PRIORITY QUEUEING ANALYSIS OF TRANSACTION-CONFIRMATION TIME FOR BITCOIN

YOSHIKI KAWASE* AND SHOJI KASAHARA

Graduate School of Information Science
Nara Institute of Science and Technology
Takayama 8916-5, Ikoma, Nara 6300192, Japan

(Communicated by Wuyi Yue)

ABSTRACT. In Bitcoin system, a transaction is given a priority value according to its attributes such as the remittance amount and fee, and transactions with high priorities are likely to be confirmed faster than those with low priorities. In this paper, we analyze the transaction-confirmation time for Bitcoin system. We model the transaction-confirmation process as a queueing system with batch service, M/G\textsubscript{B}/1. We consider the joint distribution of numbers of transactions in system and the elapsed service time, deriving the mean transaction-confirmation time. Using the result, we derive the recursive formulae of mean transaction-confirmation times of an M/G\textsubscript{B}/1 queue with priority service discipline. In numerical examples, we show the effect of the maximum block size on the mean transaction-confirmation time, investigating the accuracy region of our queueing model. We also discuss how the increase in micropayments, which are likely to be given low priorities, affects the transaction-confirmation time.

1. Introduction. Bitcoin is an autonomous decentralized virtual currency that does not have a central server or an administrator, and it succeeds to prevent fraud such as multiple payment and impersonation by encryption and peer-to-peer network technologies [11]. Bitcoin can provide immediate and secure service of international money transfer, and it is expected to be used for micropayment such as small amount remittance and billing of a piece of Internet content, due to its low fee [4]. Bitcoin demand grows rapidly in recent years, and the average number of transactions per day in 2016 is 226,000, which is about twice as much as the previous year’s 125,000 [6].

The Bitcoin virtual-currency system is based on two data types: transaction and block. A transaction includes information indicating a specific amount of money transferred from sender(s) to recipient(s). On the other hand, a block consists of several transactions, and a newly generated block is confirmed by solving a puzzle-like problem, i.e., a mathematical problem based on a cryptographic hash algorithm for the block generation. This confirmation process is called mining, and a number of nodes called miners compete for finding its answer. The miner who finds its solution first becomes a winner, and adds the new block to the blockchain. At

2010 Mathematics Subject Classification. Primary: 68M20, 90B22; Secondary: 60K25.

Key words and phrases. Bitcoin, blockchain, transaction-confirmation time, priority queue.

The second author is supported by Grant-in-Aid for Scientific Research (B) No. 15H04008 and SCAT Foundation.
this time, the solution is also included in the new block, and a new mathematical problem for the next mining is created from this newly added block. The difficulty of the puzzle-like problem is automatically adjusted so that mining finishes in 10 minutes on average [1]. The miner who succeeds in mining receives a reward called a coinbase (currently 12.5 BTC\(^1\) in 2017) and transactions’ remittance fees in the block. Note that miners keep mining work\(^2\) in order to get a coinbase even when there are no user transactions to be included into a block.

One of technical issues in Bitcoin is low transaction-processing speed due to the maximum block size [5]. Currently the block size is limited to 1 Mbyte and the time for mining is 10 minutes in average. As a result, the number of transactions processed per second is very small. In addition, the Bitcoin system has a transaction-priority mechanism, in which each transaction is prioritized according to its remittance amount, the elapsed time from previous approval, and the transaction-data size. A transaction is included in a block according to its priority value and the fee paid by user in advance [1]. Transactions with low remittance and/or low fee are likely not to be included in a block when the transaction arrival rate is high. It is reported in [6] that the recent block size is approaching the maximum block size, and therefore reducing the transaction size by separating a part from a transaction [7] have been proposed, however, no agreement has been achieved for Bitcoin community\(^3\).

In order to consider the scalability issue of Bitcoin, it is important to quantitatively characterize the transaction-confirmation process in which the priority mechanism is taken into consideration. Since transactions are processed in a block basis, the transaction-confirmation process can be modeled as a single-server queueing system with batch service. In terms of the analysis for the single-server queue with batch service, the authors in [2, 3] consider an M/G\(^B\)/1 queue and analyze the joint distribution of the number of customers in the queue and the elapsed service time.

In [8], the authors consider the impact of a fee amount on the transaction-confirmation time. They model the transaction-confirmation process as an M/G\(^B\)/1 queue with priority mechanism, deriving the mean transaction-confirmation time for each transaction-priority class. They also show from both measured data and extreme-value theory that the block-generation time follows an exponential distribution. From numerical examples, it is found that most of miners don’t follow the transaction inclusion specified for the default Bitcoin client mechanism [1]. In the default mechanism, a newly arriving transaction is included in the block under mining if the number of transactions in the block is smaller than the maximum block size. Comparing measured data and analytical results, it is conjectured that a newly arriving transaction is not included in the block currently processed.

In our previous work [9], we developed the queueing model of [8] into the one in which a newly arriving transaction is not served in the current service even when the number of transactions in the server is smaller than the maximum block size.

---

\(^1\)BTC is a unit of currency for Bitcoin.

\(^2\)The coinbase, the reward for the winner of a puzzle-like problem, is generated even for a block which includes no user transactions. A primary incentive for miners is to get the coinbase, and hence the miners keep mining work even in the absence of user transactions.

\(^3\)A large portion of miners agree with the segwit2x, in which the limit of block size is extended to 2 Mbytes and the transaction size is reduced by separating signature [6, 7, 17]. The rest of the miners have conducted developing another coin called Bitcoin cash via hard fork in August 1st, 2017. In Bitcoin cash, the block size limit can be immediately changed up to 8 Mbytes [16].
As we noted before, miners keep mining work even when there are no transactions to be included into a block. We assumed the multiple-vacation policy in which the server takes vacation repeatedly until a transaction arrives at the system in idle. We analyzed the joint distribution of the number of transactions in the server, that in the queue, and the elapsed service time, deriving the mean transaction-confirmation time. In the Bitcoin system, each transaction is given a priority before confirmation processing. The priority of a transaction is dependent on its amount, the transaction size in bit, the fee for miners, and the age of input coins, i.e., the time elapsed from the latest usage of the coin. For example, if the amount of a transaction is small and the coin in the transaction was used by another recent transaction, the transaction is given a low priority. When the amount of a transaction is high, on the other hand, a high priority is given to the transaction.

In this paper, we further develop the queueing model of [9] to the one in which priority mechanism is taken into consideration. Using the analytical result of [8], we calculate the mean transaction-confirmation time of an M/G/1 queue with priority service discipline, comparing the results with measurement data. Note that the queueing model of [8] is not work conserving, and hence they approximately analyzed the mean transaction-confirmation time of each priority class assuming that the system is work conserving. In our queueing model, on the contrary, the system is work conserving and hence we can analyze the mean transaction-confirmation time in a rigorous manner. In numerical examples, we quantitatively evaluate the effect of the block size on the transaction-confirmation time. We also investigate how the demand of low-priority transactions affects the transaction-confirmation time.

The rest of this paper is organized as follows. Section 2 summarizes the priority mechanism of the Bitcoin system. We describe our basic queueing model in Section 3, and the analysis of the queueing model is presented in Section 4. We consider the transaction-confirmation time of a queueing system with priority mechanism in Section 5. Some numerical examples are shown in Section 6, and we discuss how the increase in micropayment affects the transaction-confirmation time. Finally, we conclude the paper in Section 7.

2. Priority mechanism. There exists much literature on the introduction of Bitcoin. The readers are referred to e.g. [15]. In this section, we summarize the priority mechanism of default Bitcoin client [1].

In the Bitcoin system, the maximum block size is limited to 1 Mbyte. This implies that when a large number of transactions are issued in one-block mining period, all the transactions cannot be included into a new block. The reference Bitcoin client, called Bitcoin Core, constructs a new block in the following manner. When a miner constructs a new block, the miner selects transactions in the memory pool according to a priority-service discipline based on the amount of remittance, the transaction-data size, transaction fee, and the age of the unspent transaction

4In [8], it is assumed that a newly arriving transaction is included in the block being processed if the resulting number of transactions in the block is smaller than or equal to the maximum block size. Under this assumption, consider the following two sample paths for busy period. The first (resp. second) sample path is that a low-priority (resp. high-priority) transaction arrives at the system in idle and starts a busy period. Please note that the remaining service time of high-priority transactions in the first sample path is likely to be smaller than that in the second one. This implies the system is not work conserving.
output (UTXO). The age of a UTXO is the number of blocks that have confirmed since the UTXO was registered in the blockchain.

Let $p$ denote the priority value of a transaction. The value of $p$ is calculated by

$$p = \frac{1}{l} \sum_{k} v_k \cdot a_k,$$

where $l$ is the transaction data size, $v_k$ is the UTXO of the $k$th bitcoin address in the transaction, and $a_k$ is the corresponding UTXO age.

After priority allocation, the transaction is stored into a new block according to the following steps.

1. 50 Kbytes in the block are allocated to the highest-priority transactions, regardless of transaction fee.
2. Transactions with a fee of at least 0.00001 BTC/Kbyte are added to the block in highest-fee-per-kilobyte transactions first order. This process continues until the block size is no more than DEFAULT_BLOCK_MAX_SIZE, which is set to 750 Kbytes as default but can be changed within 1 Mbyte.
3. When the block size reaches the block-size limit, the transactions that are not included into the block remain in the memory pool of miners. Then the procedure is repeated.

In [10], the authors study trends of Bitcoin transaction fee conventions by analyzing the transaction fees paid with 55.5 million transactions that are recorded in the blockchain. They find that the confirmation time of transactions without fee are longer than those with fee. It is also reported that there are no significant difference between transaction-confirmation times for different fees. If the demand of transactions for micro payment increases in future, those transactions may suffer from a very long confirmation time because payers of micro payment are not willing to pay fee and the resulting priority of their transactions is low.

In order to verify the impact of the priority mechanism on the transaction-confirmation time, we analyzed the transaction data in [6] from October 2013 to September 2015. Table 1 shows mean transaction-confirmation times. Here, classless implies the mean confirmation time of all the transactions in the data. In terms of priority classes, we classify a transaction which has more than 0.001 BTC (100,000 satoshis) in the remittance amount as high-priority class, and that with the remittance amount smaller than or equal to 0.001 BTC as low-priority one.

In Table 1, the first (resp. second) row shows the results of period October 2013 to September 2014 (resp. October 2014 to September 2015), and the third row shows the results of two-year data. (In the following, we call the period of October 2013 - September 2014 (resp. October 2014 to September 2015) the 1st (resp. 2nd) period. The two-year period of October 2013 - September 2015 is called the overall one.) We observe that for classless, high, and low priorities, the results of the 2nd period are greater than those of the 1st one. In high priority case, the transaction-confirmation time is almost the same for those periods. In low priority case, the mean transaction-confirmation time for the 2nd period is significantly larger than that for the 1st period. We will discuss these observations in Section 6, where we compare the analytical results with trace-driven simulation.

---

5In this data, confirmation times of some transactions are smaller than or equal to zero second. We ignore them for calculating the transaction-confirmation time.
Table 1. Mean transaction confirmation time for each priority

| Period         | Classless [s] | Low priority [s] | High priority [s] |
|----------------|---------------|------------------|------------------|
| 1st (2013/10/1-2014/9/30) | 1060.67797    | 1947.47115       | 1044.25043       |
| 2nd (2014/10/1-2015/9/30) | 1171.98866    | 4887.56496       | 1070.32818       |
| overall (2013/10/1-2015/9/30) | 1124.13286    | 3888.06977       | 1059.06133       |

3. Basic queueing model. In this section, we describe a basic queueing model without priority mechanism. We will use the analytical result of the basic queueing model for deriving mean transaction-confirmation times for a priority queueing model later.

Transactions arrive at the system according to a Poisson process with rate $\lambda$. The transactions are grouped into a block, and the block is confirmed when one of miners finds the answer of the puzzle-like problem. We define the block-generation time as the time interval between consecutive block-confirmation time points. Note that the block-generation time can be regarded as the service time for our queueing model.

Let $S_i (i = 1, 2, \ldots)$ denote the $i$th block-generation time. We assume $\{S_i\}$’s are independent and identically distributed (i.i.d.), and follow a distribution function $G(x)$. Let $g(x)$ denote the probability density function of $G(x)$. The mean block-generation time $E[S]$ is given by

$$E[S] = \int_0^\infty x \, dG(x) = \int_0^\infty x \, g(x) \, dx.$$  

Transactions arriving to the system are served in a batch manner, and the maximum batch size is $b$.

When a transaction arrives at the system, the transaction enters the queue. The transaction cannot enter the server at its arrival point even when the batch size under service is smaller than $b$ or when the system is idle. In other words, the arriving transaction is served in the next block-generation time or later. This service is regarded as the gated service with multiple vacations [13], in which vacation periods are i.i.d. and follow the same distribution of the service time.

4. Analysis. In this section, we analyze the transaction-confirmation time for the basic model presented in the previous section.

Let $N_s(t)$ denote the number of transactions in the server at time $t$, $N_q(t)$ the number of transactions in the queue at time $t$, and $X(t)$ the elapsed service time at $t$. We define $P_{m,n}(x,t)(m = 0, 1 \ldots, b, \ n = 0, 1, \ldots, x, t \geq 0)$ as

$$P_{m,n}(x,t) \, dx = Pr \left\{ N_s(t) = m, N_q(t) = n, x < X(t) \leq x + dx \right\}.$$

Let $\xi(x)$ denote the hazard rate of the service time $S$, which is given by

$$\xi(x) = \frac{g(x)}{1 - G(x)}.$$  

When $\lambda E[S] < b$ holds, the system is stable and limiting probabilities exist. Letting $P_{m,n}(x) = \lim_{t \to \infty} P_{m,n}(x,t)$, we obtain from the assumptions the following

---

Note that the mining process for a block without transactions is identical to that for a block with transactions. This enables us to assume that vacation periods follow the same distribution of the service time.
differential-difference equations
\[
\frac{d}{dx} P_{m,n}(x) = -\{\lambda + \xi(x)\} P_{m,n}(x) + \lambda P_{m,n-1}(x), \quad 0 \leq m \leq b, \quad n \geq 1, \quad (1)
\]
\[
\frac{d}{dx} P_{m,0}(x) = -\{\lambda + \xi(x)\} P_{m,0}(x), \quad 0 \leq m \leq b. \quad (2)
\]
In (1), the first term of the right-hand side (r.h.s.) is obtained from the event that the numbers of transactions in the server and the queue does not change during a small time interval, and the second term comes from the event that a transaction arrives at the system with \( n - 1 \) transactions in the queue. The equation of (2) is the special case of no transaction in the queue, and hence the r.h.s. contains only the term for no event occurring during a small time interval.

We have the following boundary conditions at \( x = 0 \)
\[
P_{b,n}(0) = \sum_{m=0}^{b} \int_{0}^{\infty} P_{m,n+b}(x) \xi(x) dx, \quad n \geq 0,
\]
\[
P_{m,n}(0) = 0, \quad m = 0, 1, \ldots, b - 1, \quad n \geq 1,
\]
\[
P_{k,0}(0) = \sum_{m=0}^{b} \int_{0}^{\infty} P_{m,k}(x) \xi(x) dx, \quad k = 0, 1, \ldots, b.
\]
In the above equations, note that \( P_{k,0}(0) \) (\( 0 \leq k \leq b \)) is the probability that the number of transactions in the server is \( k \) and that in the queue is 0 at the beginning of a new service. This event can occur if the number of transactions in the server is \( m \) (\( 0 \leq m \leq b \)) and that in the queue is \( k \) at the end of the previous service.

Similarly, \( P_{m,n}(0) \) for \( n \geq 1 \) is the probability that the number of transactions in the server is \( m \) and that in the queue is \( n \geq 1 \) at the beginning of a new service. Reminding that transactions are served in a batch manner with the maximum batch size of \( b \), the event of \( P_{m,n}(0) \) for \( n \geq 1 \) can occur only if the number of transactions in the queue is greater than or equal to \( b + n \) at the end of the previous service. That is, the event of non-empty queue at the beginning of a service can occur with the event of \( b \) transactions in the server. This is why \( P_{m,n}(0) > 0 \) for \( n \geq 1 \) and \( m = 0, 1, \ldots, b - 1 \) is zero.

The normalizing condition is given by
\[
\sum_{n=0}^{\infty} \sum_{m=0}^{b} \int_{0}^{\infty} P_{m,n}(x) dx = 1.
\]

We define the following probability generating functions (pgf’s)
\[
P(z_1, z_2; x) = \sum_{n=0}^{\infty} \sum_{m=0}^{b} P_{m,n}(x) z_1^m z_2^n, \quad (3)
\]
\[
P(z_1, z_2) = \int_{0}^{\infty} P(z_1, z_2; x) dx. \quad (4)
\]
From (1) and (2), we obtain
\[
\sum_{n=0}^{\infty} \sum_{m=0}^{b} \frac{d}{dx} P_{m,n}(x) z_1^m z_2^n
\]
\[
= \sum_{n=0}^{\infty} \sum_{m=0}^{b} -\{\lambda + \xi(x)\} P_{m,n}(x) z_1^m z_2^n + \lambda z_2 \sum_{n=0}^{\infty} \sum_{m=0}^{b} P_{m,n}(x) z_1^m z_2^n.
\]
From the above equation and (3), we obtain
\[
\frac{d}{dx}P(z_1, z_2; x) = -\{\lambda + \xi(x)\}P(z_1, z_2; x) + \lambda z_2 P(z_1, z_2; x) \\
= -\{\lambda(1 - z_2) + \xi(x)\}P(z_1, z_2; x).
\]

From this differential equation, \( P(z_1, z_2; x) \) is given by
\[
P(z_1, z_2; x) = P(z_1, z_2; 0) e^{-\lambda(1 - z_2) x} \{1 - G(x)\}. \quad (5)
\]

Multiplying (5) by \( \xi(x) \) and integrating the equation yield
\[
\int_0^\infty P(z_1, z_2; x)\xi(x) \, dx = P(z_1, z_2; 0) G^*(\lambda - \lambda z_2), \quad (6)
\]
where \( G^*(s) \) is the Laplace-Stieljes transform (LST) of \( G(x) \) and given by
\[
G^*(s) = \int_0^\infty e^{-sx} dG(x).
\]

From (3) and (6), we obtain
\[
P(z_1, z_2; 0) G^*(\lambda - \lambda z_2) = \int_0^\infty \sum_{n=0}^{\infty} \sum_{m=0}^{b} P_{m,n}(x)\xi(x) \, dx \, z_1^m z_2^n. \quad (7)
\]

Substituting \( x = 0 \) into (3) yields
\[
P(z_1, z_2; 0) = \sum_{n=0}^{\infty} \sum_{m=0}^{b} P_{m,n}(0) \, z_1^m z_2^n \\
= \sum_{n=0}^{\infty} \sum_{m=0}^{b} \int_0^\infty P_{m,n+b}(x)\xi(x) \, dx \, z_1^m z_2^n + \sum_{n=0}^{b-1} \sum_{m=0}^{b-1} \int_0^\infty P_{m,n}(x)\xi(x) \, dx \, z_1^n. \quad (8)
\]

Using (7) to the first term of (8), we obtain
\[
\sum_{n=0}^{\infty} \sum_{m=0}^{b} \int_0^\infty P_{m,n+b}(x)\xi(x) \, dx \, z_1^m z_2^n \\
= \left( \frac{z_1}{z_2} \right)^b \left\{ P(1, z_2; 0) G^*(\lambda - \lambda z_2) - \sum_{n=0}^{b-1} \sum_{m=0}^{b-1} \int_0^\infty P_{m,n}(x)\xi(x) \, dx \, z_1^n \right\}.
\]

Then, (8) can be rewritten as
\[
P(z_1, z_2; 0) = \left( \frac{z_1}{z_2} \right)^b \left\{ P(1, z_2; 0) G^*(\lambda - \lambda z_2) - \sum_{n=0}^{b-1} \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x) \, dx \, z_1^n \right\} \\
+ \sum_{n=0}^{b-1} \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x) \, dx \, z_1^n. \quad (9)
\]
Substituting \( z_1 = 1 \) into (9), we obtain
\[
P(1, z_2; 0) = \left( \frac{1}{z_2} \right)^b \left\{ P(1, z_2; 0)G^*(\lambda - \lambda z_2) - \sum_{n=0}^{b-1} \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x)\,dx\,z_2^n \right\}
+ \sum_{n=0}^{b-1} \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x)\,dx.
\]
Multiplying the above equation by \( z_2^b \) yields
\[
\{ z_2^b - G^*(\lambda - \lambda z_2) \} P(1, z_2; 0) = \sum_{n=0}^{b-1} (z_2^b - z_2^n) \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x)\,dx.
\]
From the above equation, we obtain
\[
P(1, z_2; 0) = \sum_{n=0}^{b-1} \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x)\,dx.
\]
where \( \alpha_n \) is given by
\[
\alpha_n = \sum_{m=0}^{b} \int_0^\infty P_{m,n}(x)\xi(x)\,dx.
\]
Applying Rouche’s theorem [13] to (10), we can show that the equation
\[
z_2^b - G^*(\lambda - \lambda z_2) = 0,
\]
has \( b \) roots inside \( |z_2| = 1 + \epsilon \) for a small real number \( \epsilon > 0 \). One of them is \( z_2 = 1 \).
Let \( z_{2,k}^* (k = 1, 2, ..., b - 1) \) denote the root of (11). From (10), we have the following \( b - 1 \) equations
\[
\sum_{n=0}^{b-1} \left\{ (z_{2,k}^b) - (z_{2,k}^*)^n \right\} \alpha_n = 0, \quad k = 1, 2, ..., b - 1.
\]
From (9) and (10), we have
\[
P(z_1, z_2; 0) = \left( \frac{z_1}{z_2} \right)^b \left\{ \sum_{n=0}^{b-1} (z_2^n - z_2^*)^n \alpha_n G^*(\lambda - \lambda z_2) - \sum_{n=0}^{b-1} \alpha_n z_2^n \right\}
+ \sum_{n=0}^{b-1} \alpha_n z_1^n.
\]
From (4) and (5), we obtain
\[
P(z_1, z_2) = P(z_1, z_2; 0) \int_0^\infty e^{-\lambda(1-z_2)x} \{1 - G(x)\}\,dx
= P(z_1, z_2; 0) \frac{1 - G^*(\lambda - \lambda z_2)}{\lambda(1 - z_2)}.
\]
Multiplying (14) by \( \lambda(1 - z_2) \) and partially differentiating it by \( z_2 \), we have
\[
\frac{\partial P(z_1, z_2)}{\partial z_2} \lambda(1 - z_2) - P(z_1, z_2) \lambda
= \frac{\partial P(z_1, z_2; 0)}{\partial z_2} \{1 - G^*(\lambda - \lambda z_2)\} - P(z_1, z_2; 0) \frac{\partial G^*(\lambda - \lambda z_2)}{\partial z_2}.
\]
Substituting $z_1 = z_2 = 1$ into the above equation, and noting that $P(1, 1) = 1$, we have

$$P(1, 1) = P(1, 1; 0)E[S] = 1.$$  

Multiplying (14) by $z_2^b(z_2^b - G^*(\lambda - \lambda z_2))$ in order to calculate $P(1, 1; 0)$, we have

$$P(z_1, z_2; 0) = \left\{\begin{array}{c} \sum_{n=0}^{b-1} \left( z_2^n - z_2^n \right) \alpha_n G^*(\lambda - \lambda z_2) \\ - z_2^b \left( z_2^b - G^*(\lambda - \lambda z_2) \right) \sum_{n=0}^{b-1} \alpha_n z_2^n \\ - z_2^b \left( z_2^b - G^*(\lambda - \lambda z_2) \right) \left( \sum_{n=1}^{b-1} \alpha_n z_2^n + \alpha_0 z_1 \right) \end{array}\right.$$

Partially differentiating the above equation by $z_2$ and substituting $z_1 = z_2 = 1$, we obtain under the stability condition of $b > \lambda E[S]$

$$P(1, 1; 0) = \frac{\sum_{n=0}^{b-1} (b-n) \alpha_n}{b - \lambda E[S]}.$$  

Hence, the normalizing condition is given by

$$\sum_{n=0}^{b-1} \frac{(b-n) \alpha_n}{b - \lambda E[S]} E[S] = 1. \quad (15)$$  

From (12) and (15), $\alpha_n$’s are uniquely determined. From (13) and (14), we have

$$P(z_1, z_2) = \left\{\begin{array}{c} \sum_{n=0}^{b-1} \left( z_2^n - z_2^n \right) \alpha_n G^*(\lambda - \lambda z_2) \\ \sum_{n=0}^{b-1} \alpha_n z_2^n \\ \sum_{n=0}^{b-1} \alpha_n z_2^n \end{array}\right.$$

Partially differentiating (16) by $z_1$ and substituting $z_1 = z_2 = 1$, we obtain the mean number of transactions in the server as

$$\left( \frac{\partial P(z_1, z_2)}{\partial z_1} \right)_{z_1=1, z_2=1} = \left( \frac{\sum_{n=0}^{b-1}(b-n)\alpha_n}{b - \lambda E[S]} E[S] \right) \lambda E[S]$$

$$= \lambda E[S].$$

Similarly partially differentiating (16) by $z_2$, and substituting $z_1 = z_2 = 1$, we obtain the mean number of transactions in the queue as

$$\left( \frac{\partial P(z_1, z_2)}{\partial z_2} \right)_{z_1=1, z_2=1} = \frac{1}{2(b - \lambda E[S])} \left( \lambda^2 E[S^2] - 2b(b - \lambda E[S]) - b(b - 1) \right)$$

$$+ \sum_{n=0}^{b-1} \left( \lambda E[S^2](b-n) + E[S](b(b-1) - n(n-1)) + 2bE[S](b-n) \right) \alpha_n.$$
Hence, the mean number of transactions in the system $E[N]$ is given by

$$
E[N] = \frac{1}{2(b - \lambda E[S])} \left( \lambda^2 E[S^2] - b(b - 1) - 2(b - \lambda E[S])^2 + \sum_{n=0}^{b-1} \left\{ \lambda E[S^2](b - n) + E[S] \{b(b - 1) - n(n - 1)\} + 2bE[S](b - n) \right\} \alpha_n \right).
$$

Let $T$ denote the transaction-confirmation time, the time interval from the arrival time point of a transaction to its departure one. From Little’s theorem, the transaction-confirmation time is given by

$$
E[T] = \frac{1}{2\lambda(b - \lambda E[S])} \left( \lambda^2 E[S^2] - b(b - 1) - 2(b - \lambda E[S])^2 + \sum_{n=0}^{b-1} \left\{ \lambda E[S^2](b - n) + E[S] \{b(b - 1) - n(n - 1)\} + 2bE[S](b - n) \right\} \alpha_n \right)
\equiv f(\lambda).
$$

(17)

5. **Priority queue analysis.** We consider the same queueing model as the basic one except the transaction arrival process. Suppose that transactions are classified into $c$ priority classes. For $1 \leq i, j \leq c$, $i$ class transactions have priority over transactions of class $j$ when $i \leq j$. Let $\lambda_i$ ($i = 1, 2, \ldots, c$) denote the arrival rate of $i$-class transactions. We assume that $\sum_{i=1}^{c} \lambda_i E[S] < b$. We define $T_i$ as the sojourn time of class $i$ transactions. For simplicity, we introduce the following notation

$$
\bar{\lambda}_i = \sum_{k=1}^{i} \lambda_k, \quad i = 2, 3, \ldots, c.
$$

(18)

When the system is busy (resp. idle), an arriving transaction must wait for the remaining block-generation time (resp. vacation period) regardless its priority. Note that the vacation period follows the same distribution as the block generation time. Therefore, this priority queueing system with batch service is work conserving. We obtain

$$
f(\bar{\lambda}_c) = \sum_{k=1}^{c} \frac{\lambda_k}{\bar{\lambda}_c} E[T_k],
$$

(19)

where $E[T_k]$ is the sojourn time of class $k$ transactions. Since class 1 transactions are served similarly to the batch service analyzed in the previous section, $E[T_1]$ is given by

$$
E[T_1] = f(\lambda_1).
$$

(20)

Note that for $i < j$, any class-$j$ transactions don’t affect the service of class-$i$ transactions. In other words, $T_i$ is independent of transactions whose priority class is lower than $i$, and hence (19) holds not only $c$ but also $i = 2, 3, \ldots, c - 1$. This yields

$$
f(\bar{\lambda}_i) = \sum_{k=1}^{i} \frac{\lambda_k}{\bar{\lambda}_i} E[T_k]
$$
\[
\sum_{k=1}^{i-1} \frac{\lambda_k}{\lambda_i} E[T_k] + \frac{\lambda_i}{\lambda_i} E[T_i].
\]

We then obtain
\[
E[T_i] = \frac{1}{\lambda_i} \left( \bar{\lambda}_i f(\bar{\lambda}_i) - \sum_{k=1}^{i-1} \lambda_k E[T_k] \right), \quad i = 2, 3, \ldots, c. \tag{21}
\]

\(E[T_i]\)'s can be calculated recursively by (20) and (21).

In the following section of numerical examples, we consider two priority-class case: high and low. Let \(\lambda_H\) and \(\lambda_L\) denote the arrival rate of high-priority transactions and that of low-priority ones, respectively. Let also \(T_H\) and \(T_L\) denote the sojourn time of high-priority transactions and that of low-priority ones, respectively. In this two priority-class case, we obtain
\[
E[T_H] = f(\lambda_H),
\]
\[
E[T_L] = \left( \frac{\lambda_H}{\lambda_L} + 1 \right) f(\lambda_H + \lambda_L) - \frac{\lambda_H}{\lambda_L} f(\lambda_H).
\]

**Remark 1.** As we noted in Introduction, the analytical results in [8] were approximately derived due to the lack of work conserving property. Though the resulting equations in this subsection are the same as [8], those hold in a rigorous sense for the queueing model in the present paper.

6. Numerical examples.

6.1. **Basic parameter setting.** It is reported in [8] that the distribution of the block-generation time \(G(x)\) is an exponential one given by
\[
G(x) = 1 - e^{-\mu x},
\]
where \(\mu = 0.0018378995\). Then, \(E[S]\) and \(E[S^2]\) are given by
\[
E[S] = \frac{1}{\mu} = 544.0993884, \quad E[S^2] = \frac{2}{\mu^2} = 592088.2889,
\]
The LST of \(G(x)\) is given by
\[
G^*(s) = \frac{\mu}{s + \mu}.
\]
With these settings, we calculate the mean transaction-confirmation time \(E[T]\).

In order to calculate mean confirmation times of high- and low-priority transactions, we define \(\eta\) as the ratio of the transaction-arrival rate for high-priority class to that for low-priority one. We analyzed mean arrival rates of \(\lambda_H\) and \(\lambda_L\) from the data of October 2013 to September 2015 in [6]. According to the result, we set
\[
\eta = \frac{\lambda_H}{\lambda_L} = \frac{0.9349109906}{0.0360010622}.
\]
With \(\eta\), \(\lambda_H\) and \(\lambda_L\) are expressed as
\[
\lambda_H = \frac{\eta \lambda}{1 + \eta}, \quad \lambda_L = \frac{\lambda}{1 + \eta}.
\]
By using these relations, we can calculate mean transaction-confirmation times of high and low priority classes, keeping the ratio of \(\lambda_H\) to \(\lambda_L\) constant.
6.2. Model validation.

6.2.1. Comparison of analysis and simulation. In order to confirm the validity of our analysis, we first conduct the Monte-Carlo simulation with the same basic queueing model in Section 3. Figure 1 shows the comparison of analysis and simulation for the transaction-confirmation time. In this figure, the horizontal axis represents the transaction arrival rate $\lambda$ and the vertical one is the mean transaction-confirmation time $E[T]$. The block size is fixed at $b = 1000$ in the numerical simulation. It is shown from Figure 1 that the analytical result is the same as simulation, confirming the validity of analysis for the basic queueing model.

In Figure 2, we consider the queueing model with two priority classes. Here, the horizontal axis represents the transaction arrival rate $\lambda$, which is the sum of transaction arrival rates of high- and low-priority transactions, and the vertical one is the mean transaction-confirmation time $E[T]$. The block size is also fixed at $b = 1000$. From Figure 2, the analytical results of the two priority cases agree with the simulation, confirming the validity of analysis.

6.2.2. Comparison of analysis and measurement. Next, we compare the analysis and measurement. In [8], the authors analyzed two-year transaction data obtained from [6], reporting statistics such as the block-generation time, number of transactions in a block, and transaction-confirmation time. From [8], the mean transaction size is 571.34 bytes, and we set the maximum block size $b$ equal to 1750.

Table 2 shows mean transaction-confirmation times of analysis and measurement. The analytical result is calculated with the mean transaction-arrival rate equal to 0.97091, which is obtained from the measured data. We observe in this table that the analytical result is almost the same as the measurement value with relative error of 0.43% in the classless case. This result confirms that the analytical model
in the present paper captures the behavior of transactions more accurately than the model in [8], in which the mean transaction-confirmation time in classless case is calculated as 568.10 [s].

In terms of priority, the measurement result of high-priority class transactions is slightly larger than the analytical one, while we observe a large discrepancy between measurement and analytical results for low-priority transactions. This result implies that the priority queueing model considered in the paper does not capture the real behavior of Bitcoin transactions. A possible conjecture of this discrepancy is that transactions with low priority are likely to be excluded from the block generation process.

6.2.3. **Comparison of analysis and trace-driven simulation: classless case.** We conduct trace-driven simulation experiments for further validating our analytical model. From the two-year data of October 2013 to September 2015 in [6], we collected the sequences of the transaction-arrival time and block-generation one. We used the two sequences such that transactions arrive at the system according to the arrival-time sequence and blocks are generated according to the block-generation-time sequence. We investigate how the block size affects the transaction-confirmation time.
Figures 3 and 4 represent the transaction-confirmation time against the block size. In Figure 3, we use the trace data measured from October 2013 to September 2014 (the 1st period), while the simulation result of Figure 4 is based on the trace data measured from October 2014 to September 2015 (the 2nd period). Figure 3 shows a good agreement of analysis and simulation, while we observe in Figure 4 a large discrepancy between analysis and simulation when the block size is small.

In order to clarify the reason of these discrepancies, we investigate how the transaction-arrival process evolves over time. Figure 5 shows the mean transaction-arrival rate per day. In this figure, we observe little variation during the first 12 months, while the mean transaction-arrival rate significantly varies for the last three months in the measured period.

Table 3. Coefficients of variation for transaction interarrival time.

| Period               | 1st   | 2nd   | Overall         |
|----------------------|-------|-------|-----------------|
|                      | 2013/10-2014/09 | 2014/10-2015/09 | 2013/10-2015/09 |
| Classless            | 3.72401      | 15.32505     | 10.17893        |

Table 3 shows coefficients of variation for transaction interarrival times in the three measurement periods: the 1st period, 2nd period, and the overall period (October 2013 - September 2015). In this table, the coefficient of variation of the 2nd period is larger than that of the 1st period. This large coefficient of variation of the 2nd period results in a large coefficient of variation of the overall period, causing the discrepancy between analysis and simulation.
Figure 4. Transaction-confirmation time vs. block size, 2nd period (October 2014 to September 2015). \( \lambda = 1.2081311, \mu = 0.0017009449, \) and the coefficient of variation of transaction inter-arrival time: 15.3250509.

When the block size is large, there is enough space to include transactions in the next block, and hence burst transaction arrivals are likely to be served in the
next block. This causes little difference between analysis and simulation. On the other hand, when the block size is small, the system is likely to be congested due to the bursty nature of the transaction-arrival process. This results in a larger transaction-confirmation time for simulation than that for analysis.

![Figure 6](image_url)

**Figure 6.** Transaction-confirmation time vs. block size, October 2013 to September 2014. $\lambda_H = 0.7203544$, $\mu = 0.0019748858$.

### 6.2.4. Comparison of analysis and trace-driven simulation: two-priority case.

In this subsection, we consider the validation of the two-priority queueing model. Figures 6 and 7 show the transaction-confirmation time of high priority transaction against the block size. In Figure 6, we use the trace data of the 1st period (October 2013 - September 2014), while the simulation result of Figure 7 is based on the trace data of the 2nd period (October 2014 - September 2015). Figure 6 shows a good agreement of analysis and simulation, while we observe in Figure 7 a discrepancy similar to the Figure 4.

Figures 8 and 9 show the transaction-confirmation time of low priority transaction against the block size. In Figure 8, we use the trace data of 1st period, while the simulation result of Figure 9 is based on the trace data of the 2nd period. From Figure 8, the analysis agrees well with simulation, while we observe in Figure 9 a large discrepancy between analysis and simulation.

In terms of the coefficient of variation for the interarrival time, the value of the 2nd period is greater than that of the 1st period in both high- and low-priority transaction cases. (See Table 4.)

The above observations suggest that the priority queueing model works well when the variation of the transaction-interarrival time is small and the maximum block size is large. When the transaction-interarrival time varies greatly or when the maximum block size is small, however, the priority queueing model does not work well for predicting the mean transaction-confirmation time.
As we discussed in subsection 6.2.2, our priority queueing model does not capture the real behavior of Bitcoin transactions, in particularly for low-priority transactions. Further study of the priority mechanism of Bitcoin is needed and this is our future work.
Figure 9. Transaction-confirmation time vs. block size, October 2014 to September 2015. $\lambda_L = 0.0586636$, $\mu = 0.0017009449$.

Table 4. Coefficients of variation for transaction interarrival time, two-priority classes.

| Period                  | 1st                  | 2nd                  | Overall              |
|-------------------------|----------------------|----------------------|----------------------|
|                         | 2013/10-2014/09      | 2014/10-2015/09      | 2013/10-2015/09      |
| Low Priority            | 1.66569              | 3.90654              | 3.01387              |
| High Priority           | 3.70045              | 14.94895             | 9.99103              |

6.3. Impact of the block size on the transaction-confirmation time. In this section, we investigate the effect of the block size on the transaction-confirmation time.

Figure 10 shows the analytical results with block size $b = 1000, 2000, 4000,$ and $8000$. We observe that the transaction-confirmation time grows with the increase in the arrival rate. From [8], the maximum number of transactions included in the current maximum block size of 1 Mbyte is approximately given by $b = 1750$. This value is close to $b = 2000$, diverging around $\lambda = 3.6$.

We also observe that enlarging the block size results in a small transaction-confirmation time. However, the transaction-confirmation time for $b = 8000$ rapidly increases when $\lambda$ is greater than 13 transactions per second. This implies that enlarging the block size does not solve the scalability issue fundamentally.

6.4. Impact of micropayments: Two-priority case. In this subsection, we consider how the mean transaction-confirmation time is affected by priority mechanism of Bitcoin. We consider two scenarios in terms of the growth of transaction-arrival rates. The first scenario is the growth of micropayments, in which we investigate how the increase in the arrival rate of low-priority transactions affects the mean transaction-confirmation time. In the second scenario, we consider the growth
of Bitcoin-user population. We increase the overall transaction-arrival rate, keeping the ratio of the arrival rate of low-priority transactions to that of high-priority ones.

6.4.1. Impact of increase in the low priority transaction. Figure 11 shows mean transaction-confirmation times of two-priority transactions. We calculate the mean transaction-confirmation time for each priority class by analytical results for the block size $b=1000$, 2000, 4000, and 8000. In this figure, the transaction-arrival rate of the high priority ($\lambda_H$) is fixed at 0.96292, and that of the low priority ($\lambda_L$) is changed.

In Figure 11, the confirmation time for high-priority transactions is constant in each block-size case. This implies that the transaction-confirmation time for high-priority class is insensitive to the increase in the arrival rate of low-priority transactions. For $b = 2000$, the transaction-confirmation time for low-priority class rapidly increases when $\lambda_L = 2.0$. In case of $b = 4000$, which is almost twice larger than the current maximum block size, the transaction-confirmation time exponentially grows around $\lambda_L = 6.0$. We also observe the same exponential growth for case of $b = 8000$. These results suggest that enhancing the maximum block size does not improve the transaction-confirmation time for low-priority class.

6.4.2. Impact of population growth in Bitcoin. Figures 12 and 13 show mean transaction-confirmation times of high-priority class and low-priority one, respectively. In both figures, we observe the rapid increase in the transaction-confirmation time for any $b$. For example, when the block size is $b = 2000$, the transaction-confirmation time of high-priority class rapidly increases at around $\lambda_H = 3.5$ in Figure 12, and that with low priority grows at around $\lambda_L = 3.0$ in Figure 13. Similarly, when $b = 4000$, the confirmation-time of high-priority class diverges at around $\lambda_H = 7.0$ in Figure 12, and that of low-priority class rapidly increases at around
\( \lambda_L = 5.5 \) in Figure 13. These results suggest that enhancing the maximum block size is effective to mitigate the rapid growth of the transaction-confirmation time. However, its improvement is not enough to solve the scalability issue of Bitcoin.
7. **Conclusion.** In this paper, we analyzed the transaction-confirmation time for Bitcoin using a single-server queue model with batch service M/G^B/1 and priority service discipline. In this queueing model, newly arriving transactions are temporarily stored in the queue first even when the number of transactions in the server is smaller than the batch size. Assuming that the priority of a transaction depends only on its remittance amount, we derived the mean transaction-confirmation time of each priority class. We validated the analytical results by simulation experiments, and evaluated effects of the block size and transaction-arrival rate on the transaction-confirmation time. We found that the transaction-confirmation time can be decreased by changing the maximum block size. However, its improvement is not effective enough to increase the number of transactions processed per unit time.

The numerical results also indicated that the inaccuracy of the model comes from the assumption of the Poisson arrival process. In future work, we will develop a queueing model with a general arrival process, which captures bursty nature of real transaction-arrival process.

Recently, one of schemes solving the scalability issue of Bitcoin is lightening network, which provides a channel dedicated to micropayment transactions [12]. The lightening network is expected to mitigate the overloaded block-generation process, however, it is not clear how the lightening network decreases the transaction-confirmation time. We will also develop analytical models for the lightening network as future work.

**Acknowledgments.** This research was supported in part by SCAT Foundation, and Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research (B) No. 15H04008.
REFERENCES

[1] A. M. Antonopoulos, Mastering Bitcoin, O’Reilly, 2014.
[2] M. L. Chaudhry and J. G. C. Templeton, The queuing system $M/G^B/1$ and its ramifications, European Journal of Operational Research, 6 (1981), 56–60.
[3] M. L. Chaudhry and J. G. C. Templeton, A First Course in Bulk Queues, John Wiley & Sons, 1983.
[4] http://www.meti.go.jp/committee/kenkyukai/sansei/fintech_kadai/pdf/003_02_00.pdf
[5] http://www.coindesk.com/1mb-block-size-today-bitcoin/
[6] https://blockchain.info/
[7] http://www.coindesk.com/segregated-witness-bitcoin-block-size-debate/
[8] S. Kasahara and J. Kawahara, Effect of Bitcoin fee on transaction-confirmation process, arXiv:1604.00103[cs.CR].
[9] Y. Kawase and S. Kasahara, Transaction-Confirmation Time for Bitcoin: A Queueing Analytical Approach to Blockchain Mechanism, The 12th International Conference on Queueing Theory and Network Applications (QTNA2017), Qinhuangdao, China, August 21-23, 2017.
[10] M. Möser and R. Böhme, Trends, tips, tolls: A longitudinal study of bitcoin transaction fees, Financial Cryptography and Data Security, Lecture Notes in Computer Science, Springer, 8976 (2015), 19–33.
[11] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available from: https://bitcoin.org/bitcoin.pdf.
[12] J. Poon and T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, https://lightning.network/lightning-network-paper.pdf, 2016.
[13] H. Takagi, Queueing Analysis: A Foundation of Performance Evaluation, Vol. 2. Finite systems. North-Holland Publishing Co., Amsterdam, 1993.
[14] https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
[15] F. Tschorsch and B. Scheuermann, Bitcoin and beyond: A technical survey on decentralized digital currencies, IEEE Communications Surveys & Tutorials, 18 (2016), 2084–2123.
[16] https://www.bitcoincash.org/
[17] https://www.coindesk.com/segwits-slow-rollout-bitcoins-capacity-hasnt-seen/

Received October 2017; revised April 2018.

E-mail address: kawase.yoshiaki.km3@is.naist.jp
E-mail address: kasahara@is.naist.jp