Discrete-Time Analysis of Wireless Blockchain Networks

Francesc Wilhelmi
Centre Tecnològic de Telecomunicacions de Catalunya
(CTTC/CERCA)
Castelldefels, Barcelona, Spain
fwilhelmi@cttc.cat

Lorenza Giupponi
Centre Tecnològic de Telecomunicacions de Catalunya
(CTTC/CERCA)
Castelldefels, Barcelona, Spain
lorenza.giupponi@cttc.es

Abstract—**Blockchain (BC) technology can revolutionize future networks by providing a distributed, secure, and unalterable way to boost collaboration among operators, users, and other stakeholders. Its implementations have traditionally been supported by wired communications, with performance indicators like the high latency introduced by the BC being one of the key technology drawbacks. However, when applied to wireless communications, the performance of BC remains unknown, especially if running over contention-based networks. In this paper, we evaluate the latency performance of BC technology when the supporting communication platform is wireless, specifically we focus on IEEE 802.11ax, for the use case of users' radio resource provisioning. For that purpose, we propose a discrete-time Markov model to capture the expected delay incurred by the BC. Unlike other models in the literature, we consider the effect that timers and forks have on the transaction confirmation latency.**

Index Terms—**blockchain, discrete-time analysis, IEEE 802.11ax**

I. INTRODUCTION

One of the main challenges to face for Beyond 5G (B5G) networks is the need to foster the economic sustainability of the increasingly complex mobile networks. In this regard, a new economy of sharing has long been proposed by the main standardization bodies in contexts such as spectrum sharing, RAN/network sharing or network slicing. The exchange of services or resources among actors in these scenarios could be significantly automated and secured by the introduction of Blockchain (BC) to enable an economically driven and flexible network management. Through BC, providers can operate from the cloud to trade radio and network resources as-a-service, enabling novel trends such as network slicing or spectrum sharing with programmable and scientific trust.

A BC is a decentralized, distributed record of transactions stored in a permanent and near inalterable way. Unlike traditional databases, typically administered by a central entity, BCs rely on a peer-to-peer (P2P) network of so-called miners that no single party can control. Authentication of transactions is achieved through cryptographic means and a mathematical consensus protocol that determines the rules by which the

ledger is updated. This technology is useful to provide immutability, transparency, and security, which allows addressing multiple challenges in B5G networks. Besides the Bitcoin use case, BC technology has already been proposed to enhance multiple procedures in communications and networks [1], and some prominent examples are content delivery [2], network slicing [3], RAN sharing [4], or network management [5], [6]. Nevertheless, BC entails several widely documented technological challenges, such as heavy computational costs, scalability issues, high latency, and energy consumption [7].

While most of the BC applications and deployments in the real world are designed considering a stable wired communication environment, in this paper, we focus on a more challenging wireless network as the communication infrastructure of our BC. Specifically, we propose a BC-enabled resource provisioning approach where end users can apply for radio and network resources on-demand, according to an as-a-service vision, without being bound in the long term to a specific operator. The needed exchange of transactions happens over a wireless network. Without loss of generality, we consider BC transactions to be exchanged over an IEEE 802.11ax network, to account also for the additional challenge of decentralization in the unlicensed spectrum. In such types of contention-based networks, channel access is ruled by contention-based mechanisms such as Carrier Sense Multiple-Access (CSMA), which perform poorly as user density increases. In a contention-based BC network, users and miners implement a random backoff mechanism to transmit transactions and blocks. This unleashes a competition for wireless resources that can potentially degrade the performance of the BC.

In this context, we propose to model a wireless BC network through a batch-service queue based on a discrete-time Markov model that characterizes the latency introduced in the network. These aspects are key to the overall wireless system design. The longer it takes to distribute transactions and blocks in the BC network, the more the end-users performance is jeopardized, and the BC system becomes unstable, unreliable, and susceptible to forks. Based on the proposed model, we evaluate the resulting service latency using the Bianchi model [8] for IEEE 802.11ax. In close spirit to this paper, we find the works in [9], [10], where the Bianchi model was also used to characterize IEEE 802.11 links through which the BC

operation takes place. In contrast to the available literature, our model considers the effect of timers and forks. A timer (or timeout) is the maximum time for mining a block in a BC. Through timers, it is possible to avoid waiting until the maximum block size is reached, thus ensuring that blocks are mined periodically. Moreover, a fork is a split of a BC, which occurs as a side effect of distributed consensus mechanisms. By potentially invalidating transactions, forks incur into additional transaction confirmation delay. We demonstrate through simulation results that the proposed discrete-time Markov model perfectly captures the wireless BC behavior when also forks and timers are considered. Model analysis, as a function of different key parameters, allows gaining interesting insight on the dropping probability, the delay introduced by the BC (for which it is possible to identify an optimal block size), and the characteristic of the fork probability.

II. CHARACTERIZATION OF BLOCKCHAIN: RELATED WORK

A BC is a type of distributed ledger technology (DLT) that compiles transactions in blocks that are sequentially chained one after the other. The record of the transactions is maintained across several computers which communicate through a P2P network. Nodes enabled to add transactions to the BC are referred to as miners. The consensus mechanism allows the BC to operate without the need to rely on a central trusted central authority. This automation of transactions is one of the most valuable features of BC for trading in future mobile networks. BC was firstly introduced with the cryptocurrency Bitcoin [11]. Lately, and enabled by the advent of smart contracts (i.e., computer programs that self-execute the terms of a contract when specific conditions are met) introduced by Ethereum [12], the potential of BC was unleashed and incorporated into multiple domains [13].

The BC operation has been modeled in literature in multiple ways, ranging from analytical models to experimental tools [14]. In the analytical domain, one of the tools gaining importance is batch service queuing [15], [16], whereby packets leave the queue in batches, rather than individually. The batch property of these kinds of queues intuitively captures the BC idea of chains of blocks and has been already used to model the behavior of multiple technologies in communications [17] [18] [19].

Accordingly, the consensus-based confirmation procedure can be seen as a single-server batch service queue where packets (transactions) are served (mined) in batches (blocks). In this setting, the most relevant trade-off lies within the batch size and the fork probability. In both cases, the queuing delay is affected in a non-straightforward manner, and the most appropriate setting depends on factors like the arrivals rate, the number of miners, the implemented consensus mechanism, or the mining time (alternatively, the mining difficulty). Batch service queue models for BC have recently been introduced in [20], followed by the approaches in [21]–[25].

First, the work in [20] proposed a batch service queue model to understand the stochastic behavior of the transaction-confirmation process in Bitcoin. Later, this model was extended in [21] to improve its accuracy for short block sizes, which was achieved by modeling transaction inter-arrivals with a general distribution. To do so, a matrix analytic method was used to derive the steady-state distribution of the queue model states, which were defined by the number of transactions in the queue just before new transaction arrivals. This model was validated with trace-driven simulations, and results showed that exponential-type distributions are accurate for estimating the mean confirmation time in Bitcoin. Similarly, the work in [22] attempted to overcome the complexity of the model in [20], [21] by separating the batch service queue into two separate processes, corresponding to block generation and block mining, respectively. This approach was later generalized by the same authors in [24]. Finally, a further evaluation was provided in [25], where a slightly different discrete-time queue model was used to study game-theoretical aspects related to the pricing associated with mining transactions.

Differently to the approaches discussed in [20]–[22], [24], in this work we model queue states at departure instants, which was also considered in [23]. Through this approach, the authors of [23] were able to provide a more detailed performance evaluation of BC, on Ethereum. Unlike what proposed until the date, we further extend the batch service queue to incorporate the key effects of timers and forks within the BC operation. To the best of our knowledge, none of these features has been considered before in any BC model. We believe that these features are fundamental to any BC model for multiple reasons. First, timers are important to break the dependence of the mining procedure on the arrivals rate, which allows guaranteeing a maximum waiting delay, even if blocks are not filled with transactions (i.e., blocks are mined periodically). Disregarding the effects of timers leads to poor accuracy when deriving the queue status (e.g., the number of transactions in it) or the expected delay [26]. Second, the forks are a dramatic source of instability in BC. The appropriate modeling of their behavior ensures high accuracy and fidelity to capture the BC network's real dynamics.

III. SYSTEM MODEL

A. Overview of the scenario

End users are not necessarily bound to a specific operator in the long term but, at any moment, can trigger requests to get specific data services. Based on the BC technology, the User Equipment (UE) devices issue smart contracts that facilitate automatic, transparent, and auditable mechanisms whereby operators and service providers fulfill UE requests. The smart contracts (including service indicators such as duration, maximum latency, or throughput) are registered in a public BC, which enables flexible and decentralized on-demand network access.

In the proposed scenario, UE smart contracts are submitted to the closest AP [10], which, besides providing radio access to the UEs, act as peer nodes in the BC P2P network. An AP is responsible to carry out tasks such as broadcasting transactions, storing a copy of the BC, achieving consensus, or

mining blocks. In this paper, we focus on analyzing the delays to get the UE requests to reach the mobile operators, leaving for future work the study of how the service is provided by the selected operator to the UEs.

B. Blockchain delays

In the proposed BC-enabled communication environment, peers (miners) generate candidate blocks with transactions (service requests) from the UEs, which are mined once the block size S_B is reached, or when a timer T_w expires. To mine a block, peer nodes run a given consensus mechanism such as Proof-of-Work (PoW) and then propagate it over the P2P network. Upon successful block propagation, the winning miner is allowed to append the candidate block to the BC.

To assess the feasibility of the BC operation, it is important to identify its associated delays, especially when running over wireless links. Despite providing security and decentralization capabilities, the delay associated to the BC may impact the performance of the underlying applications if the service latency is too high (e.g., V2X communications), and can also introduce instability in the BC, due to the increased risk of forks. With the proposed BC-enabled procedure, we identify the following steps for confirming a UE transaction:

1) **Smart contract upload:** first, the UE selects the closest peer node to submit a smart contract to indicate the requirements of the requested service. P2P nodes verify and propagate the received transactions over the P2P network. At this stage, it is important to collect valid, non-conflicting transactions (e.g., to prevent double-spending). The delay for uploading smart contracts (T_{up}) depends on the capabilities of the communication links between UEs and peer nodes (e.g., IEEE 802.11).

2) **Queuing:** Valid transactions are queued before being included into candidate blocks. The queuing delay (T_q) depends on the number of arrivals, the mining rate, and the block size. Transactions are included into the candidate block till the block size is reached, or the timer T_w expires. The queuing delay is characterized in Section IV by modeling the BC as a batch-service queuing system.

3) **Block generation:** Peer nodes execute the consensus algorithm until finding the block's nonce, or until receiving a new valid mined block. As widely considered in the literature [27], [28], the block generation time (T_{bg}) is modeled as an exponential random variable with parameter μ, which is assumed to be the same for all the miners. For instance, in Bitcoin's PoW, the mining time is fixed to an average of ten minutes by adjusting difficulty based on miners' computational power.

4) **Block propagation:** The winning miner generates a new block, based on its candidate block, which is distributed through the P2P network with a delay of T_{bp}. Once the mined blocks are propagated, the included transactions are put into effect.

Notice that steps (2)-(4) may be repeated as a result of forks. A fork can be seen as a side effect of the consensus mechanism and the communication and distribution delay required for the synchronization of the BC, which leads to splitting the BC into different paths, each one representing a potentially different state of the BC. If a miner different than the winner (i.e., the miner that mined the block in the shortest amount of time) succeeds in generating a new block while the first successful block is still being propagated, some uninformed miners may mistakenly add the second block to their version of the BC. Assuming that M miners with the same exponentially distributed block generation rate μ start the mining process synchronously, thus allowing to model the time between mining events as a Poisson inter-arrival process, the fork probability can be defined as:

$$p_{\text{fork}} = 1 - \prod_{\forall i \neq w} \Pr(T_{\text{bg}}(i) - T_{\text{bg}}(w) > T_{\text{bp}})$$
$$= 1 - e^{-\mu(M-1)T_{\text{bp}}}, \tag{1}$$

where $T_{\text{bg}}(w)$ is the winner's block generation delay. It can be noticed that, for a fixed mining capacity, the higher the propagation delay, the higher the fork probability and the more unstable the BC will be.

To sum up, and taking into account the effect of forks, the BC operation incurs the following transaction confirmation delay to make a UE request effective:

$$T_{\text{BC}} = T_{\text{up}} + \frac{(T_q + T_{\text{bg}} + T_{\text{bp}})}{1 - p_{\text{fork}}} \tag{2}$$

IV. BATCH SERVICE QUEUE MODEL

We consider a batch-service queuing approach to model the time a transaction spends in the ledger before being validated. In a finite-length $M/M^s/1/K$ queue, s denotes the number of packets served altogether. The batch characteristic is useful to represent the BC operation, where the mining procedure starts when the block size S_B is reached, or when the timer (T_w) expires (see Fig. 1).
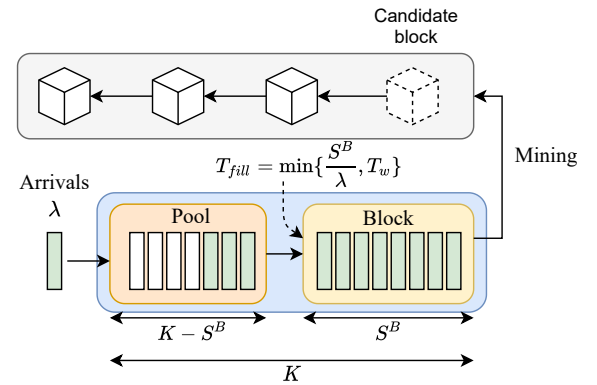

Fig. 1. Blockchain procedure through batch-service queuing.

In particular, we are interested in calculating the average number of packets in the queue (E[Q]), which allows deriving the expected queuing delay (E[D]), including both T_q and T_{bg}. For that purpose, we first model the queue status at departure instants $\boldsymbol{\pi}^d$, as done in [17]. Then, we derive the

queue's steady-state distribution $\boldsymbol{\pi}^s$ by solving a finite number of system of equations. We use Little's law [29] to obtain the expected delay, so that $E[D] = \frac{E[Q]}{\lambda(1-p_b)}$, where λ is the total arrival rate, considering all the UEs, and $p_b = \pi_K^s$ is the blocking probability (the probability of discarding a transaction because the queue of length K is full). Finally, the expected queue occupancy is obtained as $E[Q] = \sum_{k=0}^{K} k\pi_k^s$.

A. Departures Distribution

We define the state of the batch service queue at the n-th departure instant as:

$$q_n = \min\left\{q_{n-1} + a(q_{n-1}) - s(q_{n-1}), K - s(q_{n-1})\right\}, \quad (3)$$

where q_{n-1} is the state of the queue after the previous departure, $a(q_{n-1})$ is the number of new arrivals during the last inter-departure epoch, $s(q_{n-1})$ is the number of packets served in a batch, and K is the queue length. Based on this, we derive the departure probabilities by considering the set of feasible states $\boldsymbol{\Omega}(q_n)$ that are reachable from any state q_n. To further illustrate this, Fig. 2 shows an example of the possible transitions from state $q_n = S_B+1$. From that state, $a(q_{n-1})$ is limited by K (the maximum reachable state is $K-S_B+1$), and the minimum reachable state is upper bounded by $q_n - s(i)$.
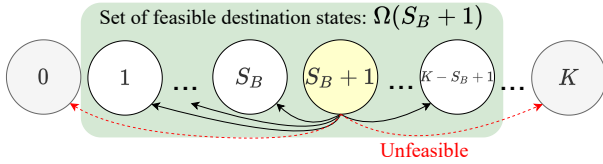

Fig. 2. Discrete-time Markov chain representing the feasible departure states from reference state $S_B + 1$.

In general, the set of feasible destination states $\boldsymbol{\Omega}(i)$ that can be reached from state i is $j \in \left[[i - s(i)]^+, K - s(i)\right]$. Notice that any transition to $j > K - s(i)$ is not possible because the transitions to be served are kept in the queue until the block is mined. Under the assumption that arrivals and departures follow independent Poisson and exponential distributions, respectively, and by being aware of the possible inter-departure transitions, the probability $p_{i,j}$ for reaching a state j from i is as follows:

$$p_{i,j} = \begin{cases} \int_0^\infty e^{-\lambda t}\frac{(\lambda t)^j}{j!} \cdot \mu e^{-\mu t}dt, & [i-s(i)]^+ \le j < K - s(i) \\ 1 - \sum_{l=0}^{K-s(i)-1} p_{i,l}, & j = K - s(i) \\ 0, & \text{otherwise} \end{cases}$$
$$(4)$$

The first case ($[i-s(i)]^+ \le j < K - s(i)$) indicates that the maximum queue size is not reached with new transactions. In turn, the second case ($j = K - s(i)$) refers to the situation in which the queue is filled because the system received, at least, $K - s(i)$ transactions during the inter-departure epoch. Other transitions are unfeasible. Since arrivals and departures

follow independent Poisson and exponential distributions, respectively, the probability $p_{i,j}$ for $[i - s(i)]^+ \le j < K - s(i)$ can be rewritten as:

$$p_{i,j} = \frac{\mu}{\mu + \lambda}\left(\frac{\lambda}{\mu + \lambda}\right)^{j-(i-s(i))} \quad (5)$$

The departure probability distribution $\boldsymbol{\pi}^d$ can be obtained from the transition-probability matrix P, which is built with probabilities $p_{i,j}$. In particular, $\boldsymbol{\pi}^d$ can be calculated by solving $\boldsymbol{\pi}^d = \boldsymbol{\pi}^d\text{P}$, provided that $\boldsymbol{\pi}^d\mathbf{1}^T = 1$.

B. Steady-state Distribution

From the departures distribution $\boldsymbol{\pi}^d$, the steady-state queue occupancy distribution $\boldsymbol{\pi}^s$ can be derived thanks to the Poisson Arrivals See Time Averages (PASTA) property [30] (or random observer property). The PASTA property states that, for Poisson processes, the queue status at any arbitrary time, π_k^s, is equal to the state probability at which random arrivals find the queue, π_k^a.

To compute the steady-state distribution, we must first consider the time the queue spends in each state i, $T(i)$, which, as shown earlier, is divided into filling and mining periods. In particular, the time $T(i)$ the system spends until the next departure from state i is given by:

$$T(i) = \min\{T_w, [S_B - i]^+/\lambda\} + \frac{1}{\mu}, \quad (6)$$

which depends on the probability $\tau(i)$ that the timer T_w expires before the minimum block size S^B is reached from state i. In particular, for any state $i < S^B$, the timer expiration probability is given by:

$$\tau(i) = \sum_{j=i}^{S^B-1} e^{-\lambda T_w}\frac{(\lambda T_w)^j}{j!} \quad (7)$$

With the departures distribution $\boldsymbol{\pi}^d$ and expected time spent in a given departure state, the steady-state probability of a state $k \in [0, K]$ is derived as in Eq. 8. Notice that, to include the effect of the timer for $k < K$, we need to consider the exact amount of packets that can be observed during the filling procedure, which conditions the probability of observing state k in the mining phase. In particular, from any departure state i, we consider the probability of observing exactly n packets (denoted by $P(n|\tau)$) independently, given that $n < S^B - i$ and that the timer expires.

C. Considering forks

To capture the effect of forks with the model, we need to consider that forks affect the mining procedure in two main ways:

1) The fact that several miners work concurrently affects the mining rate.
2) Transactions involved in a fork are re-added to the pool of unconfirmed transactions (i.e., the queue). Throughout this work, we assume the worst-case scenario where all the transactions involved in a fork are re-added.

$$\pi_k^s = \begin{cases} \frac{1}{\lambda E[T]} \sum_{i=0}^{k} \pi_i^d \left(\overline{\tau}(i) \left(\sum_{j=k-s(i)+1}^{K-s(i)} p_{i,j} \right) + \tau(i) \left(\sum_{j=i}^{S^B-1} P(n=j-i|\tau) \left(\sum_{l=k-s(j-i)+1}^{K-s(j-i)} p_{j,l} \right) \right) \right), & k < K \\ 1 - \sum_{i=0}^{K-1} \pi_i^s, & k = K \end{cases} \qquad (8)$$

Regarding implication 1), we need to consider the first order statistics of the individual exponential random variables characterizing the mining time.

Definition 1. Let X_i be an exponential random variable and $X_{(k)}$ be the k-th smallest instance among n occurrences $X_1, ..., X_n$. Assuming that $X_{(i)}$ is a reordered sequence of X_i, s.t. $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$, then $X_{(1)}$ is a random variable $\mathrm{Exp}(n\lambda)$ whose distribution is $f_{X_{(1)}}(x) = n\lambda e^{-n\lambda x}$, and whose expected value is $E[X_{(1)}] = \frac{1}{n\lambda}$.

With this, we capture the competition among miners (the first one to mine a block, wins), which leads to characterizing the mining time as a joint exponential distribution $\mathrm{Exp}(n\lambda)$.

Now, assuming synchronization among miners and that the same transactions are included in any block replica, we address implication 2) by modifying the behavior of the queue when a departure occurs. This affects the transition probability matrix P and the blocking probability p_b, since transactions may not leave the queue after a departure epoch, provided that forks occur. In particular, the number of served transactions from departure state i, namely $s(i)$, is defined as follows:

$$s(i) = \bar{p}_{\text{fork}} \cdot \min\{i, S^B\} + p_{\text{fork}} \cdot t_{b,w} \notin \mathcal{T}_{b,i \neq w}, \qquad (9)$$

where $t_{b,w} \notin \mathcal{T}_{i \neq w}$ is the set of transactions included in winner's block b that have not been affected by the fork (i.e., the set of non-overlapping transactions). Assuming that miners include exactly the same set of transactions to every block replica, $t_{b,w} = \emptyset$.

V. PERFORMANCE EVALUATION

In this Section, we validate the proposed batch-service queue model and evaluate the performance of BC IEEE 802.11ax networks in cellular-based random deployments.

A. Simulation Parameters

To characterize the IEEE 802.11ax links used by the BC nodes, we use the Bianchi's Distributed Coordination Function (DCF) model [8], which allows estimating the saturation throughput Γ of DCF overlapping devices as a two-step probability function:

$$\Gamma = \frac{p_{\text{slot,s}} E[L]}{E[T_{\text{slot}}]}, \qquad (10)$$

where $p_{\text{slot,s}}$ is the probability of transmitting successfully at a given slot, $E[L]$ is the average payload length, and the average duration of a generic slot $E[T_{\text{slot}}]$. Considering that Request To Send / Clear To Send (RTS/CTS) is enabled, the duration

of each type of slot (empty, successful, and collision slots, respectively) is as follows:

$$\begin{cases} T_{\text{slot,e}} = 9\mu s \\ T_{\text{slot,s}} = T_{RTS} + 3 \cdot T_{SIFS} + T_{CTS} + T_{DATA} + T_{ACK} \\ T_{\text{slot,c}} = T_{RTS} + T_{DIFS} \end{cases}$$
$$(11)$$

The Bianchi model is useful to capture collisions and transmission deferrals in fully-overlapping networks. As for signal propagation effects, we have considered a standard log-distance path loss model with shadowing effects. In particular, the loss observed between nodes i and j is:

$$PL(d_{i,j}) = PL_0 + 10\alpha \log_{10}(d_{i,j}) + \frac{\sigma}{2} + \frac{\gamma}{2} \frac{d_{i,j}}{10} \qquad (12)$$

Table I details the parameters used in this Section.

TABLE I
SIMULATION PARAMETERS

	Parameter	Description	Value
Depl.	N	Num. of APs/cells	19
	R	Cell radius	10 m
PHY	B	Bandwidth	20 MHz
	F_c	Carrier frequency	5 GHz
	MCS	Modulation and coding scheme	0-11
	$SUSS$	Single-user spatial streams	1
	T_{PH}	PHY header duration	20 μs
	T_{OFDM}	OFDM symbol duration	4 μs
	P_t	Transmit power	20 dBm
	PL_0	Loss at the reference dist.	5 dB
	α	Path-loss exponent	4.4
	σ	Shadowing factor	9.5
	γ	Obstacles factor	30
MAC	$CW_{min/max}$	Min/max contention window	32
	$L_{D/ACK}$	Data and ACK lengths	12.000 / 32 bits
	$L_{RTS/CTS}$	RTS and CTS lengths	160 / 112 bits
	L_{MH}	MAC Header	320 bits
	N_a	Max. A-MPDU	1 frame
	T_{PPDU}	Max. PPDU duration	5,4884 μs
	$T_{DIFS/SIFS}$	DIFS/SIFS duration	34 / 16 μs
	T_e	Empty slot duration	9 μs
	CCA	CCA threshold	-82 dBm
BC	μ	Mining capacity	15 blocks/s
	L_T	Transaction length	3.000 bits
	Q	Batch service queue length	10 packets

B. Numerical Results

We start validating our queue model with simulation results.[1] Fig. 3 shows both model and simulation results of the queuing delay, for different user arrivals rates and block sizes. For all λ and S^B values, two different timers values ($T_w = 0.5$s and $T_w = 2$s) are selected to showcase the situations in which the timer is likely or not to expire, respectively, before filling a block. Besides, the presence or

[1] The batch-service queue simulator used for validations is open-access [31].

not of forks is included in the validation. As Fig. 3 shows, model results match with simulations, thus demonstrating that the model properly captures the effect of both timers and forks.
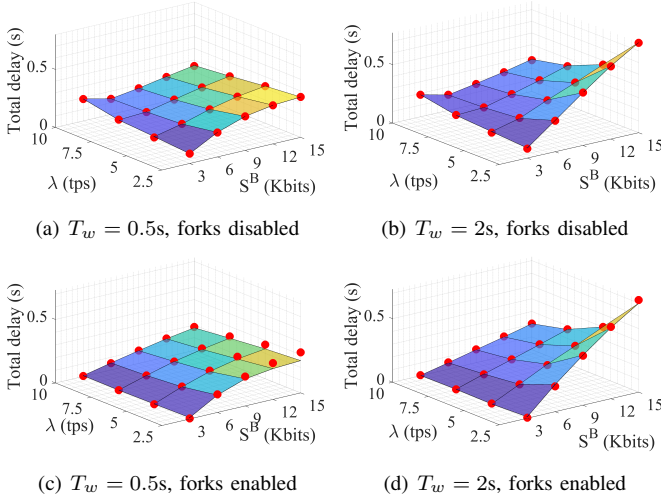

(a) $T_w = 0.5$s, forks disabled   (b) $T_w = 2$s, forks disabled

(c) $T_w = 0.5$s, forks enabled   (d) $T_w = 2$s, forks enabled

Fig. 3. Queuing delay for different UE arrivals rates (λ) in transactions per second (tps) and block size (S^B) in Kbits. The presence or not of forks is also evaluated. The solid areas correspond to model results, while the red circles correspond to simulation results.

From now on, we present further results obtained in closed-form through the presented model. In particular, to better illustrate the impact of the different parameters (λ, S^B and T_w), Fig. 4(a) and Fig. 4(b) show the queuing delay and the drop probability, respectively, observed with and without forks. For each considered block size S^B, we have averaged the results for different user arrivals, namely $\lambda = \{2.5, 5, 7.5, 10, 12.5, 15\}$ arrivals per second. This time, only model results are provided.
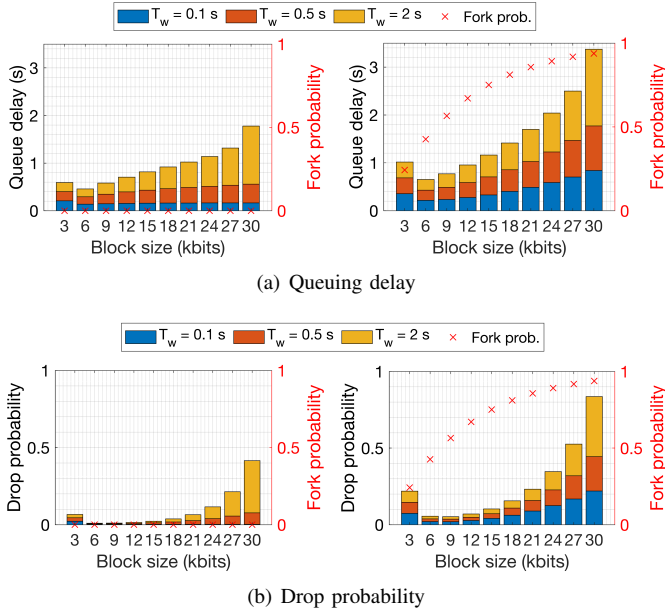

(a) Queuing delay

(b) Drop probability

Fig. 4. Analysis of the queuing delay and drop probability for different block size and timer values.

As shown, the queue delay is optimum at $S^B = 6$ Kbits. From that point, the delay increases with the block size, even if forks are enabled or not. As for the timer values, we observe that the queue delay is significantly reduced when the timer is low (e.g., $T_w = 0.5$ s). The timer is particularly useful to reduce BC latency when the number of arrivals is low and blocks are not filled with transactions at the necessary speed. Nevertheless, setting the timer to a low value contributes to generating more overhead and may also be counterproductive in terms of delay (notice that the mining time is insensitive to the number of transactions in a block).

As for the fork probability, it increases with the block size to almost one. Specifically, the higher the block size, the higher the block propagation time, which, for a fixed mining rate, contributes to making forks more likely to occur. An increase in the fork probability has a great impact on the number of drops, thus packet losses are noticed even for small timer values (i.e., even if the queue is not full).

Finally, we assess the performance of the BC-enabled resource provisioning application for different user densities. Fig. 5 shows the transaction confirmation latency of the wireless BC for validating transactions. We have considered up to 30 concurrent STAs with full-buffer traffic. The block size is set to the fixed value of 6 Kbits so that the queue delay is minimized, whereas the transactions arrival rate (λ) is set to 7.5 arrivals per second. Moreover, for each density, we have simulated 10 random deployments for averaging purposes.
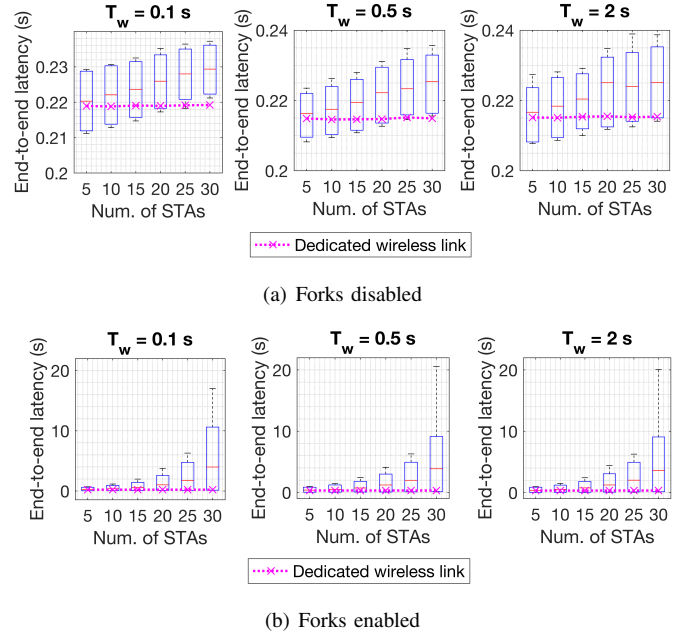

(a) Forks disabled

(b) Forks enabled

Fig. 5. Analysis of the transaction confirmation latency for different network densities. Results are shown for shared (boxplots) and dedicated (purple dashed lines) IEEE 802.11ax links.

As shown in Fig. 5(a), for the ideal case where no forks occur, the use of shared IEEE 802.11ax links is affordable even if network density increases. This is due to the low amount of network resources used by the BC application. In contrast,

when forks are considered (see Fig. 5(b)), the latency increases dramatically with the network density. The block propagation delay increases when the number of concurrent users is high. This leads to a high fork probability, which negatively impacts the number of attempts for validating transactions.

VI. Conclusions

In this paper, we have envisioned a future wireless system where users are not bound to a specific operator and apply for resources through smart contracts. BC allows secure and decentralized transactions among users and operators, but its latency performance represents a challenge for the service perception of the UE and for the stability of the BC, which will increasingly suffer from forks as the latency augments. While BC systems are normally based on wired P2P networks, in this paper we foresee the BC to be based on a wireless infrastructure. We have focused on IEEE 802.11ax, to add the additional challenge of contention in the performance evaluation. With the aim to investigate the wireless BC performance, we have analytically modeled the BC through a batch queue serving system based on a discrete-time Markov model. Results have revealed that our model perfectly captures the behaviors of the wireless BC, including timers and forks, which is novel in literature, and shown interesting optimal points of the block size and behaviors of the fork probability, as a function of both the network and BC are designed. The proposed queue model can be further extended to characterize aspects such as packet losses, burst arrivals, or different types of transactions.

References

[1] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5G and beyond networks: A state of the art survey," *Journal of Network and Computer Applications*, p. 102693, 2020.

[2] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 70–76, 2017.

[3] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain network slice broker in 5G: Slice leasing in factory of the future use case," in *2017 Internet of Things Business Models, Users, and Networks*. IEEE, 2017, pp. 1–8.

[4] X. Ling, J. Wang, T. Bouchoucha, B. C. Levy, and Z. Ding, "Blockchain radio access network (B-RAN): Towards decentralized secure radio access paradigm," *IEEE Access*, vol. 7, pp. 9714–9723, 2019.

[5] A. Asheralieva and D. Niyato, "Distributed dynamic resource management and pricing in the IoT systems with blockchain-as-a-service and UAV-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1974–1993, 2019.

[6] T. Maksymyuk, J. Gazda, M. Volosin, G. Bugar, D. Horvath, M. Klymash, and M. Dohler, "Blockchain-Empowered Framework for Decentralized Network Management in 6G," *IEEE Communications Magazine*, vol. 58, no. 9, pp. 86–92, 2020.

[7] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi E., Gn Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," *International Conference on Financial Cryptography and Data Security*, 2016.

[8] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

[9] B. Cao, M. Li, L. Zhang, Y. Li, and M. Peng, "How does CSMA/CA affect the performance and security in wireless blockchain networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4270–4280, 2019.

[10] C. E. Ngubo and M. Dohler, "Wi-Fi-Dependent Consensus Mechanism for Constrained Devices Using Blockchain Technology," *IEEE Access*, vol. 8, pp. 143 595–143 606, 2020.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer electronic cash system," Manubot, Tech. Rep., 2019.

[12] V. Buterin, "Ethereum: the ultimate smart contract and decentralized application platform," *Libro blanco de Ethereum. Consultado en http://web. archive. org/web/20131228111141/http://vbuterin. com/ethereum. html el*, vol. 8, 2013.

[13] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE Access*, vol. 6, pp. 53 019–53 033, 2018.

[14] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020.

[15] S. K. Bar-Lev, M. Parlar, D. Perry, W. Stadje, and F. A. Van der Duyn Schouten, "Applications of bulk queues to group testing models with incomplete identification," *European Journal of Operational Research*, vol. 183, no. 1, pp. 226–237, 2007.

[16] M. Chaudhry and J. Gai, "A Simple and Extended Computational Analysis of M/G j (a, b)/1 and M/G j (a, b)/1/(B+ b) Queues Using Roots," *INFOR: Information Systems and Operational Research*, vol. 50, no. 2, pp. 72–79, 2012.

[17] B. Bellalta, A. Faridi, D. Staehle, J. Barcelo, A. Vinel, and M. Oliver, "Performance analysis of csma/ca protocols with multi-packet transmission," *Computer Networks*, vol. 57, no. 14, pp. 2675–2688, 2013.

[18] X. Wen, B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu, "Ruletris: Minimizing rule update latency for tcam-based sdn switches," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 179–188.

[19] S. Kar, R. Rehrmann, A. Mukhopadhyay, B. Alt, F. Ciucu, H. Koeppl, C. Binnig, and A. Rizk, "On the throughput optimization in large-scale batch-processing systems," *Performance Evaluation*, vol. 144, p. 102142, 2020.

[20] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *International Conference on Queueing Theory and Network Applications*. Springer, 2017, pp. 75–88.

[21] Y. Kawase and S. Kasahara, "A batch-service queueing system with general input and its application to analysis of mining process for bitcoin blockchain," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CP-SCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1440–1447.

[22] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *International Conference on Computational Social Networks*. Springer, 2018, pp. 25–40.

[23] S. Geissler, T. Prantl, S. Lange, F. Wamser, and T. Hossfeld, "Discrete-time analysis of the blockchain distributed ledger technology," in *2019 31st International Teletraffic Congress (ITC 31)*. IEEE, 2019, pp. 130–137.

[24] Q.-L. Li, J.-Y. Ma, Y.-X. Chang, F.-Q. Ma, and H.-B. Yu, "Markov processes in blockchain systems," *Computational Social Networks*, vol. 6, no. 1, pp. 1–28, 2019.

[25] J. Qi, J. Yu, and S. Jin, "Nash equilibrium and social optimization of transactions in blockchain system based on discrete-time queue," *IEEE Access*, vol. 8, pp. 73 614–73 622, 2020.

[26] D. Claeys, B. Steyaert, J. Walraevens, K. Laevens, and H. Bruneel, "Tail probabilities of the delay in a batch-service queueing model with batch-size dependent service times and a timer mechanism," *Computers & operations research*, vol. 40, no. 5, pp. 1497–1505, 2013.

[27] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.

[28] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta, "The blockchain folk theorem," *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 2019.

[29] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2018, vol. 399.

[30] D. Gross and C. Harris, "Fundamentals of queueing systems," 1998.

[31] F. Wilhelmi and L. Giupponi, "Blockchain-oriented batch service queue simulator," https://doi.org/10.5281/zenodo.4680438, 2021.