How many transactions per second can bitcoin really handle ? Theoretically.

Evangelos Georgiadis

1. Introduction

Transactions are arguably the most important part in the bitcoin mechanism, with everything else facilitating the proper creation, propagation and validation; culminating with their addition to the public ledger – the blockchain. One crucial measure inevitably intertwined with transactions is, *throughput*, the number of transactions confirmed (or added to the blockchain) per second[1], or simply, tps. Understanding throughput capacity from different angles remains of paramount importance for gaining insights into the underlying infrastructure.
We compute the *exact* upper bound for the *maximal* transaction throughput of the bitcoin protocol and obtain 27 tps. The previous best known bound for the *average* transaction throughput is 7 tps. *All results are based on legacy infrastructure, i.e., pre-SegWit era.*

1.1. Organization of paper.

Previous bounds are briefly highlighted, then a set of assumptions is discussed. Since this analysis relies on in-depth knowledge of protocol specifications, additional background and details are provided about block, block structure and transactions.[2]. Based on this information, actual transaction space is calculated. At last, culminating with the back of the envelope computation of the maximal throughput bound.

2. Previous bounds: Approximate Concrete, Exact and General

The bound 7 tps is cited in papers [12, 11], other papers [14, p.5] claim 10 tps. Amongst those values 7 appears to be the most often cited in the literature, tracing back to a back of the envelope calculation in [9]. Note that these bounds while *concrete* are *approximate*. For instance, the computation involved does not distinguish between actual transaction space within a block and blocksize, and thus is not truly fine-tuned to the protocol level. Additionally, the computation depends on an approximation for transaction size.

We also employ a *meticulous* back of the envelope calculation with an in-depth focus at the protocol level. This enables direct computation with minimal assumption base[3]. Our bound 27 tps, is both *concrete* and *exact*. (Surprisingly, this is the

Date: April 1st, 2019.☺. Midnight musings.

[1]Other papers might abbreviate this as *transactions/sec* or *tx/sec* or *tx/s*. We currently prefer *tps*.

[2]For a thoughtful introduction into this space, the avid reader should consult with source [1]

[3]Economic factors as well as other system parameters, such as, propagation delay, growth rate of main chain, etc ... are not considered. We are interested in engineering maximal transaction throughput based on the protocol's specification. Essentially assuming no latency issues, etc...

same value as the throughput limit *based on SegWit* in [12]. Reminder, we assume 1 MB blocksize limit.)

More sophisticated approaches leading to general bounds are found in [13] and [14]. Those are based on additional system parameters, such as latency and growth rate of main chain, etc. Thus definitions involve other functions, which, in turn, cannot be computed directly without additional assumptions. In [13], for instance, the transactions per second or TPS function is defined in terms of the growth rate of the main chain, which needs additional assumptions for further computation. That said, once assumptions have been postulated, bounds can be developed. See also [14, p.18] for an upper bound of the transaction throughput.

3. Assumptions for the various approaches

There are three fundamental assumptions, which are as follows. The size of a block is limited to 1MB[4], the consensus algorithm stipulates that a block is generated *on average* every 10 minutes, and the notion of an "average, or more robustly, median sized transaction."

Our viewpoint differs slightly yet significantly. We are interested in *engineering maximal throughput*, given the constraints of the 1 MB blocksize limit and the block generation rate. This task is accomplished via finding appropriate transaction types within the transaction universe. In particular, we engineer transactions that are minimalistic in complexity and byte size.

3.1. **Assumptions pertaining to the known 7 tps bound.** In all fairness, this result in [9] is based on transactions of the form: 2 inputs and 2 outputs. Further, it takes the economic aspect into consideration, a focus that deviates substantially from our analysis. Penultimately, the result is not fine-tuned at the protocol level – does not distinguish actual transaction space from blocksize limit, etc... Finally, the computation assumes approximate and obsolete "average or median sized transactions."

That said, a 1 input and 1 output instance was also considered (under similar assumptions) resulting in 9 tps. This is still multiples away from our 27 tps bound!

On a different note, the metric used to compute tps is in terms of KB; this might be more appropriate when considering latency,etc[5].

[4]We assume legacy infrastructure, pre-SegWit era. Additionally, 1MB is defined as 1,000,000 bytes.

[5]Once again not the avenue this analysis aims at.

4. Block structure

According to bitcoin's specifications [6], the block structure is as follows. The block consists of the blocksize, blockheader, transaction counter, followed by the list of transactions.

Blockstructure and Properties

Size (in bytes)	Field	Description
4	Blocksize	Size of block in bytes
80	Blockheader	Blockheader consists of multiple fields
1-9	Transactioncounter	Number of transactions that follow
Variable	Transactions	Actual transactions in this block

5. Actual Transaction space

The size of the block cannot exceed 1 MB. Thus, the size for the transaction space is actually, less than 1 MB; more precisely[6]

$$\begin{aligned}\textbf{Transaction Space} &= 1\text{MB(blocksize)} - 80\,\text{bytes(blockheader)} \\ &\quad - 4\,\text{bytes(blocksize in bytes)} - 3\,\text{bytes(transactioncounter)} \\ &= 999913\,\text{bytes}.\end{aligned}$$

The question now is how many transactions can fit into this space.

6. Transactions

Bitcoin's transaction mechanism is based on *Script*, a simple Forth-like, stacked based, left to right processing language [4]. There are different types of transactions in the transaction universe [8], varying in degree of complexity and size. Transactions, in algorithmic abstraction, are about inputs and outputs. For a thoughtful introduction into this sphere, we encourage the interested reader to consult with chapter 6 in [1].

6.1. **Transactions within a block.** There is some form of transactions hierarchy within every block. Every block must have a *coinbase transaction*, which in turn, must be the first transaction of the block. Coinbase transactions have some unique characteristics distinguishing them from the rest. For example, coinbase transactions have exactly 1 input with prescribed special form, and according to BIP 34 [3] are christened with *Height*, an additional 'property' that adds to size. There are a few other wrinkles that (hopefully) bear no relevance for this analysis.

[6]How did we obtain the value for *Transactionscounter* before determining the actual number of transactions that can fit inside the transaction space ? The answer is, this value was computed after we found the size of a 'minimal transaction'. Another good question, why 3 bytes and not 2 bytes ? Well, any number greater than 252 and less than 65536, by bitcoin specification (see [5]), takes 3 bytes. Note that one byte is pre-fixed to the number to indicate its length.

6.2. **General data structure of a transaction.** For sake of completeness we are reproducing relevant parts of the table from [7], detailing the general structure of a transaction.

Raw Transaction Format and Properties

Size (in bytes)	Field	Description
4	Version	Version number
Varies	Tx_in count	Number of inputs in this transaction
Varies	Tx_in	Transaction inputs
Varies	Tx_out count	Number of outputs in this transaction
Varies	Tx_out	Transaction outputs
4	Lock_time	Time or block number

6.2.1. *General structure of each input of a transaction.* For sake of completeness we are reproducing relevant parts of the table from [7], detailing the general structure of each input of a transaction, `Tx_in`.

General format of each input of a transaction and properties: Tx_in

Size (in bytes)	Field	Description
36	Previous_outpoint	Previous outpoint being spent
Varies	Script bytes	Number of bytes in scriptSig $\leq$ 10,000 bytes
Varies	ScriptSig	Script
4	Sequence	Sequence number

6.2.2. *General structure of each input of a transaction.* For sake of completeness we are reproducing relevant parts of the table from [7], detailing the general structure of each output of a transaction, `Tx_out`.

General format of each output of a transaction and properties: Tx_out

Size (in bytes)	Field	Description
8	Value	Number of satoshis to spend. Maybe 0.
≥ 1	Tx_script length	Number of bytes in the pubkey script. Maximum is 10,000 bytes
Varies	ScriptPubKey	Script

6.3. **General data structure of a coinbase transaction.** For sake of completeness we are reproducing relevant parts of the table from [7], detailing the general structure of a coinbase transaction.

Coinbase Input Format and Properties

Size (in bytes)	Field	Description
32	Hash(null)	A 32-byte null, as coinbase has no previous outpoint
4	Index	0xffffffff
Varies	Script bytes	No. of bytes in coinbase script $\leq$ 100 bytes
≥ 4	Height	Block height required since BIP34
Varies	Coinbase script	Coinbase field
4	Sequence	Sequence number

6.4. **Byte lightest transaction within the transaction universe.** Within the transaction universe, the skinniest and thus byte lightest transaction appears to come via *anyone-can-spend output*, a transaction which has no constraints to how its output can be spent. In other words, a transaction that can be made spendable by anyone. The transaction is defined as spending an `OP_TRUE` *anyone-can-spend output* and generating one `OP_TRUE` *anyone-can-spend output*. Before we analyze its size, a few pointers about properties and usecases.

6.4.1. *Properties.* The properties of *anyone-can-spend output* are outlined in [10].

scriptPubKey: (empty)
scriptSig: `OP_TRUE`

While this is a *valid* transaction, it is *non-standard*.

6.4.2. *Use cases.* At first glance, this appears to be an utterly useless, if not eerie, type of transaction. Even bitcoin and blockchain expert, Andreas Antonopoulos, in [2] believes it to have no practical use other than for donations. Though, on second thought, the following reasons might provide some incentive for reconsideration.

- Useful for establishing new theoretical bounds on throughput.
- A strategy for *coin laundering*; effectiveness depends on execution.
- A strategy for *network disruption*; substantially increases network traffic.

6.4.3. *Counting Size/Bytes:* Going through the list of values in the above tables. From top to bottom.

$$\begin{aligned}\text{Size of non-coinbase transaction} &= (\text{Version} + \texttt{Tx\_in count} + \texttt{Tx\_in} \\ &\quad + \texttt{Tx\_out count} + \texttt{Tx\_out} + \texttt{Lock\_time}) \\ &= 4 + 1 + (36 + 1 + 1 + 4) + 1 + (8 + 1 + 0) + 4 \\ &= 4 + 1 + 42 + 1 + 9 + 4 \\ &= 61\,\text{bytes}.\end{aligned}$$

$$\begin{aligned}\text{Size of coinbase transaction} &= (\text{Version} + \text{Hash} + \text{Index} + \texttt{Tx\_in count} + \texttt{Tx\_in} \\ &\quad + \texttt{Tx\_out count} + \texttt{Tx\_out} + \text{Height} + \text{Sequence} \\ &\quad + \texttt{Lock\_time}) \\ &= 4 + 32 + 4 + 1 + (1 + 1) + 1 + (8 + 1 + 0) + 4 + 4 + 4 \\ &= 65\,\text{bytes}.\end{aligned}$$

7. Maximal Transaction Throughput

Thus the maximal transaction throughput is computed as follows.

$$\lfloor \frac{(999913 - 65)}{61} \rfloor + 1 = 16391$$

transactions can fit inside a block.

Hence, bitcoin's maximal transaction throughput is: (converting minutes to seconds)

$$\lfloor 16391/(10 \times 60) \rfloor = 27\,\text{tps}.$$

Acknowledgement

We thank Giulia Fanti for comments on [14] and Yonatan Sompolinsky for comments on [13, 9].

References

[1] Andreas M. Antonopoulos. Mastering bitcoin: programming the open blockchain. 2nd Edition. O'Reilly Media Inc., 2017.
[2] Bitcoin multisig and p2sh transactions with Andreas Antonopoulos - 01/13/14. https://youtu.be/K-ccC9YZ8UI?t=2717, 2014. Accessed April,1st 2019.
[3] Gavin Andresen. 34. Block v2, Height in Coinbase. 2012-07-06. https://github.com/bitcoin/bips/blob/master/bip-0035.mediawiki
[4] Bitcoin wiki. Script. https://en.bitcoin.it/wiki/Script. April,1st 2019.
[5] Bitcoin Developer Reference. Unsigned Integers. https://bitcoin.org/en/developer-reference#compactsize-unsigned-integers.
Accessed April,1st 2019.
[6] Bitcoin wiki. Block. https://en.bitcoin.it/wiki/Block.
Accessed April,1st 2019.
[7] Bitcoin Developer Reference. Unsigned Integers. 2019. https://bitcoin.org/en/developer-reference#raw-transaction-format.
Accessed April,1st 2019.
[8] Bitcoin wiki. Transaction. https://en.bitcoin.it/wiki/Transaction.
Accessed April,1st 2019.
[9] Bitcoin wiki. Scalability. https://en.bitcoin.it/wiki/Scalability.
Accessed April,1st 2019.
[10] Bitcoin wiki. Script. Anyone-Can-Spend Outputs. https://en.bitcoin.it/wiki/Script#Anyone-Can-Spend_Outputs. Accessed April,1st 2019.
[11] Chenxing Li, Peilun Li, Wei Xu, Fan Long, and Andrew Chi-Chih Yao. Scaling nakamoto consensus to thousands of transactions per second. ArXiv preprint, 2018. https://arXiv:1805.03870.
[12] K. Croman, C Decke, I. Eyal, A.E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E.G. Sirer, D.S. An, and R. Wattenhofer. On Scaling decentralized blockchains. In 3rd Workshop on Bitcoin and Blockchain Research (2016). https://link.springer.com/chapter/10.1007/978-3-662-53357-4_8.
[13] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in bitcoin. In Financial Cryptography and Data Security. 19th International Conference, FC 2015, pages 507–527, 2015.
[14] V. Bagaria, S. Kannan, D. Tse, G. Fanti, and P. Viswanath. Deconstructing the blockchain to approach physical limits. https://arxiv.org/abs/1810.08092.

MathCognify Technologies, Cheung Kong Center 19/F, 2 Queen's Road, Central, Hong Kong

Email address: eg(?)mathcognify.com