages sent to announce a single transaction will be

$$\begin{aligned} m_{INV} &= c_1 + \sum_{i=2}^{N}(c_i - 1) \\ &= \sum_{i=1}^{N} c_i - \sum_{i=2}^{N} 1 \\ &= N\overline{c} - (N-1) \\ &\approx N(\overline{c} - 1) \end{aligned} \tag{1}$$

as each node sends it out to all of its peers except the one that it got the transaction from in the first place.

The number of GETDATA and TX messages is lower, as only the nodes that have not learned yet about the new transaction will ask for it and subsequently get it. In other words, a node responds with GETDATA only to the first INV it receives; all subsequent INV messages announcing that same

transaction will be ignored. Since each node needs to get the new transaction, the number of GETDATA messages 'used' for a transaction is equal to the number of nodes minus one:

$$m_{GETDATA} = N - 1 \tag{2}$$

and the resulting number of TX messages is the same:

$$m_{TX} = m_{GETDATA} = N - 1 \tag{3}$$

For simplicity, we disregard retransmissions which are taken care of by the TCP protocol.

The total number of messages needed for a single transaction to propagate through the network is, then,

$$\begin{aligned} m_t &= m_{INV} + m_{GETDATA} + m_{TX} \\ &= N\overline{c} - (N-1) + N - 1 + N - 1 \\ &= N(\overline{c}+1) - 1 \\ &\approx N(\overline{c}+1) \end{aligned} \tag{4}$$

which corresponds to message arrival rate of about

$$\omega_m \approx \lambda_t N(\overline{c}+1) \tag{5}$$

expressed in messages per second.

As can be seen, transaction relay protocol in Bitcoin is not very efficient. This may be attributed to the fact that Bitcoin network is permissionless and volatile, as any node can join or leave at any time. In order to be able to improve efficiency of the protocol, we need to look at the two main sources of inefficiency.

First source of inefficiency is the number of messages required to propagate a transaction to all nodes which is rather high, in particular the number of INV messages. In an ideal world, one each of INV, GETDATA, and TX messages should suffice for a single transaction per node; however, the number of INV messages is higher by a factor of $\overline{c}-1$ which is at least 6, according to measurements [1,21].

Therefore, most of the INV messages exchanged in this process are, strictly speaking, unnecessary, but a node cannot know which among its peers already knows about the transaction so it sends it to all of them, minus the original source, of course.

Furthermore, all of these messages are short which does not improve bandwidth utilization [10].

Second source of inefficiency is due to the fact that each transaction, which has a length of about 100 to 500 Bytes, is sent in separate TX messages; again, this does not improve bandwidth utilization.

3 Improving the transaction relay protocol

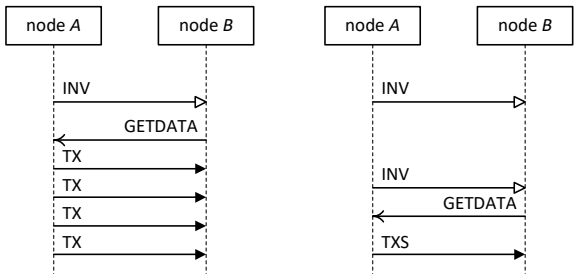
These observations motivated us to look for simpler solutions to both inefficiencies. In the following, we describe two such solutions.

3.1 Multiple transactions in a single message

The latter inefficiency may be reduced by defining a new message type which we will refer to as TXS. This does not require a big change of the Bitcoin protocol as **the TXS message is conceptually similar to the BLOCKTXN message that has been introduced as part of the compact block protocol** [4]. Similar to BLOCKTXN, the TXS message may contain a number of transactions in a serialized format, together with their count. Unlike BLOCKTXN, it does not need the header hash of a block as the transactions contained within are not yet part of any block. Instead, the TXS message could use a 32-byte all-zero number (which is not a valid hash for any block) as the block hash in the BlockTransactions structure [3], so we would just need **an additional message code** to implement it. **On account of these differences, the TXS message is not a replacement for the BLOCKTXN one, the more so because the ability and willingness to send and receive TXS messages needs to be announced separately, as outlined below.**

Defined in this manner, the TXS message would be a better match for the existing INV and GETDATA messages that may contain up to 50,000 data item inventories – i.e., item hashes and their types [2]. For the same number of transactions, the total number of messages needed for transaction propagation may thus be reduced through the use of TXS messages.

Transaction relay using standard, TX messages and the proposed TXS message is shown schematically in Fig. 1, assuming that INV and GETDATA messages contain five transaction inventories.

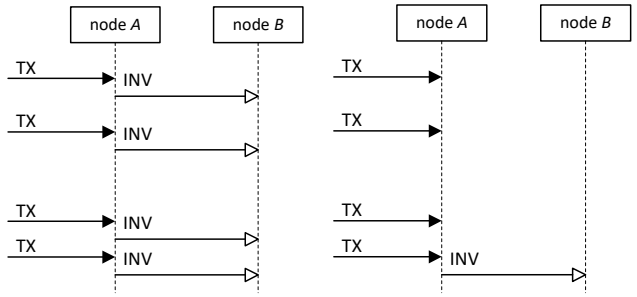


(a) Transaction relay using single-transaction TX messages.

(b) Transaction relay using multi-transaction TXS message.

Fig. 1 Transaction relay using single- and multi-transaction messages.

The TXS message would also be a good match for the MEMPOOL message. Namely, nodes in Bitcoin P2P network may leave and rejoin at will. When they rejoin the network, they need to bring their ledgers up to date by downloading the blocks it missed during its absence, which is achieved through the process known as Initial Block Down-



(a) Standard relay: INV announcement is sent as soon as the transaction is received and verified.

(b) Deferred relay: INV announcement is sent after a predefined number of transactions is received and verified, or after a predefined timeout.

Fig. 2 Standard and deferred transaction relay.

load [17,22]. (The same technique is used when a node joins the network for the first time.) However, downloading just blocks does not suffice if the node wants to take part in block mining, in which case it needs to download the missing transactions [18]. To this end, the node can ask its peers for the transactions they have recorded in their mempools using the MEMPOOL message. Once the peers respond with INV messages, the node can select the transactions it doesn't have and ask for them using GETDATA message. Then, the peers can use TXS message to send many transactions at once, rather than one by one.

While there is no conceptual limitation on the number of transactions that may be sent in a single TXS message, some limit arising from practical considerations should be imposed. We will discuss this in more detail in Sections 4.2 and 5 below.

3.2 Deferred transaction relay

Unfortunately, the mere availability of a multi-transaction TXS message does not address the former inefficiency, that of excessive number of INV messages. This number could be reduced if several new transactions were announced within a single INV message. This feature requires no change in the structure of Bitcoin messages, as an INV message is already allowed to contain up to 50,000 inventories, i.e., block and transaction hashes.

However, this facility is 'underutilized' as the default data propagation protocol requires (or, rather, assumes) that a node sends the INV announcement about a transaction to its peers *immediately* upon verifying a newly received transaction, as shown in Fig. 2(a). Newly received, in this case, refers to both transactions injected from an external client to the node and transactions received from the peers since the last INV announcement.

To reduce the number of such announcements and, consequently, reduce the wasted bandwidth, we may allow the nodes to defer the sending of an INV announcement until several transactions are collected, and then announce all of them with a single INV message. This improvement, schematically shown in Fig. 2(b), will be hereafter referred to as deferred transaction relay. We note that this approach can also be used in other applications with intermittent node connectivity [23,24].

Implementation-wise, we have two options at our disposal. First, a node could wait until a predefined number of new transactions has arrived and has been accumulated, and then announce them to all of its peers. This number could even be dynamically adjusted in accordance with the volume of transaction traffic. However, transaction arrivals are random and it is possible (though not very likely) that the threshold size of the deferred transaction collection will not be reached for several seconds or even longer. This indicates that a machine learning-based approach would perform well in this scenario [11,12].

Second, we can impose a timeout so that the announcements can be sent when the threshold is reached, or when the timeout expires even if the predefined threshold number of deferred transactions is not reached.

Together, these two mechanisms allow for limiting and controlling transaction propagation latency. As they are not mutually exclusive, both can be deployed at the same time. Fig. 3 shows an example of timing of messages and the node transaction queue size when both limiting of the number of transactions and a timeout are utilized.

In this manner, all transactions are guaranteed to be sent to all nodes, albeit with slightly longer delay due to the additional waiting period. In Section 5 we will discuss the extent of this increase in delay.

We note that, with or without the timeout feature, there is some inefficiency in terms of the content of deferred INV message. Namely, some of the deferred transactions may be already known to some of the peers. The optimum solution would be for the node to keep track of which peers have announced which transactions and prepare individualized INV messages that would announce only those transactions that the given recipient peer has not learned of yet.

It is questionable whether the added complexity (which is most certainly manageable) is worth investing. Moreover, a peer might had already learned about the transaction (or several of them) but has not informed the current node yet due to deferred transaction relay. This means that it is impossible to eliminate all redundancy that stems from the fact that all peers get the same INV announcement. However this is not critical: a transaction inventory in an INV message contains only a 4-byte data item type and a 32-byte hash for each transaction, hence the amount of redundant information is probably small enough to be ignored.

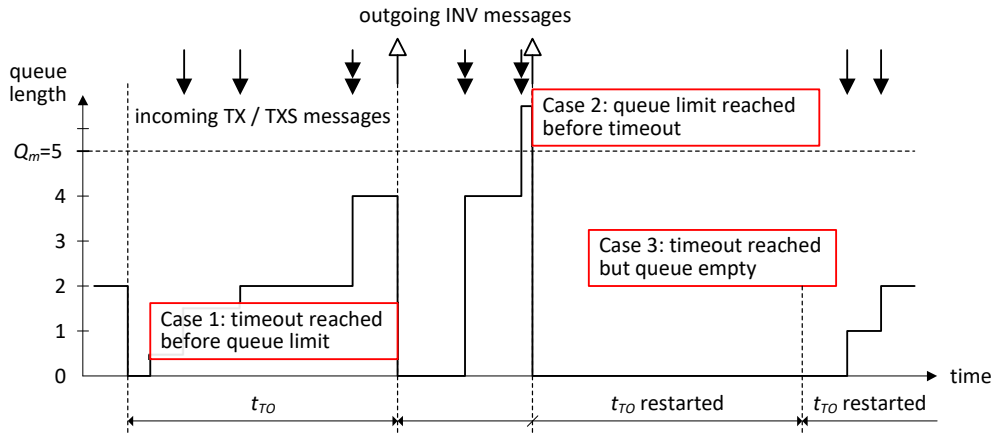


Fig. 3 Pertaining to the implementation of deferred transaction relay.

4 Implementing the changes

We will now present the actual protocols to implement the proposed changes.

4.1 Deferred transaction relay

In the deferred transaction relay, shown in pseudocode in Algorithm 2, we use two parameters: *threshold* which is the maximum number of transactions for which the INV announcement is deferred and *timeout* which is the maximum time for verified transactions to spend in the *deferred* transactions list.

```
Data: incoming TX message with new transactions
Data: set of Peers
Data: list of deferred transactions
Data: threshold (maximum size of deferred list)
Data: timeout for sending the INV announcement
Loop;
while deferred.count() < threshold or timer.time() < timeout
 do
    receive TX message from a peer;
    drop all known transactions;
    check TX message, drop all known transactions;
    if some transactions left then
        verify all new transactions; add new transactions to
          deferred list;
    end
end
if deferred.count() > 0 then
    for p ∈ Peers do
        prepare a new INV message containing all deferred
          transactions;
        send the new INV message to peer p;
        clear deferred list
    end
    restart timer;
    go to Loop
end
```

Algorithm 2: Deferred transaction relay.

Newly arrived and verified transactions are temporarily stored in a list which we will refer to as the deferred list. When the size of the list exceeds a predefined threshold, an INV message with all the transactions from the list is sent to all the peers. In this case, the number of transactions in the INV message may actually exceed the predefined threshold when the multiple-transaction TXS messages are used.

In addition, each batch of transactions announced to the node's peers via INV messages resets (actually, restarts) a timer that counts down from the timeout value. When the timer expires, the list of deferred transactions is checked. If it is empty, no action is taken; otherwise, all of its transactions are sent to all the peers of the node using INV messages. In the latter case, the number of transactions may actually be lower than the predefined threshold. The timer is then restarted.

4.2 Multi-transaction message

Multi-transaction messages are prepared by inspecting the list of requested transactions in the GETDATA message and filling the TXS message up to the predefined length limit with actual transactions taken from the node mempool. **A size limitation of 1MByte might seem appealing, as this happens to be the current limitation on block size. However, deferring transaction propagation until a TXS with close to 1MByte-worth of transactions can be sent would likely lead to unacceptable transmission propagation delays. As we will see in Section 5 below, smaller transaction counts are much more desirable from the performance standpoint.**

If the limit is set to a smaller value – perhaps derived from the 1500Bytes which is commonly used as the MTU (Maximum Transmission Unit) in Ethernet networks – one or more TXS messages may be prepared and sent in response to a single GETDATA message.

The node may even use both values and use the 1500Bytes size to respond to a GETDATA message in regular or deferred transmission relay, and the full 1MByte size to respond to the MEMPOOL message during an initial block download.

If the peer that sent the GETDATA message does not accept TXS messages, transactions will be sent one by one using the standard TX message format.

4.3 Compatibility with existing Bitcoin software

The proposed deferred transaction relay is fully compatible with existing Bitcoin software without any substantial change. Namely, INV messages already allow multiple transaction inventories, up to 50,000 of them. Thus nodes that run existing software can receive such messages without a problem. On the other hand, INV messages with a single transaction inventory can be received by nodes that run modified software. So from the standpoint of compatibility, this change requires no change and no adjustment.

TXS messages require a slightly different approach. In this case, backward compatibility with existing versions of Bitcoin software requires the introduction of another message, SENDTXS. This message would be sent by each node that can send and receive multiple-transaction TXS messages to its peers, informing them about this capability. Similar parameterless messages, namely SENDHEADERS and SENDCMPCT, are already used in different versions of the software to announce the ability to use HEADERS messages and compact block protocol, respectively. **While the formats of BLOCKTXN and TXS messages are indeed similar, the difference in required capabilities means that there is a need for two separate announcement messages and two separate data-carrying messages, SENDCMPCT and SENDTXS, and BLOCKTXN and TXS, respectively.**

Upon reception of SENDTXS message, nodes capable of processing TXS messages will label the sending peer as TXS-capable. Note that they would also inform their peers in the same manner, i.e., with SENDTXS messages. Nodes that run older software and cannot deal with TXS messages would simply ignore this message.

When a node needs to request data from a TXS-capable peer, a /TXS option could be added to GETDATA message, similar to the existing /CMPCTBLOCK option, so that a node willing to receive TXS messages may indicate so in the immediately preceding GETDATA message. Nodes that cannot send a TXS message would simply ignore the option and send the requested transactions in standard TX messages, one transaction each.

In this manner, nodes capable of using TXS messages can use them to send transactions to other such nodes, while nodes that run older software can continue to exchange single-transaction TX messages with all other nodes, be they TXS-capable or not.

Overall, introduction of SENDTXS and TXS messages does require minimum additional complexity. First, generating and processing new TXS messages is very similar to that of the existing TX and BLOCKTXN ones. Second, SENDTXS requires that labels be attached to a subset of node's peers and subsequently processing messages to and from nodes from that subset in a way that is slightly different from the standard one; similar provisions already exist in recent versions of Bitcoin software. Last, but not least importantly, processing of existing messages does not need any changes at all.

4.4 The impact on security

Transaction security is not affected by the use of TXS messages. **namely, requesting several transactions at once is already allowed in Bitcoin; whether transactions are then sent one by one using TX messages, as the standard stipulates, or in batches using TXS messages, does not impact security. In particular, no extra encryption or signing steps are necessary.**

Furthermore, as the TXS message does not prescribe any transaction format, transactions can be sgned using either the standard format, or the newer Segregated Witness (SegWit)[1] format [13] which aims to prevent transaction malleability and thus alleviate some of the attacks that use it.

It is worth noting that most attacks launched against Bitcoin were actually hacking attacks on miners or wallets, rather than attacks on the protocol itself[2]. Therefore, the slight increase in transaction propagation delay will not affect transaction-related attacks such as double-spending [19], as all transactions involved in such an attack will be delayed in the same manner. The increase in transaction propagation delay does not mitigate, but also does not facilitate, network attacks such as the eclipse attack [9] as they generally depend on the set of peers connected to the node in question.

5 Performance evaluation

To evaluate the performance of the proposed changes, namely the deferred transaction relay and the multi-transaction TXS

[1] Segregated Witness, Bitcoin developer documentation, `https://en.bitcoin.it/wiki/Segregated_Witness`, last accessed: December 13, 2021.

[2] History of Bitcoin page at `https://en.wikipedia.org/wiki/History_of_bitcoin`, last accessed December 13, 2021.

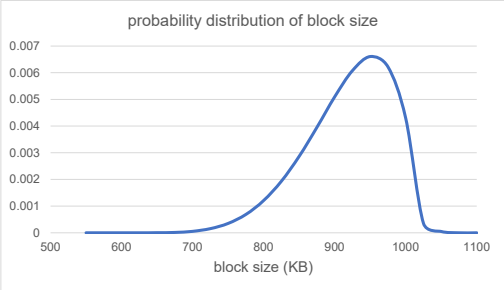


Fig. 4 Probability distribution of block size.

messages, we have built a simulator of Bitcoin network using AnyLogic multi-paradigm simulation software by The AnyLogic Company, Oakbrook Terrace, IL[3].

We note that it was not possible to run the modified protocol on a real Bitcoin network. Indeed we could have modified the software daemon and run it on a number of nodes all over the world using a suitable cloud provider such as Amazon AWS[4] or Microsoft Azure[5]. However, the total number of modified nodes would be too low to reach any meaningful conclusions about the performance of the modified protocol. In addition, the nodes would disrupt the operation of the network. Consequently, simulation was deemed to be the best choice to evaluate performance of the proposed scheme.

Our simulator has 1000 nodes interconnected in a Bitcoin-like P2P network. Maximum and minimum connectivity for a node was 9 and 52, respectively, resulting in mean connectivity of about 16.35 and network diameter of 4. Network delays are calculated according to the cumulative distribution function of delay obtained by measurements [1], taking into account the difference in intra- and inter-continental delays between Europe, Asia, and the Americas. We assume that the network throughput is 1GBps while individual nodes have the throughput of 10MBps.

Transactions are injected at a randomly chosen node using uniform distribution. Transaction arrival rate was set to 4.5 transactions per second for the entire network, which is close to the real Bitcoin network. Transaction size range from 100 to 511 bytes with uniform distribution.

Blocks were mined at a rate of one per ten minutes using exponential distribution. Block size was selected randomly between 600 and 1025KBytes using a beta distribution shown in Fig. 4; mean block size was about 915KBytes which is close to the data reported by tracker sites[6].

Using the simulator set with parameter values outlined above, we have run a number of experiments using different protocol options.

[3] http://www.anylogic.com
[4] https://aws.amazon.com/.
[5] https://azure.microsoft.com/.
[6] https://www.blockchain.com/charts

In one set of experiment runs, the threshold for deferred transaction list was varied between 1 and 12, with the timeout feature disabled. In the case where the threshold is set to one, required transactions are immediately transmitted one by one so the results for the case when multi-transaction TXS messages are used are the same as those obtained without that feature.

In the second set of experiment runs, the threshold for deferred transaction list was varied in the same range as before but the timeout feature was enabled and the timeout was set to a fixed value of 5 seconds.

In the third and final set of experiment runs, the threshold for deferred transaction list was fixed at 5 but the timeout was varied between 0.5 and 6 seconds. For reference, we have added the values corresponding to the case where threshold was set to 1, copied from the first experiment; the resulting values were labeled *imm* for 'immediate.'

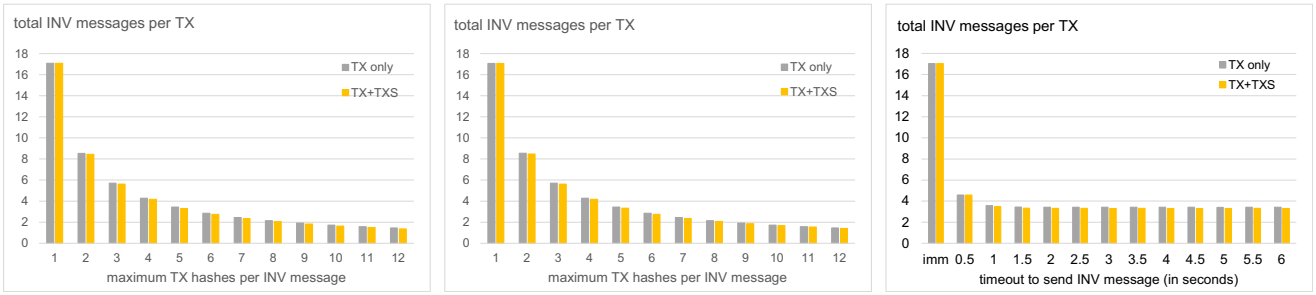
Within each experiment run, we have first run the simulator without multi-transaction TXS messages enabled and then switched the TXS on for the second part of the experiment run. **Each part of the experiment lasted for 15,600 seconds, or about 4 hours and 20 minutes. The total number of transactions generated during each portion was around 72,000, for a total of about 144,000 for each run. Total number of blocks installed in the blockchain was about 51 to 52 in each experiment run.**

The results are grouped around two main focal points. First, we present the manner in which the use of deferred transaction relay and TSX message format actually improves the low level performance, as measured by the reduction in the number of different messages exchanged. Second, we present the impact on the performance of Bitcoin application itself, as assessed through the transaction propagation times and transaction acceptance latency, namely the time from transaction injection in the network to the time that the transition is ultimately accepted as part of a regular Bitcoin block.

5.1 Reducing the number of messages

Mean number of TX messages sent per transaction is shown in Fig. 5. In this and all subsequent diagrams, (gray) columns on the left correspond to the case without TXS messages enabled while the (yellow) columns on the right were obtained with TXS messages switched on, unless explicitly stated otherwise.

Fig. 5(a) shows network behavior without the timeout feature. As can be seen, the initial number of INV messages when the threshold is set to one is about 17 per transaction, which reflects the mean node connectivity of 16.53. However, when the deferred transaction feature is enabled, the number of INV messages quickly drops: initially by half at

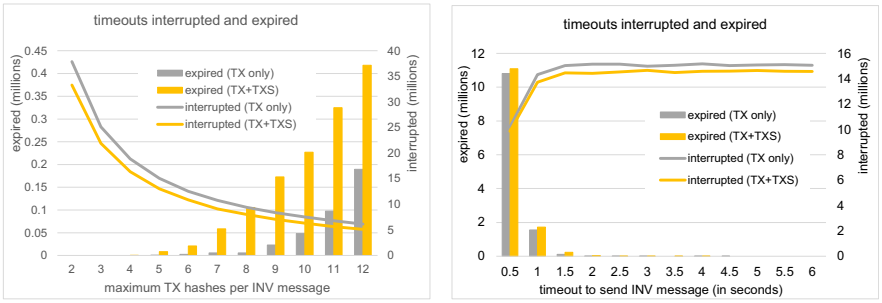


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 5 Mean number of INV messages per transaction.



(a) Variable deferred transaction limit, fixed 5 second timeout.

(b) Variable timeout, limit of 5 deferred transactions.

Fig. 6 Total timeouts interrupted and expired. Note the different vertical scale for two sets of results.

threshold value of 2, and then at a slower pace. At the threshold value of 12, it takes only about 1.44 INV messages per transaction which is a reduction of about 91.5%.

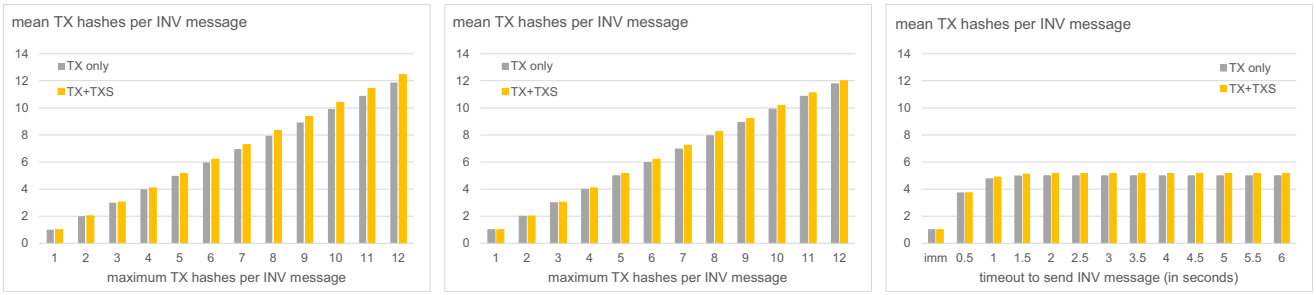
Fig. 5(b) shows network behavior under variable threshold but with the timeout enabled. The number of INV messages shows a rapid drop with the threshold, again reaching 1.44 per transaction at the threshold of 12. Interestingly enough, the impact of the timeout feature is not visible in this setup. This may be explained as follows: the mean transaction arrival rate is 4.5 per second, which is valid for the entire network but also for each node individually as transactions have to be propagated to each node. In the time period of 5 seconds (which corresponds to the timeout limit), the mean number of new transaction arrivals per node should be well over 20. As maximum threshold value is 12, threshold limiting will likely be activated before timeout limiting which will be reached in a small percentage of cases only. As the result, the diagrams in Figs. 5(a) and Fig. 5(b) are strikingly similar.

In both cases, the values obtained with the TXS feature enabled are below, but very close to, those obtained without the TXS multi-transaction messages enabled. This is due to the fact that a TXS message may bring several transactions to the node. Upon verification, they will be added to the node mempool but also to the deferred transaction list. As there may be more than one such transaction, there is nonzero probability that the actual number of transactions in the deferred list will exceed the threshold, albeit temporarily, which will help improve the efficiency of the deferred transaction relay technique.

Fig. 5(c) shows the behavior of the network under variable timeout but with the threshold fixed at 5. As can be expected, the values shown in this diagram with timeout of 5 seconds are virtually identical to those in Fig. 5(b) obtained with the threshold of 5. The number of INV messages initially shows a rapid drop, but levels off at timeouts of 2 to 2.5 seconds and above, at about 3.44 messages per transaction which is nearly 80% below the initial value.

As noted above, threshold limiting will be activated before the timeout limiting, as can be observed in the diagrams in Fig. 6 which show the total number of timeouts expired (columns) and interrupted (lines); note the different vertical scales for the two pairs of diagrams. (For obvious reasons, there is no diagram for the first experiment run, the one without timeouts.)

The number of expired timeouts increases with either threshold value, Fig. 6(a), or the timeout value, Fig. 6(b), while the corresponding numbers for expired timeouts decrease in both cases. However, the total number of expired timeouts is about two orders of magnitude higher than the

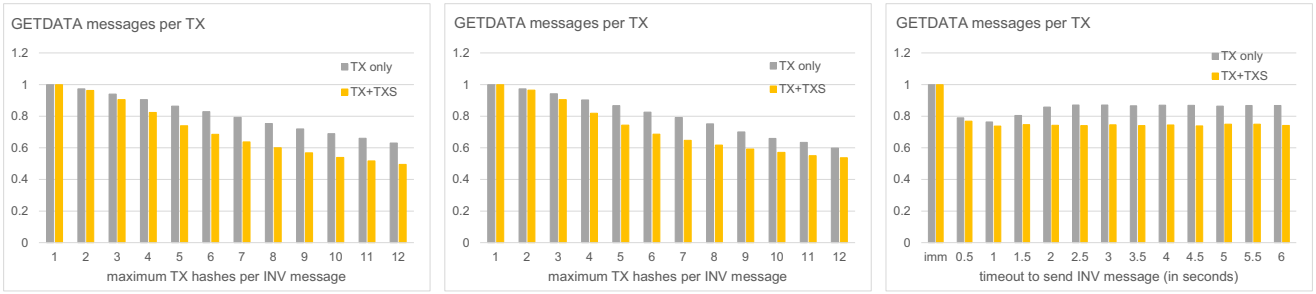


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 7 Mean number of transaction hashes per INV message.



(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

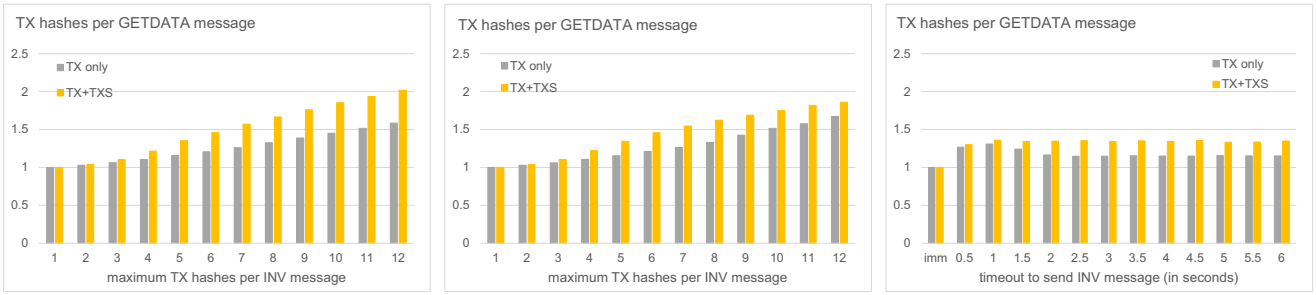
Fig. 8 Mean number of GETDATA messages per transaction.

number of interrupted ones in the experiment with fixed threshold, Fig. 6(a), hence the different vertical scales. In the experiment with fixed threshold, Fig. 6(b), there is a difference in the numbers of expired and interrupted timeouts, but the scales are approximately the same.

Diagrams in Fig. 7 show mean number of transaction hashes per INV message, again for three sets of experiments. Similar to the previous set, the diagrams obtained with variable deferred transaction list threshold show an almost linear rise with the threshold, from the default value of 1 which corresponds to the standard transaction relay, all the way up to nearly 12 for threshold value of 12. The diagrams with or without the timeout feature, Figs. 7(a) and 7(b), are nearly identical. On the other hand, the diagram obtained under variable timeout, Fig. 7(c), shows an increase up to the value of about 4.75 to 4.9 at timeout values of 2 to 2.5 seconds, and levels off at higher timeout values where it does not exceed 5. This mirrors the behavior of the mean number of INV messages per transaction in Fig. 5(c). Interestingly enough, the values obtained with TXS multi-transaction messages enabled are slightly higher than when this feature is disabled, which reflects the fact that a TXS message may bring in more transactions and, thus, make it easier for the deferred transaction list size to exceed the predefined threshold.

The number of GETDATA messages, is shown in Fig. 8, again separately for three experiment runs. As can be seen, the number of GETDATA messages sent per transaction starts at 1 for each run, but then tends to drop as the threshold increases, both with or without timeout, Figs. 8(a) and 8(b). The decline is not very steep, as the number goes down to about 0.6 messages per transaction. (We should keep in mind that GETDATA messages, just like INV messages, can contain several transaction hashes – in fact, up to 50,000 of them.) In both cases, the values obtained with the TXS multi-transaction feature enabled are noticeably lower, down to about 0.49 in case without the timeout and about 0.53 with the timeout. This is due to batch arrivals of transactions when TXS is enabled, which makes it easier for an INV message to contain more transaction hashes and, consequently, leads to more information per GETDATA message.

In case of fixed threshold but limited timeout, Fig. 8(c), the number of GETDATA messages per transaction drops to about 0.76 (0.74, in case TXS is enabled) at the timeout of 2 seconds. However, at higher timeout values the values obtained with standard transaction relay experience a slight increase to about 0.87, whereas those obtained with TXS feature enabled remain lower at about 0.74 or so. This is again

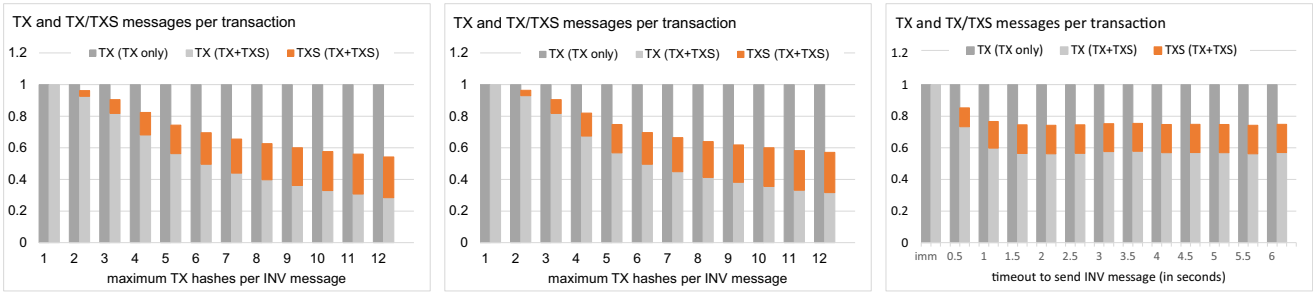


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 9 Mean number of transaction hashes per GETDATA message.

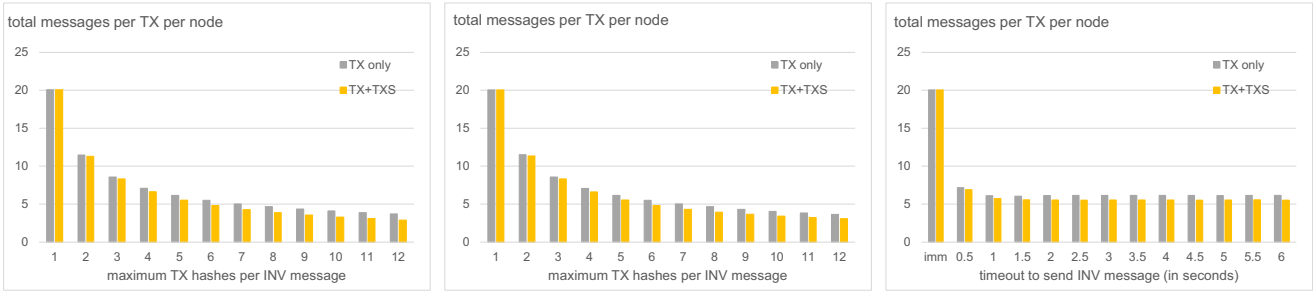


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 10 Mean number of TX and TXS messages per transaction.



(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 11 Total number of messages (INV, GETDATA, TX and TXS if applicable) per transaction.

due to the prevalence of threshold limitations over timeout limitations, as explained above.

Mean number of transaction hashes per GETDATA message, shown in diagrams in Fig. 9, tend to highlight these observations from another angle. For comparison purposes, all diagrams are shown at the same vertical scale. The higher information content of GETDATA messages in case the TXS multi-transaction messages are enabled can be seen clearly. Moreover, the experiment run with timeouts enabled leads to more GETDATA messages with fewer transaction hashes and, thus, to higher traffic in the network.

Total number of TX and TXS messages per transaction is shown in the diagrams in Fig. 10, with the left (gray) bar showing the number of TX messages when TXS messages are not used, and stacked light gray and orange bars on the right show the number of TX and TXS messages, respectively, when both are used. In the latter case, TX messages will still be used when there is only a single transaction to send, hence the need for two separate items.

Figs. 10(a) and 10(b) show that the number of TXS messages steadily increase with the increase in threshold values, while the number of TX messages decreases. As expected, the total number of messages when both TX and TXS messages are used is noticeably lower for the same number of transactions. Similar observations can be made for the diagram in Fig. 10(c) which shows the behavior of the network with variable timeout and limited deferred transaction threshold. The number of TX and TXS messages first decreases but then levels off, and their ratio remains at a fairly steady value, which is higher than the minimum obtained in the first two experiment runs.

Finally, we can compare the three approaches in terms of the total number of messages – INV, GETDATA, TX, and TXS ones – exchanged in the network during the process of transaction propagation. The corresponding diagrams are shown in Fig. 11. The diagrams in Figs. 11(a) and 11(b) demonstrate that the deferred transaction relay, regardless of the actual implementation details, is capable of significant reduction of the number of messages needed to propagate transactions through the network. Even allowing just two transactions in the deferred transaction list reduces the total number of messages by about 42.8%, and allowing three or four brings this down by 57.3% or 64.8%, respectively; further increases of the threshold value reduce this even further but the savings are getting progressively smaller.

As can be seen from Fig. 11(c), allowing even a small timeout of only 0.5 or 1 second reduces the number of messages by 52% or 64.3%, respectively. However, the savings tend to level off beyond about 2 or 2.5 seconds, remaining at about 69.3% below the initial value despite further increase of the timeout value.

5.2 Total number of messages and propagation delays

The use of TXS message format and the deferred transaction relay protocol do indeed lead to a reduction in the number of messages exchanged during the transaction propagation process. Question is, how does it reflect on the performance of the Bitcoin application itself?

The answer to this question can be found by observing the mean propagation delay, measured as mean time for a transaction to reach 99% of nodes in the network. The corresponding diagrams are shown in Fig. 12. As we may expect, the deferred transaction relay with limited threshold leads to an increase in propagation delay which is higher when timeouts are disabled, *cf.* Figs. 12(a) and Fig. 12(b). This is a consequence of the randomness of transaction arrivals: waiting for the threshold is interrupted by the timeout, if one is enabled, and the transactions for which the hashes have been received so far are being requested, regardless of how many there are. As the threshold limit is usually reached before the timeout limit, as Fig. 6 shows, many among the INV messages will carry only a couple of transactions so there will be more such messages, as explained above.

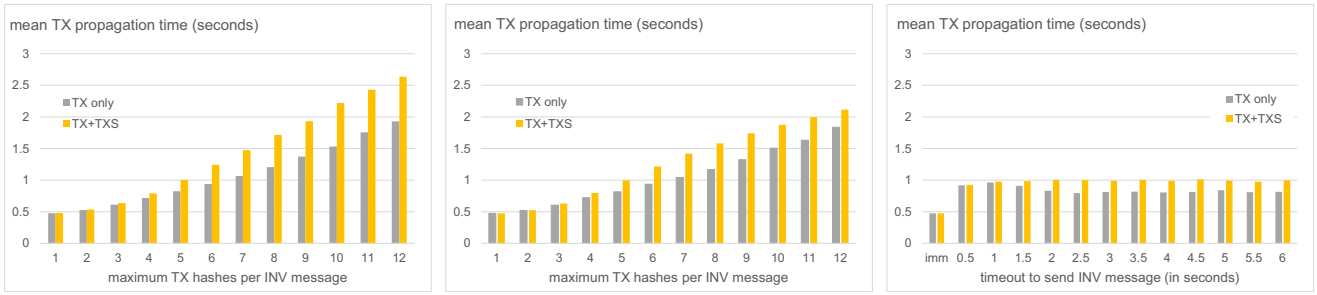
When timeouts are enabled, Fig. 12(b), delays decrease because more timeouts expire before the threshold is reached, so the transactions spend less time waiting to be sent out. Similar behavior may be observed when the threshold is fixed and the timeout is variable, as in Fig. 12(c). In this case, the threshold value is set to five which is rather low, so with longer timeout values, the number of interrupted timeouts quickly becomes very large, and transactions are sent out in batches of five or so, which is why the delay tends to stabilize beyond timeouts of 1.5 to 2 seconds.

In all three experiments, the delays are higher when multi-transaction (i.e., TXS) messages are allowed. **This is due to fact that the initiator node has to wait before sending a TXS message, which means that it will take longer for the recipient node to collect the threshold number of transactions or reach the timeout limit before sending the said message.**

This observation is confirmed by the 99% propagation delay, i.e., mean time for a transaction to reach 99% of the nodes in the network, the values of which are shown in Fig. 12. As can be seen, when there is no timeout, Fig. 12(a), the threshold is the single limiting factor, and each node will wait to collect the threshold value of transaction hashes before sending them out. On account of that, mean propagation delay increases almost linearly with the threshold.

Total transaction latency, from the time a transaction is injected into the network to the time that the block containing that transaction reaches 99% of the nodes, is shown in Fig. 13. As can be seen, there is no significant difference in the values of the latency with or without deferred transaction relay and/or multi-transaction TXS messages. Together with the transaction propagation delay results of Fig. 12, this indicates that the proposed protocol changes do not disrupt the operation of the Bitcoin network.

The corresponding propagation latency for blocks, i.e., mean time needed for a block to reach 99% of the nodes, is shown in Fig. 14. Similar to the 99% propagation delay for transactions, this latency is generally unaffected by the use of deferred transaction relay and multi-transaction TXS messages. This confirms that performance of the Bitcoin network is not impacted in a significant way by the proposed protocol changes. It also means that the probability of forks, which mostly depends on the block propagation times [15], will not change, as is the case with miner's fees [19] that depend on the miner's hashing power.

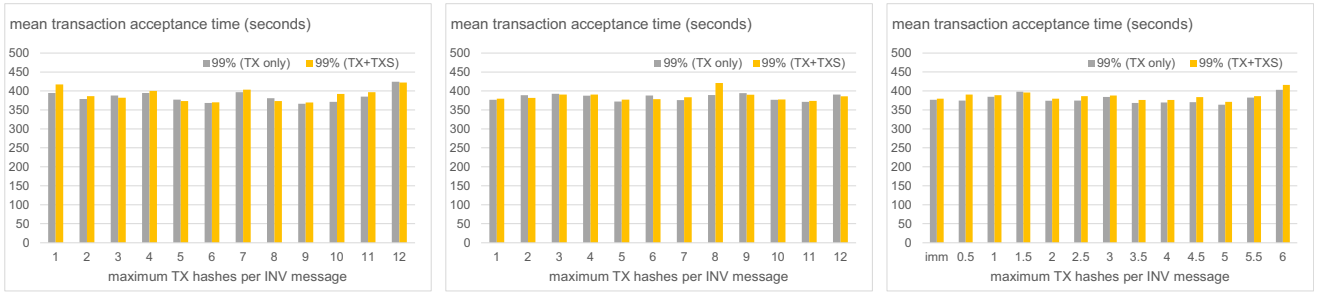


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 12 Mean time (i.e., propagation delay) for a transaction to reach 99% of nodes.

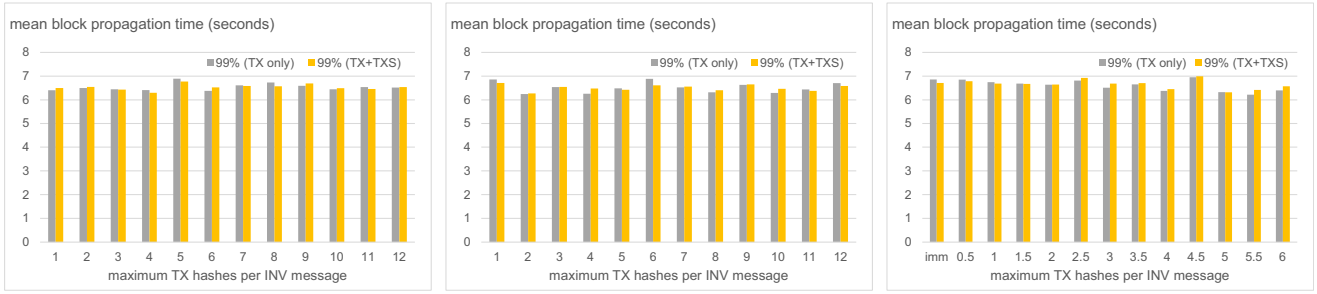


(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 13 Mean time for a transaction to be accepted: i.e., mean time for the block containing the transaction to reach 99% of nodes.



(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

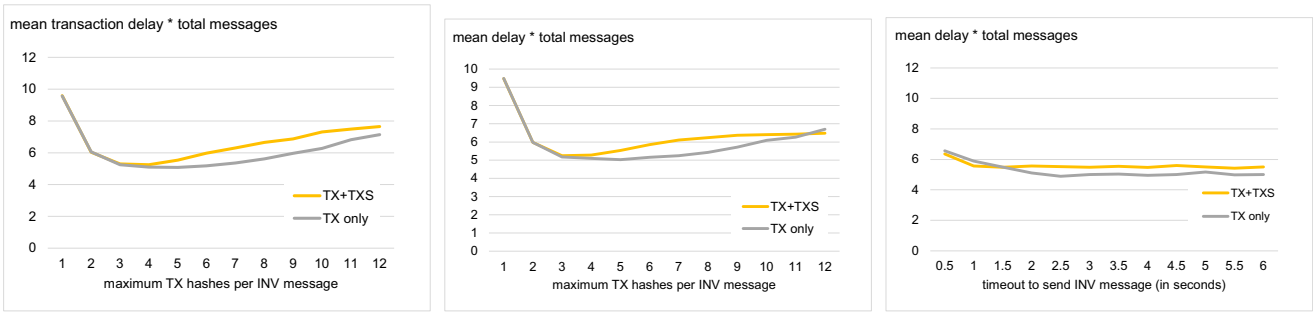
(c) Variable timeout, limit of 5 deferred transactions.

Fig. 14 Mean time (i.e., propagation delay) for a block to reach 99% of nodes.

5.3 Optimal values of queue threshold and timeout

Given that the total number of messages and the transaction propagation delay change in opposite directions, we may try to find an optimum combination of threshold and timeout values. The criterion for optimization could be the product of the total number of messages per node and the mean propagation delay in seconds, divided by the product of the number of nodes; the resulting values are shown in Fig. 15.

It may be noted that the curves for the first two experiment runs, Figs. 15(a) and 15(b), show a broad but visible minimum at threshold values from 3 to 7; the true minimum is at threshold value of 4 in both cases. Interestingly enough, the curves obtained when both TX and TXS messages are used are identical with, or slightly above, the curves obtained with TX messages only, which means that the use of TXS messages is useful when the threshold is set to 2 or 3, but does not bring much benefit at higher threshold values.



(a) Variable deferred transaction limit, no timeout.

(b) Variable deferred transaction limit, fixed 5 second timeout.

(c) Variable timeout, limit of 5 deferred transactions.

Fig. 15 Mean delay for a transaction multiplied by total number of messages per transaction per node.

In case of variable timeout, there are no such minima, at least not in the observed range of timeout values, but the curves, both for TX only and for TX and TXS, reach their minimum value at the far end of the timeout value range (i.e., around 6 seconds).

A conclusion that may be drawn from this is that setting the timeout to some small value of, say, 2 seconds, and simultaneously setting the threshold value to about 4 or 5, would lead to a substantial reduction in the number of messages whilst limiting the increase in propagation delay, thus allowing us to obtain the best of both worlds.

We also note that larger reductions in the number of messages were reported in [20] but at the expense of about 100% increase in propagation delays; and it needs a non-trivial modification of the Bitcoin software. On the contrary, the deferred transaction relay needs only a small change in the software and does not require any additional protocols; plus, it allows for fine tuning of the tradeoff between the number of messages and transaction propagation delay.

6 Related work

Most of the work concerning transaction relay has strived to improve its speed. Propagation of unverified transactions has been proposed in [5] as a way of speeding up data propagation, but transactions (and blocks) that are later found invalid are essentially a waste of bandwidth. In a similar vein, [26] proposes RepuLay, a relay protocol that uses dynamically calculated node reputation in order to determine whether to verify a transaction or not before sending it further on. The protocol claims to reduce the number of sent transactions by about half in the most favorable case, but the speed of transaction propagation in this case is not highlighted. Moreover, the analysis in [26] assumes fixed time slots which is unrealistic.

Reducing network traffic has been addressed by a much smaller number of papers. The Erlay protocol [20] does not broadcast a new transaction to all the peers of a node, using only a subset of the peers for each transaction (this is called low-fanout flooding). Then, the node will periodically engage its peers in mutual updating of transaction mempools using set reconciliation techniques [14]. However, there are problems with this approach. First, it increases the computational complexity due to the need to calculate initial and subsequent symmetrical set differences. **Second, reconciliation requires additional time as well as new message formats necessary to accomplish it. In particular, it uses a new 8-byte short transaction ID which is incompatible with existing Bitcoin implementations. Finally, there is a probability that a peer does not know of all the transactions that the other one does, and vice versa. In this case, the missing transactions must be exchanged in the usual manner (i.e., via the INV-GETDATA-TX handshake), in which case the benefits obtained through the improved protocol are lost.**

Similar to Erlay, the Shrec protocol [8] propagates transactions through limited flooding to a random subset of peers, which makes it resistant to some types of attacks. To shorten the announcement and data request packets, Shrec uses short transaction IDs with low probability of collision. However, the number of messages is not significantly reduced save for the use of limited flooding which effectively increases transaction propagation latency.

Another improvement has been developed as part of the Dandelion++ project [6] but its primary goal is anonymization of transaction source nodes, rather than a reduction of wasteful bandwidth usage. The protocol also requires non-trivial changes to the existing software but this is justified by the improvement in source privacy.

Our recent paper [16] proposed deferring of transaction dissemination with the explicit purpose of reducing the number of transaction announcements (INV) and data request (GETDATA) messages. However, it did not use multi-transaction messages which is described in the current paper.

Statements and Declarations

V. B. Mišić is an Associate Editor of the *Peer-to-Peer Networking and Applications*. He was not involved in the peer review process in any capacity.

The authors have no other competing interests, financial or otherwise.

The work of V. B. Mišić and J. Mišić was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through their respective Discovery Grants.

7 Conclusion

In this report we have described two additions to the Bitcoin protocols: the deferred transaction relay and the message to carry multiple raw transactions. Both changes are simple and compatible in concept with the existing protocols. Simulation results show that the deferred transaction protocol allows for controlled reduction of the number of messages needed to propagate a transaction, at the expense of slight increase in transaction propagation delay. By adjusting the threshold and timeout values, we can tune the tradeoff between the number of messages and propagation delay. This modification is fully backward compatible with the existing protocols and allows for interoperability of nodes that can and cannot use it. The multi-transaction message format is likely to be useful in cases where nodes need to update their mempools upon returning to the network; it is also backward compatible with the existing software.

References

1. Ben Mariem, S., Casas, P., Donnet, B.: Vivisecting blockchain P2P networks: Unveiling the Bitcoin IP network. In: ACM CoNEXT Student Workshop (2018)
2. P2P network guide. https://bitcoin.org/en/p2p-network-guide, last accessed: December 13, 2021
3. Full protocol specification. https://en.bitcoin.it/wiki/Protocol_specification, last accessed: December 13, 2021
4. Corallo, M.: BIP 152: compact block relay (2016). https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki, last accessed December 13, 2021
5. Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: Proc. 13th IEEE Int. Conf. Peer-to-Peer Computing (P2P'13), vol. 26 (2013)
6. Fanti, G., Venkatakrishnan, S.B., Bakshi, S., Denby, B., Bhargava, S., Miller, A., Viswanath, P.: Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. In: 2018 ACM International Conference on Measurement and Modeling of Computer Systems – SIGMETRICS '18 (2018)
7. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. In: Conf. Theory Appl. of Cryptography, pp. 437–455 (1990)
8. Han, Y., Li, C., Li, P., Wu, M., Zhou, D., Long, F.: Shrec: bandwidth-efficient transaction relay in high-throughput blockchain systems. In: Proceedings of the 11th ACM Symposium on Cloud Computing, pp. 238–252 (2020)
9. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on Bitcoin's peer-to-peer network. In: USENIX Security Symposium, pp. 129–144 (2015)
10. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach Featuring The Internet, 6th edn. Addison-Wesley Longman, Boston, MA (2016)
11. Liu, Y., Feng, T., Peng, M., Guan, J., Wang, Y.: Dream: Online control mechanisms for data aggregation error minimization in privacy-preserving crowdsensing. IEEE Transactions on Dependable and Secure Computing (2020)
12. Liu, Y., Wang, H., Peng, M., Guan, J., Wang, Y.: An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning. IEEE Internet of Things Journal **8**(10), 8616–8631 (2020)
13. Lombrozo, E., Johnson, L., Wuille, P.: BIP 152: Segregated witness (consensus layer) (2015). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki, last accessed December 13, 2021
14. Minsky, Y., Trachtenberg, A., Zippel, R.: Set reconciliation with almost optimal communication complexity. Technical Report TR2000-1813, Cornell University (2000)
15. Mišić, V.B., Mišić, J., Chang, X.: On forks and fork characteristics in a Bitcoin-like distribution network. In: 2nd IEEE Int. Conf. Blockchain (Blockchain-2019). Atlanta, GA (2019)
16. Mišić, V.B., Mišić, J., Chang, X.: Making transaction propagation more efficient: Deferred transaction relay in Bitcoin. In: IEEE Globecom 2020. Taipei, Taiwan (2020)
17. Motlagh, S.G., Mišić, J., Mišić, V.B.: An analytical model for churn process in Bitcoin network with ordinary and relay node. Peer-to-Peer Networking and Applications **13**, 1931–1942 (2020)
18. Motlagh, S.G., Mišić, J., Mišić, V.B.: Impact of node churn in the Bitcoin network. IEEE Transactions on Network Science and Engineering **7**(3), 2104–2113 (2020)
19. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
20. Naumenko, G., Maxwell, G., Wuille, P., Fedorova, A., Beschastnikh, I.: Erlay: Efficient transaction relay for bitcoin. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 817–831. London, UK (2019)
21. Neudecker, T., Andelfinger, P., Hartenstein, H.: A simulation model for analysis of attacks on the Bitcoin peer-to-peer network. In: IFIP/IEEE Int. Symp. Integrated Network Management (IM), pp. 1327–1332 (2015). DOI 10.1109/INM.2015.7140490
22. P2P Network (2019). https://www.bitcoin.org/en/p2p-network-guide#initial-block-download, last accessed: December 13, 2021
23. Quan, W., Cheng, N., Qin, M., Zhang, H., Chan, H.A., Shen, X.: Adaptive transmission control for software defined vehicular networks. IEEE Wireless Communications Letters **8**(3), 653–656 (2018)
24. Quan, W., Liu, Y., Zhang, H., Yu, S.: Enhancing crowd collaborations for software defined vehicular networks. IEEE Communications Magazine **55**(8), 80–86 (2017)
25. Wu, J., Guo, S., Huang, H., Liu, W., Xiang, Y.: Information and communications technologies for sustainable development goals: state-of-the-art, needs and perspectives. IEEE Communications Surveys & Tutorials **20**(3), 2389–2406 (2018)
26. Zhang, M., Cheng, Y., Deng, X., Wang, B., Xie, J., Yang, Y., Zhang, J.: Accelerating transactions relay in blockchain networks via reputation. In: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), pp. 1–10. IEEE (2021)