


RESEARCH ARTICLE

Game-theoretic policy computing and simulation for blockchained buffering system via diffusion approximation

Wanyang Dai 

Department of Mathematics and State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
Corresponding author: Wanyang Dai; Email: nan5lu8@nju.edu.cn

Keywords: blockchained queueing buffer system; diffusion approximation; dynamic resource pricing; federated learning; game-theoretic scheduling; Nash equilibrium policy

MSC: 60J70; 60H35; 60K30; 65C20; 68M20; 90B15; 90B22; 90B36; 91A15

Abstract

We study 2-stage game-theoretic problem oriented 3-stage service policy computing, convolutional neural network (CNN) based algorithm design, and simulation for a blockchained buffering system with federated learning. More precisely, based on the game-theoretic problem consisting of both “win-lose” and “win-win” 2-stage competitions, we derive a 3-stage dynamical service policy via a saddle point to a zero-sum game problem and a Nash equilibrium point to a non-zero-sum game problem. This policy is concerning users-selection, dynamic pricing, and online rate resource allocation via stable digital currency for the system. The main focus is on the design and analysis of the joint 3-stage service policy for given queue/environment state dependent pricing and utility functions. The asymptotic optimality and fairness of this dynamic service policy is justified by diffusion modeling with approximation theory. A general CNN based policy computing algorithm flow chart along the line of the so-called *big model* framework is presented. Simulation case studies are conducted for the system with three users, where only two of the three users can be selected into the service by a zero-sum dual cost game competition policy at a time point. Then, the selected two users get into service and share the system rate service resource through a non-zero-sum dual cost game competition policy. Applications of our policy in the future blockchain based Internet (e.g., metaverse and web3.0) and supply chain finance are also briefly illustrated.

1. Introduction

In this paper, we study a blockchained buffering system with federated learning as shown in [Figure 1](#).

The main focus is of three folds: 2-stage game-theoretic problem oriented 3-stage service policy computing of users-selection and rate scheduling/dynamic pricing, convolutional neural network (CNN) based algorithm design, and simulation case studies. The game-theoretic problem consists of both zero-sum and non-zero-sum 2-stage game competitions (representing “win-lose” and “win-win” 2-stage competitions). Furthermore, the computed policy is proved to be asymptotically optimal and fair via diffusion approximation. The asymptotic optimality means that the whole workload and total cost of the system are asymptotically minimized. The asymptotic fairness means that no user can change his personal policy unilaterally for profit. The computed 3-stage service policy as shown in [Figure 2](#) is based on a 2-stage game-theoretic problem whose solution is represented by a saddle point to a zero-sum game problem and a Nash equilibrium point to a non-zero-sum game problem (see also the related concepts in Dai [8, 9], Marchi [19], Nash [21], and Rosen [25]). Furthermore, in the 2-stage game problems, each user has his own utility function in terms of price, queue length, and service rate. This utility function is a generalization of the existing one in Dai [6, 8, 9], Ye and Yao [28], and references therein. Examples of such a utility function can be the so-called proportionally fair allocation, minimal potential

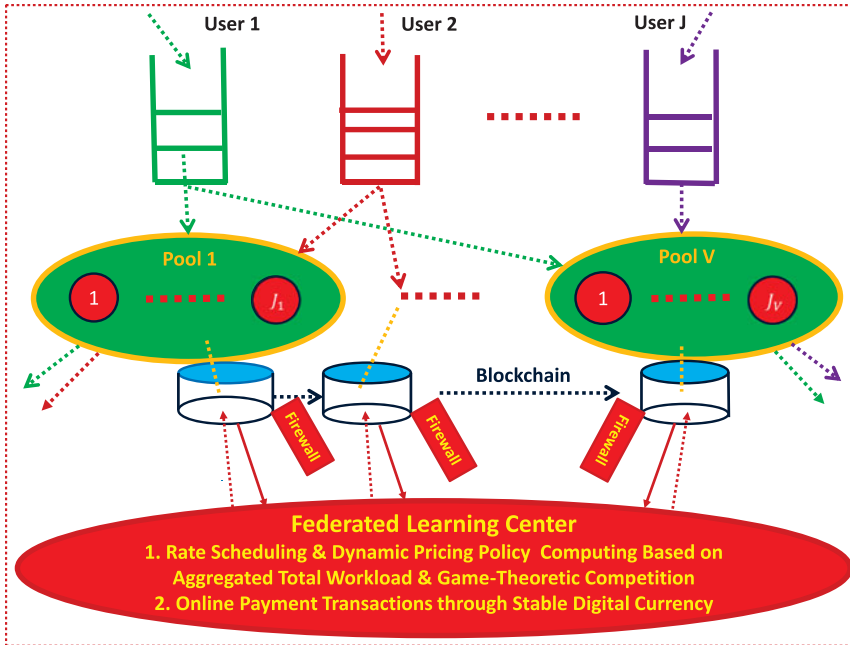


Figure 1. A blockchained buffering system with federated learning, which consists of J users and V pools.

delay allocation, and (β, α) -proportionally fair allocation (see e.g., Ye and Yao [28]), which are widely used in internet protocols and communication systems.

In Figure 1, we present such a generalized service system consisting of V service pools and J buffer queues corresponding to J -parallel users for two positive integer numbers V and J . A blockchain system is added to this system for security and distributed data storage. A federated learning center is also added to this system for dynamic policy computing and online payment transaction via stable digital currency. This blockchained buffering and federated learning system is a generalized platform of the recent studies (see e.g., Ayaz *et al.* [2], Dai [8, 9, 11], Demertzis *et al.* [14], Qu *et al.* [22]), which come up in different research areas such as metaverse, sixth generation of wireless communication (6G), internet of vehicles (IoV), web3.0, etc. Due to the security consideration of the system, blockchains are used to protect privacy among different users as shown in Figure 1. Moreover, the way developed in Dai [8, 9] uses a single-dimensional aggregated total workload process of the system to dynamically design general-dimensional decision parameter vector at each time point in the federated learning (FL) center. Then, the FL center sends the computed parameters back to their corresponding individual service pools, respectively, for local information upgrades and local model training. This idea to employ the single-dimensional aggregated workload process in the design of Dai [8, 9] is motivated from the state space collapse property (or more classically, the Little’s law) widely studied in queueing literature (see e.g., Bramson [3] and Little [17]). The purpose to use this idea in the processing of a multiclass queueing network environment is aimed to reduce the system’s dimension and to avoid the curse of dimensionality. When state space collapse phenomenon happens in a queueing system, the multiple class queueing processes will display a certain proportional relationship with the single-dimensional aggregated workload process, which will significantly reduce the system computational complexity. In the design of Dai [8, 9], the proportional relationship represented by the state space collapse property is generalized to be represented by a solution to a game-theoretic problem. This solution can be trained and computed in the FL center, which can be used as an online scheduling policy. Under certain traffic flow and service discipline assumptions, this dynamic policy is proved to be asymptotically fair

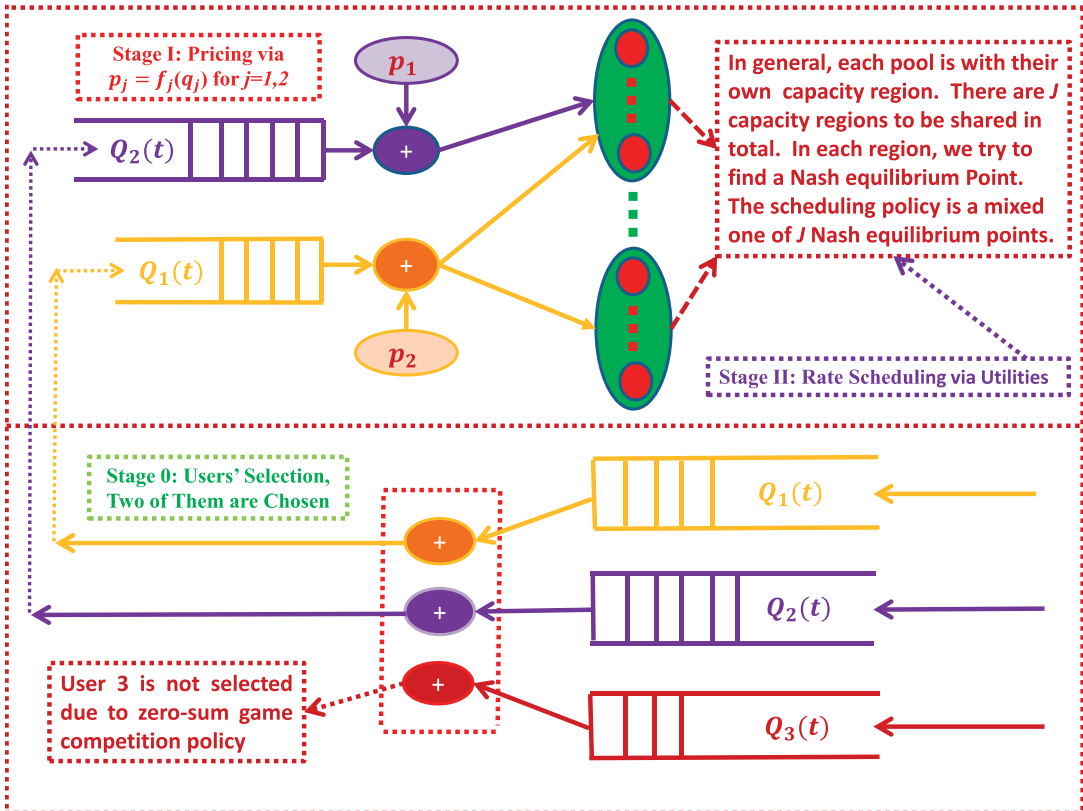


Figure 2. A 3-stage processing flow chart of users-selection, dynamic pricing, and rate scheduling for a multiple pool service system with J -users, where J is taken to be 3 for an illustration.

and optimal in certain sense (that will be elaborated later) through diffusion approximation under the so-called diffusive scaling and the well-known heavy-traffic regime. When the game-theoretic problem reduces to an optimization problem and without the blockchain security consideration, readers are also referred to Dai [6], Ye and Yao [28] for related studies. Furthermore, the studies in Dai [8, 9] consider a general multiple service pool system with a more general input flow process (i.e., a J -dimensional triply stochastic renewal reward process (TSRRP)). In the meanwhile, the studies in Dai [6], Ye and Yao [28] focus on the analysis when the input process is a conventional J -dimensional renewal process.

The contribution of this research is of three folds. The first fold is to add the dynamic pricing capability to the study in Dai [9]. The second fold is to design a general CNN based policy computing algorithm flow chart along the line of the so-called *big model* framework. The third fold is to provide detailed simulation examples. However, the main focus of our current paper is on the design and analysis of a joint 3-stage service policy extended from the previously mentioned 2-stage game-theoretic policy concerning users-selection, dynamic pricing, and online rate resource allocation for given queue/environment state dependent pricing and utility functions. The asymptotic optimality and fairness of this designed 3-stage service policy is justified by applying the well-known heavy traffic approximation and modeling technique together with our newly added dynamic pricing functionality. In the meanwhile, it is also supported by our newly conducted simulation case studies with three users and through explicitly constructing the solutions to their corresponding dual cost game competition problems. Note that, the allocated rate to a user is corresponding to the service time for the user. Furthermore, the service for the user includes four steps: user registration, security checking, dynamic policy computing, and online

payment transaction through stable digital currency as shown in [Figure 1](#). Thus, the service time for the user will be the summation of the processing times corresponding to the four service steps.

The design procedure is illustrated in [Figure 2](#), where J (e.g., $J = 3$) users want to receive services from the V service pools. However, only two of them can be selected to receive services at each time point according to chosen state dependent pricing and utility functions. After the selected two users get into services, they need to share the limited capacity from different service pools in a cooperative way. The selection process is determined by a dynamic policy corresponding to a solution (called a saddle point) to a zero-sum dual cost game competition problem at each particular time point. The sharing process for the selected two users is to compete the system rate service resource and is determined by a dynamic policy through a solution (called a Nash equilibrium point) to a non-zero-sum dual cost game competition problem. Note that, as shown in [Figure 2](#), the associated 2-stage game-theoretic problem is a $J \times V$ -dimensional problem. In general, an explicit solution is not available. Thus, we design a CNN based algorithm flow chart for general usage. However, to support our current designed policy, we choose smaller J and V to conduct simulation case studies, which is presented in [Sections 4–5](#).

The input data flows from different users to our system as indicated in [Figure 1](#) are characterized by a J -dimensional TSRRP that can be further approximated by a diffusion process with the target for our effective simulation. Furthermore, the service rate capacity available for resource-competing users at each pool is modeled as a randomly capacity region evolving with a finite state continuous Markov chain (FS-CTMC). The arrived data flows can immediately get into service if the servers are available. Otherwise, they will be stored in the queueing buffers waiting for service. Besides buffers, the decision information after service for each user will be stored in a distributed data base called blockchain as designed in [Figure 1](#).

There are many reasons (e.g., security, decentralization, smart contract) for today's FinTech system and future Internet (e.g., metavers and web3.0) to choose blockchain as a key technology (see, e.g., Buterin [4], Dai [8,9,11], Iansiti and Lakehani [16], Nakamoto [20], Rajan and Visser [23]). Blockchain consisting of data blocks is an orderly distributed database with encryption and is frequently referred to as a ledger. Each data block contains the proposed (or calculated) decision information with customer's private and public keys at a single time point with a time-stamp and a link to a previous block. The management of a blockchain can be realized via various smart contracts in a decentralized way. Traditionally, a smart contract within a blockchain can be considered as a digitalized regulation rule with common sense. In this paper, we will make the rule dynamically evolving according to an online decision-making policy, i.e., a solution to a dynamic game based competition problem.

More precisely, we will study the resource allocation and dynamic pricing of a joint saddle & Nash equilibrium service policy for such a system. When users get into service either from buffers or from outside of the system, the computation of their processing policy concerning resource allocation may depend on long history data stored in the blockchain (e.g., represented by a conditional mean defined process) and can be dynamically priced via stable digital currency. Each queue may be served at the same time through multiple smart service pools while each pool may also serve multiple queues simultaneously by running intelligent policies. Note that, to reflect the dynamic evolving nature of real-world systems and to realize the decentralized operation in a blockchain, the users to be selected at a time is random, the number of pools to serve a specific queue is random, and the number of queues to be served by a given pool is also random. The effectiveness of our proposed policy is in terms of revenue, profit, cost, system delay, etc. We model them through some utility (or hash) functions in terms of the performance measures of their internal data flow dynamics such as queue length and workload processes. To demonstrate the usefulness of our policy, we derive a reflecting diffusion with regime-switching (RDRS) model for the performance measures under our designed policy to offer services to different users in a cost-effective, efficient, and fair way. Based on this RDRS model, our proposed policy is effectively implemented with numerical simulations for some case studies of the system in [Figures 1–2](#). Concerning the numerical scheme through Monte Carlo simulation for a Brownian motion driven stochastic system, readers are referred to Dai [10] and references therein for more details.

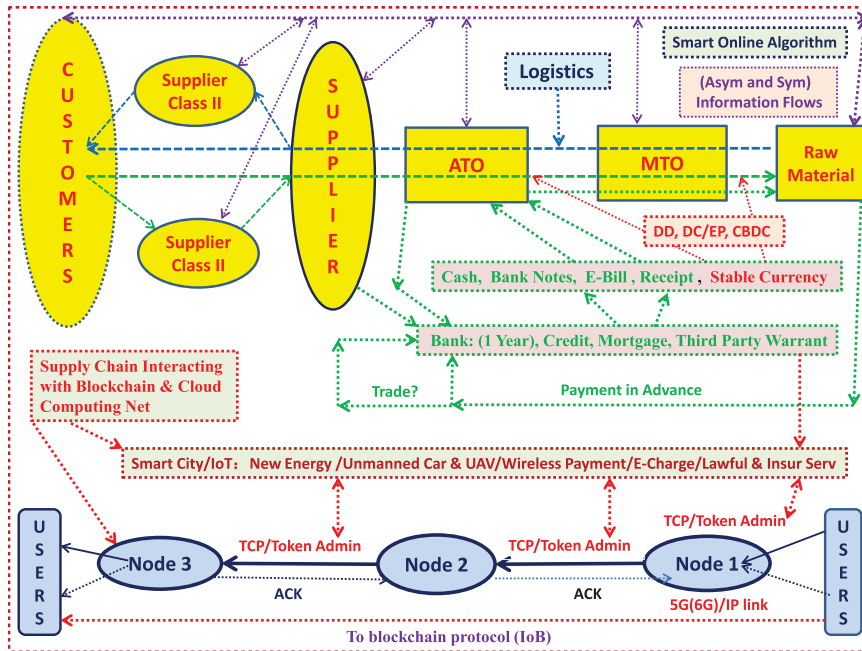


Figure 3. A generalized supply chain with finance transactions via stable digital currencies. In this figure, ATO means assemble to order, MTO means make to order, DD means digital dollar, DC/EP means (China Central Bank) digital currency/electronic payment, CBDC means (European) Central Bank digital currency, transmission control protocol means transmission control protocol, IoB means Internet of Blockchains, Asym and sym mean asymmetry and symmetry, respectively.

Since dynamic resource pricing is our major concern, we give some discussions about the concept of stable digital currency and its applications. More precisely, stable digital currency is a digital token used in digital informational and data network systems. It can be traced back to the optimal pricing of bits (or ports) in telecommunication managements and admission controls through token buffers in communication networks around the mid and late 90s by Bell Labs’ researchers (see e.g., the related discussions in Dai [6], Elwalid and Mitra [15]). Along this line, Nakamoto [20] extended the concept of bit (or port) to the bitcoin in the year of 2008 and Buterin [4] further enhanced this concept to Ethereum in the year of 2013. Note that, for both the bitcoin and the Ethereum, they are still not real stable digital currencies. However, during this evolvement, Dai (see e.g., Maker [18]) made his effort to endow the Ethereum with real value and invented DaiCoin. Since then, this concept and lawful implementations are becoming more and more popular with the emergence of US digital dollar (DD), (China Central Bank) digital currency/electronic payment (DC/EP), and (European) Central Bank digital currency (CBDC). The latest application of dynamic resource pricing can be found in a metaverse system (see e.g., Dai [11]).

To show the importance of stable digital currency, an example based on a supply chain finance service system is displayed in Figure 3. The business model presented in this example can be considered as a generalized online digital payment system with service lead time involvements and consists of 4 typical service stages to make goods eventually delivered to customers: raw material procurements, make to order (MTO), assemble to order (ATO), and agent sales (see e.g., the upper-half of Figure 3 where agents are further classified into two levels of suppliers). Usually, during the procurement and service stages, the cash cannot be paid until the delivery of procured products. Thus, the bank notes, e-bill, receipt, etc. through credit, mortgage, and the third party warrant as shown in the middle of Figure 3 are widely used in real-world practice. To improve the efficiency and security of this type of payments, the stable

digital currency such as US DD, (China) DC/EP, and (European) CBDC as designed in Figure 3 is a suitable choice. Furthermore, as shown in the upper-right corner of Figure 3, data information among companies can be asymmetric or symmetric. Frequently, they are not exchangeable. Thus, our policy can be integrated into this system to develop a smart online algorithm in terms of dynamic resource pricing to solve this problem. From the lower-half design in Figure 3, we can see that our supply chain system can be mapped into and interact with an information system through wireless 5G/6G network or wireline IP network (and even future IoB). Then, many online payments and transactions with lawful services can be handled as shown in the middle of Figure 3.

The remainder of this paper is organized as follows. In Section 2, we formulate our system model for dynamic resource pricing via stable digital currency. In Section 3, we present our main theorem based on a 3-stage (i.e., users-selection, dynamic pricing, and resource-competition scheduling) policy. A corresponding general CNN based algorithm flow chart is also presented in this section. In Section 4, we present two illustrative policy examples. In Section 5, we conduct simulation case studies to show the effectiveness of our policy. In Section 6, we theoretically prove our main theorem. In Section 7, we give the conclusion of this paper.

2. System model

In this section, we present our service model with dynamic resource pricing capability. It owns V number of service pools associated with a set of positive integers $\mathcal{V} \equiv \{1, \dots, V\}$ and owns J number of queues for J -parallel users corresponding to a set of positive integers $\mathcal{J} \equiv \{1, \dots, J\}$. Furthermore, we assume that the buffer storage in each queue is nonnegative. Each pool owns J_v number of flexible parallel-servers with v belonging to a positive integer set \mathcal{V} . Let the prime denote the transpose of a vector or a matrix. Then, associated with the queues, there is a J -dimensional arrival process $A = \{A(t) = (A_1(t), \dots, A_J(t))', t \geq 0\}$ and it is called a data packet arrival process. In this situation, $A_j(t)$ for each $j \in \mathcal{J}, t \geq 0$, and some positive integer $n \in \{1, 2, \dots\}$ is the number of data packets that arrive at the j th queue during time interval $(0, t]$. In addition, in a real world service system such as a banking service or a supply chain system, the associated input ethereum/cash flows and supply/demand processes can be digitalized and mapped into the data packet based framework. The size of a data packet is a random number $\zeta \in \{1, 2, \dots\}$. Our system is assumed under an external random environment driven by a stationary FS-CTMC $\alpha = \{\alpha(t), t \in [0, \infty)\}$ with a finite state space $\mathcal{K} \equiv \{1, \dots, K\}$, whose generator matrix is given by $G = (g_{il})$ with $i, l \in \mathcal{K}$, and

$$g_{il} = \begin{cases} -\gamma(i) & \text{if } i = l, \\ \gamma(i)q_{il} & \text{if } i \neq l, \end{cases} \tag{2.1}$$

where $\gamma(i)$ is the holding rate for the continuous time chain staying in a state $i \in \mathcal{K}$ and $Q = (q_{il})$ is the corresponding transition matrix of its embedded discrete-time Markov chain (see e.g., Resnick [24]). Moreover, define τ_n for each nonnegative integer $n \in \{0, 1, \dots\}$ by:

$$\tau_0 \equiv 0, \quad \tau_n \equiv \inf\{t > \tau_{n-1} : \alpha(t) \neq \alpha(t^-)\}. \tag{2.2}$$

Note that, the external random environment may be caused by different factors (see e.g., the explanations in Choudhury *et al.* [5], Dai [6], Wang and Moayeri [27]). Here, in our blockchained case, the FS-CTMC $\alpha(\cdot)$ may be considered as a history-data dependent system randomly evolving parameter for a targeted stationary random variable $\alpha(\infty)$, which can be generated by (or approximated through) a conditional mean defined process, i.e.,

$$\alpha(t) = E\left[\alpha(\infty) \middle| \mathcal{F}_t\right], \tag{2.3}$$

where $\{\mathcal{F}_t, t \in [0, \infty)\}$ is a filtration generated by blockchain history information. For example, when the blockchain database read and write (R/W) times are exponentially distributed, the martingale representation theorem for a jump-diffusion process (see e.g., Applebaum [1]) can be applied to (2.3) to generate the required FS-CTMC assumption. Then, we can model the arrival process $A_j(\cdot)$ for each positive integer $j \in \mathcal{J}$ as a big data flow stream through a TSRRP as in Dai [8]. More precisely, the process $A_j(\tau_n + \cdot)$ for each $n \in \{0, 1, \dots\}$ is a counting process corresponding to a (conditional) delayed renewal reward process with arrival rate $\lambda_j(\alpha(\tau_n))$ and mean reward $m_j(\alpha(\tau_n))$ associated with finite squared coefficients of variations $\alpha_j^2(\alpha(\tau_n))$ and $\zeta_j^2(\alpha(\tau_n))$ during time interval $[\tau_n, \tau_{n+1})$.

Now, we let $\{u_j(k), k = 1, 2, \dots\}$ be the sequence of times between the arrivals of the $(k - 1)$ th and the k th reward batches of packets at the j th queue. The associated batch reward is given by $w_j(k)$ and all the data packets arrived with it are indexed in certain successive order. Therefore, we can present the renewal counting process corresponding to the inter-arrival time sequence $\{u_j(k), k = 1, 2, \dots\}$ for each $j \in \mathcal{J}$ as follows,

$$N_j(t) = \sup \left\{ n \geq 0 : \sum_{k=1}^n u_j(k) \leq t \right\}. \tag{2.4}$$

Thus, we can restate the definition of an TSRRP $A_j(\cdot)$ quantitatively through the expression,

$$A_j(t) = \sum_{k=1}^{N_j(t)} w_j(k). \tag{2.5}$$

Each data packet will first get service in the system and then leave it. The service is managed by a blockchain. In this blockchain, the service for a data packet is composed of two parts: security checking and policy computation (or real data payload transmission). After completing the service, the security information and the policy (or the transmission result) will be stored and copied to all the participating partner nodes for storage and in the meanwhile to produce nonce values and private keys. Moreover, we denote $\{v_j(k), k = 1, 2, \dots\}$ to be the sequence of successive arrived packet lengths at queue j , which is assumed to be a sequence of strictly positive i.i.d. random variables with average packet length $1/\mu_j \in (0, \infty)$ and squared coefficient of variation $\beta_j^2 \in (0, \infty)$. In addition, we suppose that all the inter-arrival and service time processes are mutually (conditionally) independent when the environmental state is fixed. Associated with each $j \in \mathcal{J}$ and each nonnegative constant h , we employ $S_j(\cdot)$ to denote the renewal counting process corresponding to $\{v_j(k), k = 1, 2, \dots\}$. In other words,

$$S_j(h) = \sup \left\{ n \geq 0 : \sum_{k=1}^n v_j(k) \leq h \right\}. \tag{2.6}$$

Define $Q_j(t)$ to be the j th queue length with $j \in \mathcal{J}$ at each time $t \in [0, \infty)$ and $D_j(t)$ to be the number of packet departures from the j th queue in $(0, t]$. Therefore, the queueing dynamics governing the evolving of the internal data flow in and out within our unified service platform can be modeled by:

$$Q_j(t) = Q_j(0) + A_j(t) - D_j(t), \tag{2.7}$$

where each queue is assumed to have an infinite storage capacity to buffer data packets (jobs) arrived from a given user.

Note that, in a DaiCoin and blockchain based mortgage system (see e.g., Maker [18]), $Q_j(t)$ is the number of Ethereums available at time t . In this case, we need to dynamically determine how many Dais should be loaned to customer j for each Ethereum at time t according to the value of $Q(t)$. Similarly, in a banking system, $Q_j(t)$ can be the number of loan demands waiting at time t . In this case, we need

to determine what is the loan interest rate at time t according to the value of $Q(t)$. Furthermore, in communication and cloud-computing based service systems, we need to price the bit service ratio at time t according to the value of $Q(t)$. In all, we need to dynamically price our service in a real-world system according to the evolving of $Q(t)$ with the evolution of time t . For convenience, we will use the unified terminology “price $P_j(t)$ ” to denote the price (the number of Dais or interest ratio) associated with $Q_j(t)$ and $\alpha(t)$ at time t . In economics, there are different pricing functions with respect to $Q(t)$ and $\alpha(t)$ (see e.g., Dai and Jiang [13]). Here, we assume that $P_j(t)$ is a positive function in terms of $Q_j(t)$ and $\alpha(t)$, i.e.,

$$P_j(t) = f_j(Q_j(t), \alpha(t)). \tag{2.8}$$

Note that, $P_j(t)$ is a specific value at time t for given values of $Q_j(t)$ and $\alpha(t)$. In general, $P_j(t)$ is a random variable at time t since both $Q_j(t)$ and $\alpha(t)$ are random variables at time t . From the expression of (2.8), we can see that the price also depends on the random environment movement (e.g., the season’s movement). In addition, we suppose that $f_j(\cdot, \cdot)$ in (2.8) is Lipschitz continuous with respect to $Q_j(t)$. Then, we can introduce a utility (or a hash) function with respect to the valued queue length $P_j(t)Q_j(t)$ for user $j \in \mathcal{J}$ at each service pool $v \in \mathcal{V}$ as follows,

$$U_{vj}(P(t)Q(t), \Lambda(t)) \text{ with } P(t)Q(t) = (P_1(t)Q_1(t), \dots, P_J(t)Q_J(t)), \tag{2.9}$$

where $P(t) = (P_1(t), \dots, P_J(t))$ and $\Lambda(t) = (\Lambda_1(t), \dots, \Lambda_J(t))$. Moreover, $\Lambda_j(t)$ for each $t \in [0, \infty]$ and $j \in \mathcal{J}$ is the summation of all service rates allocated to the j th user at time t from all possible pools and servers. Here, we remark that, $\Lambda_j(t)$ may be given in a feedback control form and it depends on the current price $P(t)$, the current queue length $Q(t)$, and the system state $\alpha(t)$ at a given time t . In other words, we have that $\Lambda_j(t) = \Lambda_j(P(t)Q(t), \alpha(t))$. Furthermore, we note that, the upper case $\Lambda_j(t)$ used here denotes service capacity allocation process. It is not directly related to the lower case λ_j as used in (6.3) of Subsubsection 6.1.3, which denotes the nominal arrival rate for user j .

Now, we define $W(t)$ and $W_j(t)$ to be the (expected) total workload in the system at time t and the one associated with user j at time t , to wit,

$$W(t) = \sum_{j=1}^J W_j(t), \quad W_j(t) = \frac{Q_j(t)}{\mu_j}. \tag{2.10}$$

In the following study, we will use $W(t)$ and $Q(t)$ as performance measures, $f = (f_1, \dots, f_J)$ in (2.8) as pricing function, and $\{U_{vj}, j \in \mathcal{J}, v \in \mathcal{V}\}$ in (2.9) as utility (or hash) functions. Based on these measures and functions, we can propose a joint dynamical pricing and rate scheduling policy (P, Λ) with users-selection at each time point for different service pools and servers to all the users. Under the policy, the total workload $W(t)$ and its corresponding total cost are minimized, where the total cost is a summation of costs serving all users in certain sense. For an exact definition of the total cost, it is given in (3.22) though the so-called dual-costs. Furthermore, we assume that the available resources from different pools and servers can be flexibly allocated and shared between the system and users, i.e., the system operates under a concurrent resource occupancy service regime. Based on these facts, we can define $T_j(t)$ to be the cumulative amount of service given to the j th queue up to time t , i.e.,

$$T_j(t) = \int_0^t \Lambda_j(P(s)Q(s), \alpha(s)) ds. \tag{2.11}$$

Hence, if we let $S_j(t)$ be the total number of jobs (packets) that finishes service in the system by time t , we know that $D_j(t) = S_j(T_j(t))$.

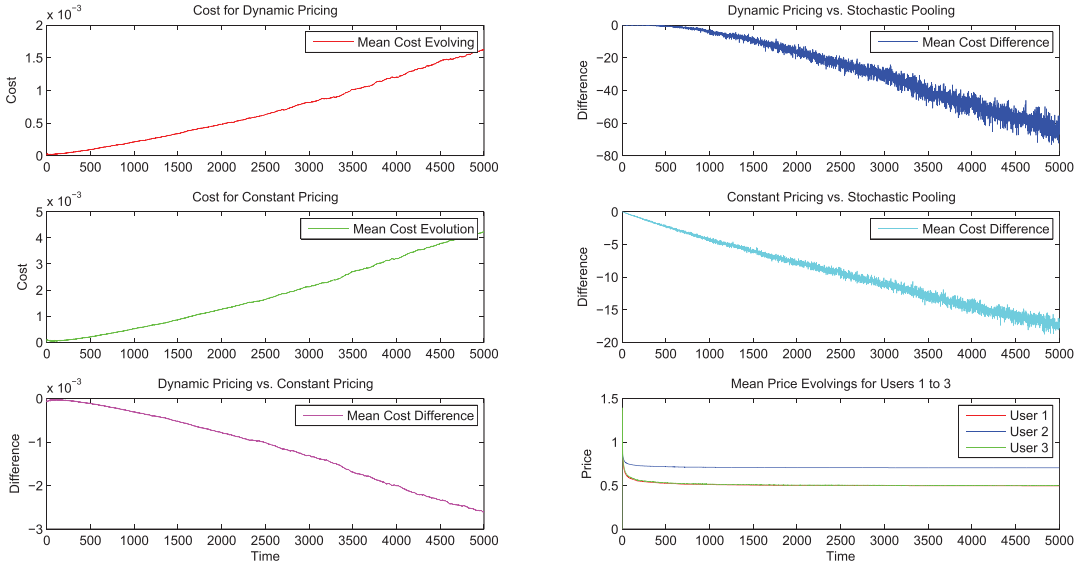


Figure 4. In this simulation, the number of simulation iterative times is $N = 6,000$, the simulation time interval is $[0, T]$ with $T=20$, which is further divided into $n = 5,000$ subintervals as explained in Subsection 5. Other values of simulation parameters introduced in Definition 3.1 and Subsubsection 4 are as follows: $initialprice1 = 2.25$, $initialprice2 = 1.5$, $initialprice3 = 2.25$, $upperboundprice1 = 4$, $upperboundprice2 = 2$, $upperboundprice3 = 4$, $lowerboundprice1 = 0.49$, $lowerboundprice2 = 0.7$, $lowerboundprice3 = 0.49$, $queupolicylowerbound1 = 0$, $queupolicylowerbound2 = 0$, $queupolicylowerbound3 = 0$, $\lambda_1 = 10/3$, $\lambda_2 = 5$, $\lambda_3 = 10/3$, $m_1 = 3$, $m_2 = 1$, $m_3 = 3$, $\mu_1 = 1/10$, $\mu_2 = 1/20$, $\mu_3 = 1/10$, $\alpha_1 = \sqrt{10/3}$, $\alpha_2 = \sqrt{20}$, $\alpha_3 = \sqrt{10/3}$, $\beta_1 = \sqrt{10}$, $\beta_2 = \sqrt{20}$, $\beta_3 = \sqrt{10}$, $\zeta_1 = 1$, $\zeta_2 = \sqrt{2}$, $\zeta_3 = 1$, $\rho_1 = \rho_2 = \rho_3 = 1,000$, $\theta_1 = -1$, $\theta_2 = -1.2$, $\theta_3 = -1$.

3. Main theorem

TSRRPs can effectively model big data arrival streams. However, it is difficult to directly conduct the analysis of the associated physical queueing model in (2.7) or its related physical workload model in (2.10) due to the non-Markovian characteristics of TSRRPs. Thus, in this paper, we will develop a scheme by applying the well-known heavy traffic approximation and modeling technique to establish the RDRS model corresponding to our newly designed game-competition based dynamic resource pricing and scheduling policy by considering our queueing system under the asymptotic regime, where it is heavily loaded (load balanced), i.e., under the so-called heavy traffic condition. Furthermore, we will prove the correctness of RDRS modeling via diffusion approximation while we will also show the effectiveness of the identified model for our newly proposed pricing and scheduling policy by presenting simulation case studies. The corresponding simulation results are displayed in Figures 4–5 (as follows) and Figure 11 and their interpretations are presented in Subsection 5.

3.1. RDRS model

In this subsection, we first present the basic idea of our main claim in terms of our RDRS modeling under a smart contract policy. Second, for convenience, we introduce the definition of RDRS model. More precisely, for each $t \geq 0$ and $j \in \mathcal{J}$, we introduce two sequences of diffusion-scaled processes: $\hat{Q}^r(\cdot)$ and $\hat{W}^r(\cdot)$ by:

$$\hat{Q}_j^r(t) \equiv \frac{Q_j^r(r^2t)}{r}, \quad \hat{W}^r(t) \equiv \frac{W^r(r^2t)}{r}, \tag{3.1}$$

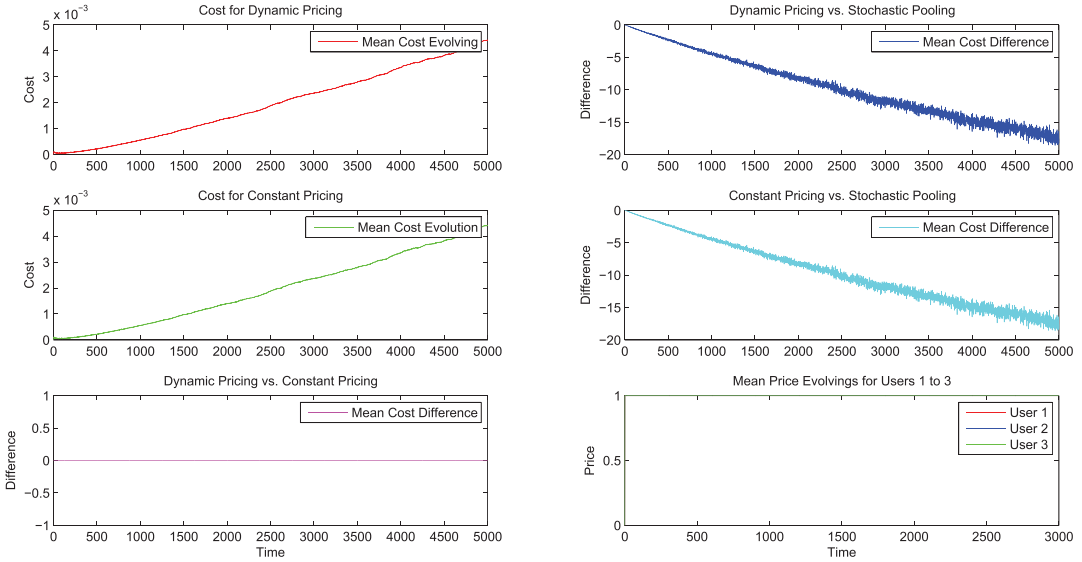


Figure 5. In this simulation, the number of simulation iterative times is $N = 6,000$, the simulation time interval is $[0, T]$ with $T = 20$, which is further divided into $n = 5,000$ subintervals as explained in Subsection 5. Other values of simulation parameters introduced in Definition 3.1 and Subsubsection 4 are as follows: $initialprice1 = 1, initialprice2 = 1, initialprice3 = 1, upperboundprice1 = 1, upperboundprice2 = 1, upperboundprice3 = 1, lowerboundprice1 = 1, lowerboundprice2 = 1, lowerboundprice3 = 1, queuopolycylowerbound1 = 0, queuopolycylowerbound2 = 0, queuopolycylowerbound3 = 0, \lambda_1 = 10/3, \lambda_2 = 5, \lambda_3 = 10/3, m_1 = 3, m_2 = 1, m_3 = 3, \mu_1 = 1/10, \mu_2 = 1/20, \mu_3 = 1/10, \alpha_1 = \sqrt{10/3}, \alpha_2 = \sqrt{20}, \alpha_3 = \sqrt{10/3}, \beta_1 = \sqrt{10}, \beta_2 = \sqrt{20}, \beta_3 = \sqrt{10}, \zeta_1 = 1, \zeta_2 = \sqrt{2}, \zeta_3 = 1, \rho_1 = \rho_2 = \rho_3 = 1,000, \theta_1 = -1, \theta_2 = -1.2, \theta_3 = -1.$

where $\{r, r \in \mathcal{R}\}$ is supposed to be a strictly increasing sequence of positive real numbers and tends to infinity. Then, our main claim can be presented as follows.

The sequence of 2-tuple scaled processes in (3.1) corresponding to a game-competition based dynamic resource pricing and scheduling policy with users’ selection, which is designed in the subsequent subsection, converges jointly in distribution. More precisely, under the heavy traffic condition described in Section 6, we have that:

$$(\hat{Q}^r(\cdot), \hat{W}^r(\cdot)) \Rightarrow (\hat{Q}(\cdot), \hat{W}(\cdot)) \text{ along } r \in \mathcal{R}, \tag{3.2}$$

where $\hat{W}(\cdot)$ is presented by an RDRS model and $\hat{Q}(\cdot)$ is an asymptotic queue policy process with dynamic pricing globally over $[0, \infty)$ through a saddle point to zero-sum game-competition problem and a Pareto minimal-dual-cost Nash equilibrium point to a non-zero-sum game-competition problem.

Definition 3.1. A u -dimensional stochastic process $\hat{Z}(\cdot)$ with $u \in \mathcal{J}$ is claimed as an RDRS with oblique reflection if it can be uniquely represented as:

$$\begin{cases} \hat{Z}(t) &= \hat{X}(t) + \int_0^t R(\alpha(s), s) d\hat{Y}(s) \geq 0, \\ d\hat{X}(t) &= b(\alpha(t), t)dt + \sigma^E(t)d\hat{H}^E(t) + \sigma^S(t)d\hat{H}^S(t). \end{cases} \tag{3.3}$$

Furthermore, $b(\alpha(t), t) = (b_1(\alpha(t), t), \dots, b_u(\alpha(t), t))'$ is a u -dimensional vector, $\sigma^E(t)$ and $\sigma^S(t)$ are $u \times J$ matrices, $R(\alpha(t), t)$ with $t \in \mathbb{R}_+$ is a $u \times u$ matrix, and $(\hat{Z}(\cdot), \hat{Y}(\cdot))$ is a coupled almost surely

continuous solution of (3.3) with the following properties for each $j \in \{1, \dots, u\}$,

$$\left\{ \begin{array}{l} \hat{Y}_j(0) = 0; \\ \text{Each component } \hat{Y}_j(\cdot) \text{ of } \hat{Y}(\cdot) = (\hat{Y}_1(\cdot), \dots, \hat{Y}_u(\cdot))' \text{ is non-decreasing;} \\ \text{Each component } \hat{Y}_j(\cdot) \text{ can increase only at a time } t \in [0, \infty) \text{ that } \hat{Z}_j(t) = 0, \text{ i.e.,} \\ \int_0^\infty \hat{Z}_j(t) d\hat{Y}_j(t) = 0. \end{array} \right.$$

In addition, a solution to the RDRS in (3.3) is called a strong solution if it is in the pathwise sense and is called a weak solution if it is in the sense of distribution.

In terms of the well-posedness of an RDRS, readers are referred to a general discussion in Dai [7]. Furthermore, in Definition 3.1, the stochastic processes $B^E(\cdot)$ and $B^S(\cdot)$ are, respectively, two J -dimensional standard Brownian motions, which are independent each other. For each state $i \in \mathcal{K}$ and a time $t \in [0, \infty)$, the nominal arrival rate vector $\lambda(i)$, the mean reward vector $m(i)$, the nominal throughput vector $\rho(i)$, and a constant parameter vector $\theta(i)$ are given as follows,

$$\left\{ \begin{array}{l} \lambda(i) = (\lambda_1(i), \dots, \lambda_J(i))', \\ m(i) = (m_1(i), \dots, m_J(i))', \\ \rho(i) = (\rho_1(i), \dots, \rho_J(i))', \\ \theta(i) = (\theta_1(i), \dots, \theta_J(i))'. \end{array} \right. \tag{3.4}$$

The covariance matrices are given by:

$$\left\{ \begin{array}{l} \Gamma^E(i) = \left(\Gamma_{kl}^E(i) \right)_{J \times J} \\ \equiv \text{diag} (\lambda_1(i)m_1^2(i)\zeta_1^2(i) + \lambda_1(i)m_1^2(i)\alpha_1^2(i), \\ \dots, \lambda_J(i)m_J^2(i)\zeta_J^2(i) + \lambda_J(i)m_J^2(i)\alpha_J^2(i)), \\ \Gamma^S(i) = \left(\Gamma_{kl}^S(i) \right)_{J \times J} \\ \equiv \text{diag} (\lambda_1(i)m_1(i)\beta_1^2, \dots, \lambda_J(i)m_J(i)\beta_J^2). \end{array} \right. \tag{3.5}$$

The Itô's integrals with respect to the Brownian motions are defined as:

$$\left\{ \begin{array}{l} \hat{H}^e(t) = \left(\hat{H}_1^e(t)', \dots, \hat{H}_J^e(t)' \right)' \text{ with } e \in \{E, S\}, \\ \hat{H}_j^e(t) = \int_0^t \sqrt{\Gamma_{jj}^e(\alpha(s))} dB_j^e(s). \end{array} \right. \tag{3.6}$$

3.2. A 3-stage users-selection and dynamic pricing/rate scheduling policy

In this subsection, we design a 3-stage users-selection, dynamic pricing, and rate scheduling policy through a 2-stage game-theoretic problem consisting of both zero-sum and non-zero-sum game-competitions myopically at each time point for the purpose as stated in the introduction of the paper.

3.2.1. General service capacity region

In our system, the jobs in the j th queue for each $j \in \mathcal{J}$ may be served at the same time by a random but at most $V_j (\leq V)$ number of service pools corresponding to selected utility (hash) functions at a particular time point. With this simultaneous service mechanism, the total service rate for the j th queue at the time

point is the summation of the rates from all the pools possibly to serve the j th queue. More precisely, we index these pools by a subset $\mathcal{V}(j)$ of the set \mathcal{V} as follows,

$$\mathcal{V}(j) \equiv \{v_{1j}, \dots, v_{Vj}\} \subseteq \mathcal{V}, \tag{3.7}$$

where, v_{lj} with $l \in \{1, \dots, V_j\}$ denotes the v_{lj} th pool in $\mathcal{V}(j)$. In the same way, a pool denoted by $v \in \mathcal{V}$ can possibly serve at most J_v number of job classes represented by a subset $\mathcal{J}(v)$ of the set \mathcal{J} , i.e.,

$$\mathcal{J}(v) \equiv \{j_{v1}, \dots, j_{vJ_v}\} \subseteq \mathcal{J}, \tag{3.8}$$

where j_{vl} with $l \in \{1, \dots, J_v\}$ indexes the j_{vl} th job class in $\mathcal{J}(v)$. To be more illustrative, let L be a $J \times V$ constituent matrix such that:

$$L_{jv} = \begin{cases} 1 & \text{if user } j \text{ can be served by pool } v, \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\mathcal{V}(j)$ consists of all the non-zero components of the j th row of L while $\mathcal{J}(v)$ consists of all the non-zero components of the v th column of L . Furthermore, in each pool v , there are J_v number of flexible parallel-servers with rate allocation vector:

$$c_v(t) = (c_{j_{v1}}(t), \dots, c_{j_{vJ_v}}(t))', \tag{3.9}$$

where $c_{j_{vl}}(t)$ with $l \in \{1, \dots, J_v\}$ is the assigned service rate to the j_{vl} th user at pool v and time t . Similarly, corresponding to the $l \in \{1, \dots, J_v\}$, we will also denote the rate $c_{j_{vl}}(t)$ by $c_{vj}(t)$ for an index $j \in \mathcal{J}(v)$.

Note that, the vector in (3.9) takes values in a capacity region $\mathcal{R}_v(\alpha(t))$ driven by the FS-CTMC $\alpha = \{\alpha(t), t \in [0, \infty)\}$. For each given $i \in \mathcal{K}$ and $v \in \mathcal{V}$, the set $\mathcal{R}_v(i)$ is a convex region containing the origin and has $L_v (> J_v)$ boundary pieces (see e.g., the upper-left graph in Figure 2). In this region, every point is defined according to the associated users, i.e., $x = (x_{j_{v1}}, \dots, x_{j_{vJ_v}})$. On the boundary of $\mathcal{R}_v(i)$ for each $i \in \mathcal{K}$, J_v of them are $(J_v - 1)$ -dimensional linear facets along the coordinate axes. The other ones denoted by $\mathcal{O}_v(i)$ are located in the interior of $\mathcal{R}_+^{J_v}$. It is called the *capacity surface* of $\mathcal{R}_v(i)$ and it has $B_v = L_v - J_v (> 0)$ linear or smooth curved facets $h_{vk}(c_v, i)$ on $\mathcal{R}_+^{J_v}$ for $k \in \mathcal{U}_v \equiv \{1, 2, \dots, B_v\}$, i.e.,

$$\mathcal{R}_v(i) \equiv \{c_v \in \mathcal{R}_+^{J_v} : h_{vk}(c_v, i) \leq 0, k \in \mathcal{U}_v\}. \tag{3.10}$$

Furthermore, if we define $C_{U_v}(i)$ to be the sum capacity upper bound for $\mathcal{R}_v(i)$, the facet in the center of $\mathcal{O}_v(i)$ is linear and is assumed to be a non-degenerate $(J_v - 1)$ -dimensional region. More precisely, it can be represented by

$$h_{vk_{U_v}}(c_v, i) = \sum_{j \in \mathcal{J}(v)} c_j - C_{U_v}(i), \tag{3.11}$$

where $k_{U_v} \in \mathcal{U}_v$ is the index corresponding to $C_{U_v}(i)$. In addition, we suppose that any one of the J_v linear facets along the coordinate axes forms an $(J_v - 1)$ -user capacity region associated with a particular group of $J_v - 1$ users if the queue corresponding to the other user is empty. In the same manner, we can provide an interpretation for the $(J_v - l)$ -user capacity region for each $l \in \{2, \dots, J_v - 1\}$.

Concerning the allocation of the service resources over the capacity regions to different users, we adopt the so-called head of line service discipline. Equivalently, the service goes to the packet at the head of the line for a serving queue where packets are stored in the order of their arrivals. The service

rates are determined by a utility (or hash) function of the environmental state, the price for each user, and the number of packets in each of the queues. More precisely, for each state $i \in \mathcal{K}$, a price vector $p = (p_1, \dots, p_J)$, and a queue length vector $q = (q_1, \dots, q_J)'$, we define $\Lambda_j(pq, i)$ with $j \in \mathcal{J}$ to be the rate vector (in qubits/ps) of serving the j th queue at all its possible service pools, i.e.,

$$\Lambda_j(pq, i) = c_j^{\mathcal{Q}(pq)}(i) = (c_{v_{lj}}^{\mathcal{Q}(pq)}(i), \dots, c_{v_{vj}}^{\mathcal{Q}(pq)}(i)), \tag{3.12}$$

where,

$$\mathcal{Q}(pq) \equiv \{j \in \mathcal{J}, q_j = 0\}. \tag{3.13}$$

Furthermore, let $\Lambda_{v \cdot}(pq, i)$ for each $v \in \mathcal{V}$ be the rate vector for all the users possibly served at service pool v , i.e.,

$$\Lambda_{v \cdot}(pq, i) = c_{v \cdot}^{\mathcal{Q}(pq)}(i) = (c_{j_{v1}}^{\mathcal{Q}(pq)}(i), \dots, c_{j_{vV_j}}^{\mathcal{Q}(pq)}(i)). \tag{3.14}$$

Thus, $c_{v_{lj}}^{\mathcal{Q}(pq)}(i) = c_j^{\mathcal{Q}(pq)}(i)$ if the pool index $v_{lj} \in \mathcal{V}(j)$ for an integer $l \in \{1, \dots, V_j\}$ with $j \in \mathcal{J}$ while the total rate used in (2.11) can be represented by:

$$\Lambda_j(P(s)Q(s), \alpha(s)) = \sum_{v \in \mathcal{V}(j)} c_{vj}^{\mathcal{Q}(P(s)Q(s))}(\alpha(s)). \tag{3.15}$$

In the end, we impose the convention that an empty queue should not be served. Then, for each $v \in \mathcal{V}$ and $\mathcal{Q} \subseteq \mathcal{J}$ (e.g., a set as given by (3.13)), we can define:

$$c_{j_{vl}}^{\mathcal{Q}}(i) \equiv \begin{cases} = 0 & \text{if } j_{vl} \in \mathcal{Q} \text{ with } l \in \{1, \dots, J_v\}, \\ > 0 & \text{if } j_{vl} \notin \mathcal{Q} \text{ with } l \in \{1, \dots, J_v\}, \end{cases} \tag{3.16}$$

$$c_{v_j}^{\mathcal{Q}}(i) \equiv c_{j_{vl}}^{\mathcal{Q}}(i) \text{ for some } j \in \mathcal{J}(v) \text{ corresponding to each } l \in \{1, \dots, J_v\}, \tag{3.17}$$

$$F_{\mathcal{Q}}^v(i) \equiv \left\{ x \in \mathcal{R}_v(i) : x_{j_{vl}} = 0 \text{ for all } j_{vl} \in \mathcal{Q} \text{ with } l \in \{1, \dots, J_v\} \right\}. \tag{3.18}$$

Therefore, for all \mathcal{Q} such that $\emptyset \subsetneq \mathcal{Q} \subseteq \mathcal{J}(v)$ corresponding to each $v \in \mathcal{V}$, if $c_{v \cdot}^{\mathcal{Q}}(i)$ is on the boundaries of the capacity region $\mathcal{R}_v(i)$, we have the following observation that:

$$\begin{cases} \sum_{j \in \mathcal{J}(v)} c_{vj}^{\emptyset}(i) & \geq \sum_{j \in \mathcal{J}(v)} c_{vj}^{\mathcal{Q}}(i), \\ \sum_{j \in \mathcal{J}(v) \setminus \mathcal{Q}} c_{vj}^{\emptyset}(i) & \leq \sum_{j \in \mathcal{J}(v) \setminus \mathcal{Q}} c_{vj}^{\mathcal{Q}}(i), \end{cases} \tag{3.19}$$

where $c_{v \cdot}^{\emptyset}(i) \in \mathcal{O}_v(i)$ and \emptyset denotes the empty set. Typical examples of our capacity region are referred to the upper-left graph in Figure 2 for more details.

3.2.2. A dynamic pricing and scheduling policy with users-selection

For our purpose, we classify all the users into two types. More precisely, we first need to smartly choose the users to be served. In other words, at each time point and for each pool v , we intelligently select a set $\mathcal{M}(i, v) \equiv \{j_{v1}(i), \dots, j_{vM_v}(i)\}$ of users to get into services with $j_{vl} \in \mathcal{J}$ and $l \in \{1, \dots, M_v\}$ for a given positive integer number $M_v \leq J_v$. Among these chosen users, we need to conduct the dynamic pricing while realize optimal and fair resource allocation. Therefore, we design a strategy by mixing a

saddle point and a static Pareto maximal-utility Nash equilibrium policy myopically at each time point t to a mixed zero-sum and non-zero-sum game problem for each state $i \in \mathcal{K}$ and a given valued queue length vector $pq = (p_1q_1, \dots, p_Jq_J)'$. Here we note that $p = (p_1, \dots, p_J)'$ is a given price vector and $q = (q_1, \dots, q_J)'$ is a given queue length vector such that $p_j = f_j(q_j, i)$ as in (2.8) for each $j \in \mathcal{J}$ and $i \in \mathcal{K}$. The saddle point corresponds the users' selection while the Pareto optimality represents the full utilization of resources in the whole game system and the Nash equilibrium represents the fairness to all the chosen users. More exactly, in this game, there are J users (players) associated with the J queues. Each of them has his own utility function $U_{vj}(p_jq_j, c_{vj})$ with $j \in \mathcal{J}(v)$ and $v \in \mathcal{V}(j)$. This utility function is a generalization of the existing one in Dai [6, 8, 9], Ye and Yao [28], and references therein. Examples of such a utility function can be the so-called proportionally fair allocation, minimal potential delay allocation, and (β, α) -proportionally fair allocation (see e.g., Ye and Yao [28]), which are widely used in internet protocols and communication systems. Every chosen user selects a policy to maximize his own utility function at each service pool v while the summation of all the users' utility functions and the summation of the utility functions associated with the chosen users are also maximized. To wit, we can formulate a generalized users-selection, pricing, and resource-scheduling game problem by extending the ones in Examples 4.1–4.3 as follows,

$$\begin{cases} \max_{c_v \in F_{\mathcal{Q}}^v(i)} U_{00}(pq, c) & = U_{00}(pq, c^*(i)), \\ \max_{c_v \in F_{\mathcal{Q}}^v(i)} U_{0j}(pq, c) & = U_{0j}(pq, c^*(i)), \quad j \in \mathcal{M}(i, v) \cap (\mathcal{J}(v) \setminus \mathcal{Q}(q)), \\ \max_{c_v \in F_{\mathcal{Q}}^v(i)} (-U_{0j}(pq, c)) & = -U_{0j}(pq, c^*(i)), \quad j \in (\mathcal{J}(v) \setminus \mathcal{Q}(q)) \setminus \mathcal{M}(i, v) \end{cases} \quad (3.20)$$

while we have that:

$$\begin{cases} \max_{c_v \in F_{\mathcal{Q}}^v(i), j \in \mathcal{M}(i, v) \cap (\mathcal{J}(v) \setminus \mathcal{Q}(q))} U_{vj}(pq, c) & = U_{vj}(pq, c^*(i)), \\ \max_{c_v \in F_{\mathcal{Q}}^v(i), j \in (\mathcal{J}(v) \setminus \mathcal{Q}(q)) \setminus \mathcal{M}(i, v)} (-U_{vj}(pq, c)) & = -U_{vj}(pq, c^*(i)). \end{cases} \quad (3.21)$$

Note that, the rate vector c in (3.20)-(3.21) is given by:

$$c = ((c_{j11}, \dots, c_{j1J_1}), \dots, (c_{jv1}, \dots, c_{jvJ_v})),$$

and the utility functions used in (3.20)-(3.21) are defined by:

$$\begin{cases} U_{00}(pq, c) & = \sum_{j \in \mathcal{J}(v) \setminus \mathcal{Q}(q)} \sum_{v \in \mathcal{V}(j)} U_{vj}(p_jq_j, c_{vj}), \\ U_{0j}(pq, c) & = \sum_{v \in \mathcal{V}(j)} U_{vj}(p_jq_j, c_{vj}) & \text{for each } j \in \mathcal{J}(v) \setminus \mathcal{Q}(q), \\ U_{vj}(pq, c) & = U_{vj}(p_jq_j, c_{vj}) & \text{for each } j \in \mathcal{J}(v) \setminus \mathcal{Q}(q) \text{ and } v \in \mathcal{V}(j). \end{cases}$$

Then, by extending the concepts of Nash equilibrium point, saddle point, and Pareto optimality in Dai [8, 9], Marchi [19], Nash [21] and Rosen [25], we have the following definition concerning a utility based 2-stage game-theoretic policy via a saddle point to a zero-sum game problem and a static Pareto maximal-utility Nash equilibrium point to a non-zero-sum game problem myopically at each particular time point.

Definition 3.2. For each state $i \in \mathcal{K}$, a price vector $p \in R_+^J$, and a queue length vector $q \in R_+^J$ such that (2.8) is satisfied, we call the rate vector:

$$c^*(i) \in F_{\mathcal{Q}(q)}(i) \equiv F_{\mathcal{Q}(q)}^1(i) \times \dots \times F_{\mathcal{Q}(q)}^V(i),$$

a utility based 2-stage game-theoretic policy if it is a solution to the game problem in (3.20)-(3.21), which is obtained by firstly obtaining a saddle point to a zero-sum game problem and secondly obtaining a

static Pareto maximal-utility Nash equilibrium point to a non-zero-sum game problem, such that, for each $j \in \mathcal{J}(v) \setminus \mathcal{Q}(q)$ and any given $c(i) \in F_{\mathcal{Q}(q)}(i)$, the following facts are true,

$$\begin{cases} U_{00}(pq, c^*(i)) \geq U_{00}(pq, c(i)), \\ U_{vj}(pq, c^*(i)) \geq U_{vj}(q, c^*_{-j}(i)) \text{ if } j \in \mathcal{M}(i, v) \cap (\mathcal{J}(v) \setminus \mathcal{Q}(q)), v \in \{0\} \cup \mathcal{V}(j), \\ -U_{vj}(pq, c^*(i)) \geq -U_{vj}(pq, c^*_{-j}(i)) \text{ if } j \in (\mathcal{J}(v) \setminus \mathcal{Q}(q)) \setminus \mathcal{M}(i, v), v \in \{0\} \cup \mathcal{V}(j), \\ c^*_{-j}(i) \equiv (c^*_{-1}(i), \dots, c^*_{j-1}(i), c_{j+1}(i), \dots, c^*_j(i)). \end{cases}$$

3.3. Main theorem under the policy

Before stating our main theorem, we first introduce another concept of the so-called 2-stage dual-cost game-theoretic policy via a saddle point to a zero-sum game problem and a minimal-dual-cost Pareto Nash equilibrium point to a non-zero-sum game problem myopically at each given time point for a given price parameter $p \in R^J_+$. Then, based on the 2-stage dual-cost game-theoretic policy, we can inversely obtain the price vector and determine the target rate vector. To do so, we formulate a 2-stage minimal-dual-cost game problem in (3.22), which is corresponding to the utility based one in (3.20)-(3.21). More precisely, for a given $i \in \mathcal{K}$, a price parameter $p \in R^J_+$, a rate vector $c \in \mathcal{R}(i) \equiv \mathcal{R}_1(i) \times \dots \times \mathcal{R}_V(i)$, and a parameter $w \geq 0$, the 2-stage minimal-dual-cost game problem can be presented as follows:

$$\begin{cases} \min_{q \in R^J_+} C_{00}(pq, c), \\ \min_{q_j \in R_+, j \in \mathcal{M}(i, v) \cap (\mathcal{C}(c) \cap \mathcal{J}(v))} C_{vj}(pq, c), \\ \min_{q_j \in R_+, j \in (\mathcal{C}(c) \cap \mathcal{J}(v)) \setminus \mathcal{M}(i, v)} (-C_{vj}(pq, c)), \end{cases} \tag{3.22}$$

subject to

$$\sum_{j \in \mathcal{M}(i, v) \cap \mathcal{C}(c)} \frac{q_j}{\mu_j} \geq w,$$

where, the cost function $C_{vj}(pq, c)$ for each $j \in \mathcal{J}(v)$ and $v \in \{0\} \cup \mathcal{V}(j)$ is defined by:

$$\begin{cases} C_{00}(pq, c) = \sum_{j \in \mathcal{C}(c) \cap \mathcal{J}(v)} \sum_{v \in \mathcal{V}(j)} C_j(p_j q_j, c_{vj}), \\ C_{0j}(pq, c) = \sum_{v \in \mathcal{V}(j)} C_{vj}(p_j q_j, c_{vj}), \\ C_{vj}(pq, c) = C_{vj}(p_j q_j, c_{vj}) = \frac{1}{\mu_j} \int_0^{q_j} \frac{\partial U_{vj}(p_j u, c_{vj})}{\partial c_{vj}} du \text{ for } j \in \mathcal{C}(c) \cap \mathcal{J}(v), v \in \mathcal{V}(j), \end{cases}$$

and $\mathcal{C}(c)$ is an index set associated with the non-zero rates and non-empty queues, i.e.,

$$\mathcal{C}(c) \equiv \left\{ j : c_{.j} \neq 0 \text{ componentwise with } j \in \mathcal{J} \right\}.$$

In other words, if the environment is in state $i \in \mathcal{K}$, we try to find a queue state q for a given $c \in \mathcal{R}(i)$, a price parameter vector $p \in R^J_+$, and a given parameter $w \geq 0$ such that the individual user’s dual-costs and the total dual-cost over the system are all minimized at the same time while the (average) workload meets or exceeds w . Then, we have the following definitions.

Definition 3.3. For each state $i \in \mathcal{K}$, a price vector $p \in R^J_+$, and a rate vector $c(i) \in \mathcal{R}(i)$, a queue length vector $q^* \in R^J_+$ is called a dual-cost based 2-stage game-theoretic policy if it is a solution to the game problem in (3.22), which is obtained by firstly obtaining a saddle point to a zero-sum game

problem and secondly obtaining a static Pareto minimal-dual cost Nash equilibrium point to a non-zero-sum game problem, such that, for each $j \in \mathcal{C}(c)$, $v \in \{0\} \cup \mathcal{V}$, and any given $q \in R_+^J$ with $q_j = 0$ when $j \in \mathcal{J} \setminus \mathcal{C}(c)$, we have that:

$$\begin{cases} C_{00}(pq^*, c(i)) & \leq C_{00}(pq, c(i)), \\ C_{vj}(pq^*, c(i)) & \leq C_{vj}(pq_{-j}^*, c(i)) \text{ if } j \in \mathcal{M}(i, v) \cap (\mathcal{C}(c) \cap \mathcal{J}(v)), \\ -C_{vj}(pq^*, c(i)) & \leq -C_{vj}(pq_{-j}^*, c(i)) \text{ if } j \in (\mathcal{C}(c) \cap \mathcal{J}(v)) \setminus \mathcal{M}(i, v), \\ q_{-j}^* & \equiv (q_1^*, \dots, q_{j-1}^*, q_j, q_{j+1}^*, \dots, q_J^*). \end{cases} \tag{3.23}$$

Note that, once we obtain the queue policy point q^* with respect to the given price vector p from Definition 3.3, we can inversely deduce the corresponding price policy vector p in terms of q^* , i.e., $p = g(q^*)$ as in (2.8). This relationship can be used to design iterative algorithms in our numerical simulations. Furthermore, in Definition 3.3, we have used more strict concept of ‘‘Pareto optimal Nash equilibrium point’’, this concept can be relaxed to ‘‘Pareto optimal point’’ and the related theoretical discussion keeps true. In certain cases and when it is necessary, we can shift the Pareto optimal point to the Pareto optimal Nash equilibrium point by some mapping techniques.

Definition 3.4. Let $\hat{Q}^{r,(P,G)}(\cdot)$ and $\hat{W}^{r,(P,G)}(\cdot)$ be the diffusion-scaled queue length and workload processes, respectively, under an arbitrarily feasible dynamic pricing and rate scheduling policy (P, G) satisfying the Lipschitz condition in (2.8). A vector process $\hat{Q}(\cdot)$ is called an asymptotic dual-cost based 2-stage game-theoretic policy globally over the whole time horizon if, for any $t \geq 0$ and $v \in \{0\} \cup \mathcal{V}(j)$ with $j \in \mathcal{J}$, we have that:

$$\liminf_{r \rightarrow \infty} C_{00}(P(t)\hat{Q}^{r,(P,G)}(t), \rho_j(\alpha(t))) \geq C_{00}(P(t)\hat{Q}(t), \rho_j(\alpha(t))). \tag{3.24}$$

Furthermore, for each $j \in \mathcal{M}(\alpha(t), v, t) \cap (\mathcal{C}(c) \cap \mathcal{J}(v))$, we have that:

$$\liminf_{r \rightarrow \infty} C_{vj}(P(t)\hat{Q}_{-j}^{r,(P,G)}(t), \rho_j(\alpha(t))) \geq C_{vj}(P(t)\hat{Q}(t), \rho_j(\alpha(t))). \tag{3.25}$$

In addition, for each $j \in (\mathcal{C}(c) \cap \mathcal{J}(v)) \setminus \mathcal{M}(\alpha(t), v, t)$, we have that:

$$\liminf_{r \rightarrow \infty} \left(-C_{vj}(P(t)\hat{Q}_{-j}^{r,(P,G)}(t), \rho_j(\alpha(t))) \right) \geq -C_{vj}(P(t)\hat{Q}(t), \rho_j(\alpha(t))). \tag{3.26}$$

Note that, in (3.25)-(3.26) and for each $j \in \mathcal{J}$, we have that:

$$\hat{Q}_{-j}^{r,(P,G)}(t) = (\hat{Q}_1(t), \dots, \hat{Q}_{j-1}(t), \hat{Q}_j^{r,(P,G)}(t), \hat{Q}_{j+1}(t), \dots, \hat{Q}_J(t)). \tag{3.27}$$

Next, let $q^*(w, p, \rho(i))$ be a dual-cost based 2-stage game-theoretic policy corresponding to the game problem in (3.22) in terms of each given number $w \geq 0, p \in R_+^J$, and $i \in \mathcal{K}$ at a given time t . Furthermore, let $p(w, q^*, \rho(i))$ denote its corresponding inverse price vector with respect to q^* and construct price policy vector:

$$p^*(w, q^*, \rho(i)) = f(p(w, q^*, \rho(i))), \tag{3.28}$$

such that the Lipschitz condition in (2.8) is satisfied. Then, our main theorem can be presented as follows.

Theorem 3.5. For the game-competition based users-selection, dynamic pricing, and scheduling policy determined by (3.20)-(3.22) and (3.28) with $Q^r(0) = 0$ and conditions (6.3)-(6.8) (that will be detailed

in Section 6), the convergence in (3.2) is true. Furthermore, the limit queue length $\hat{Q}(\cdot)$ and the total workload $\hat{W}(\cdot)$ in (3.2) have the relationship:

$$\begin{cases} \hat{Q}(t) &= q^*(\hat{W}(t), \hat{P}(t), \rho(\alpha(t))), \\ \hat{P}(t) &= p^*(\hat{W}(t), \hat{Q}(t), \rho(\alpha(t))), \end{cases} \tag{3.29}$$

where $\hat{P}(\cdot)$ the inverse price vector process defined through (3.28) and $\hat{W}(\cdot)$ is a 1-dimensional RDRS in strong sense with:

$$\begin{cases} b(i, t) &= \sum_{j \in \cup_{v \in \mathcal{V}} \mathcal{M}(i, v, t)} \frac{\theta_j(t)}{\mu_j}, \\ \sigma^E(t) &= \sigma^S(t) = (\hat{\sigma}_1(t), \dots, \hat{\sigma}_J(t)), \\ \hat{\sigma}_j(t) &= \begin{cases} \frac{1}{\mu_j} & \text{if } j \in \cup_{v \in \mathcal{V}} \mathcal{M}(i, v, t), \\ 0 & \text{otherwise,} \end{cases} \\ R(i, t) &= 1 \end{cases} \tag{3.30}$$

for $t \in [0, \infty)$ and some constant $\theta_j(i)$ for each $j \in \cup_{v \in \mathcal{V}} \mathcal{M}(i, v, t)$. In addition, there is a common supporting probability space, under which and with probability one, the limit queue length $\hat{Q}(\cdot)$ is an asymptotic dual-cost based 2-stage game-theoretic policy globally over time interval $[0, \infty)$. Finally, the limit workload $\hat{W}(\cdot)$ is also asymptotic minimal in the sense that:

$$\liminf_{r \rightarrow \infty} \hat{W}^{r, (P, G)}(t) \geq \hat{W}(t). \tag{3.31}$$

The proof of Theorem 3.5 will be given in Section 6.

3.4. CNN-based algorithm flow chart

Based on the policy derived in Theorem 3.5, we can design a CNN based algorithm flow chart in Figure 6.

More precisely, for a constant $T \in [0, \infty)$, we divide the interval $[0, T]$ equally into n subintervals $\{[t_i, t_{i+1}], i \in \{0, 1, \dots, n - 1\}\}$ with $t_0 = 0, t_n = T$, and $\Delta t_i = t_{i+1} - t_i = \frac{T}{n}$. Furthermore, let

$$\Delta F(t_i) \equiv F(t_i) - F(t_{i-1}), \tag{3.32}$$

for each process $F(\cdot) \in \{B^E(\cdot), B^S(\cdot), \hat{W}(\cdot), \hat{Y}(\cdot)\}$. Then, we can develop an iterative procedure as shown in Figure 6 to conduct RDRS model based simulation studies and to illustrate the efficiency of our designed policy. In this algorithm, the main part is the policy computing concerning users-selection, dynamic pricing via queueing strategy, and rate scheduling in the federated learning center at each time t_{i+1} for given initial conditions at t_0 . As shown in Figure 2 and as mentioned in Introduction of this paper, our game-theoretic problem is a $J \times V$ -dimensional problem. In general, an explicit solution is not available. Thus, we need a numerical method as designed in Figure 6 to solve the 2-stage game-theoretic problem. The target is to get an associated saddle point to its corresponding zero-sum game problem and a Nash equilibrium point to its non-zero-sum game problem. In real-world system, J and V may take large numbers and our CNN algorithm exhibits *big model* behavior. Since our designed platform is a cloud computing (or even the near future quantum-cloud computing) based one, this *big model* can be effectively solved. However, to illustrate the usage of our designed algorithm and demonstrate the efficiency of our designed policy, we choose smaller J and V to conduct simulation case studies, which is presented in Sections 4–5.

Finally, note that the limit total workload $\hat{W}(t_{i+1})$ in Figure 6 can be replaced by the total workload in a real-world system if the input load of the system is close to heavy traffic (that will be detailed

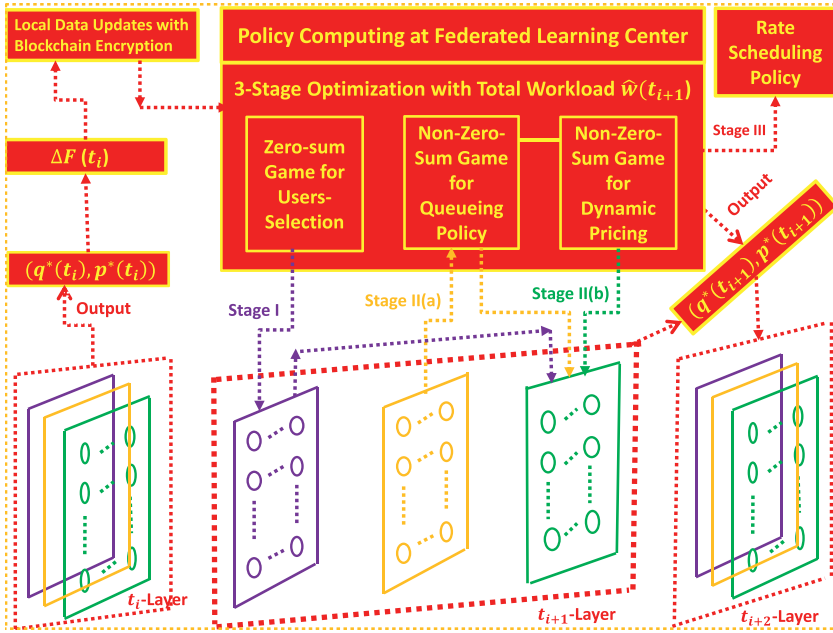


Figure 6. CNN based algorithm flow chart.

in Section 6). Furthermore, for the CNN algorithm flow chart designed in Figure 6, there are actually 4 processing stages in the federated learning center. The 2-stage game-theoretic problem is divided into 3 stages with an additional Stage II(b) to handle the dynamic pricing. The Stage III is directly corresponding to the integral relationship between the utility functions and their dual cost functions in (3.22).

4. Three illustrative policy examples

To be more illustrative and as a preparation of the following simulation case studies, we here present a 2-stage dynamic pricing and rate scheduling example and a 3-stage users-selection, dynamic pricing, and rate scheduling example based on a single-pool service system as shown in Figure 2. Hence, we will omit the pool index v .

4.1. The first example

The first example is corresponding to a 2-user case as shown in Figure 7, which can be used to model the DaiCoin based digital payment system with two types of Ethers (corresponding to two DaiCoins) as in Maker [18]. It can also be used to model a MIMO wireless communication channel (i.e., a single base station equipped with two antennas) shared by two-users or a quantum computer with two eigenmodes (see e.g., Dai [9, 12]). In this case, we are interested in the problem about how to price the two users and conduct the computing rate (i.e., power) resource allocation cooperatively inside a service system. More precisely, we take $V = 1$ with $J = 2$ and assume that the state space of the FS-CTMC $\alpha(t)$ defined in Subsection 2 consists only of a single state (i.e., $\alpha(t) \equiv 1$ for all $t \in [0, \infty)$). In a MIMO wireless environment, this case is associated with the so-called pseudo static channels. Then, the capacity region denoted by \mathcal{R} is supposed to be a non-degenerate convex one confined by five boundary lines including the two ones on x -axis and y -axis as shown in Figure 7. The capacity upper bound of the region satisfies $c_1 + c_2 = 2,000$. This region is corresponding to a degenerate fixed MIMO wireless channel of the generally randomized one in Dai [6]. For each price vector $p = (p_1, p_2) \in \mathbb{R}_+^2 = [0, \infty) \times [0, \infty)$

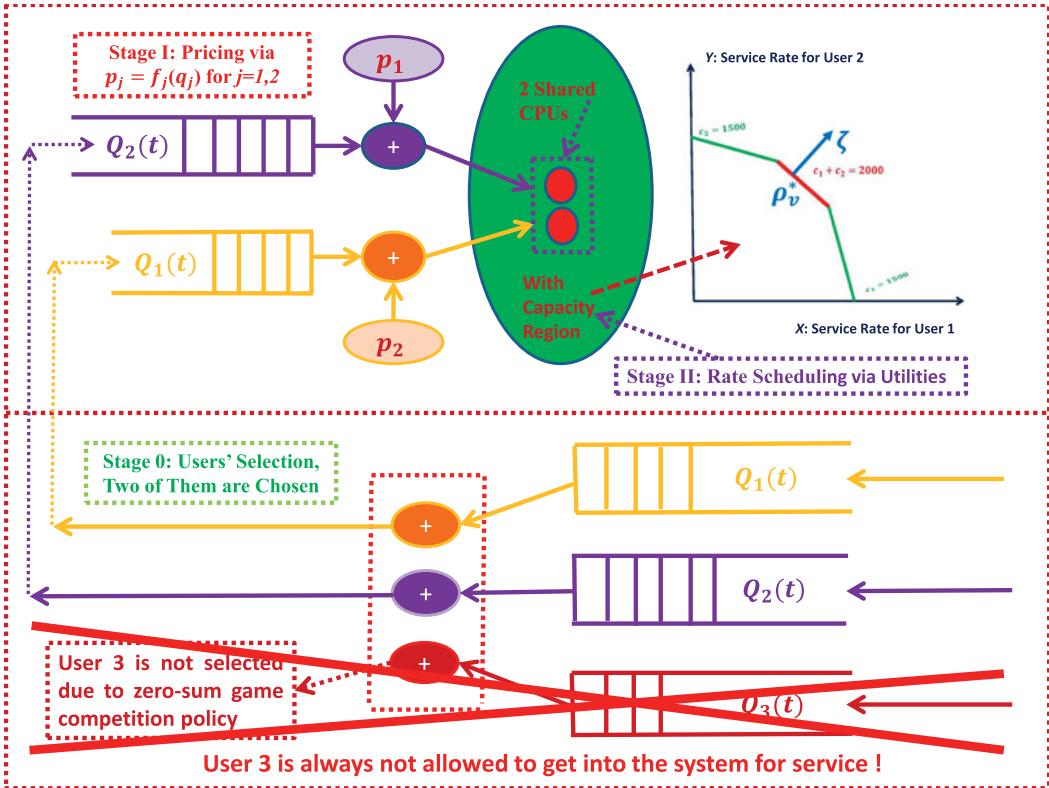


Figure 7. A 2-stage processing flow chart of dynamic pricing and rate scheduling for a single pool service system with 2-users.

corresponding to the process $P(t)$ in (2.8) and the queue length vector $q = (q_1, q_2) \in R_+^2 = [0, \infty) \times [0, \infty)$ corresponding to the process $Q(t)$ defined in (2.7) at a particular time point, we take the utility functions in terms of rate vector $c = (c_1, c_2) \in \mathcal{R}$ for user 1 and user 2, respectively, by:

$$U_1(pq, c) = U_1(p_1q_1, c_1) = p_1q_1 \ln(c_1), \quad U_2(pq, c) = U_2(p_2q_2, c_2) = -\frac{(p_2q_2)^2}{c_2^2}, \quad (4.1)$$

where $\ln(\cdot)$ is the logarithm function with the base e . Note that, the utility functions U_1 and U_2 in (4.1) are called proportionally fair and minimal potential delay allocations, respectively, which are widely used in communication systems. Furthermore, in a (quantum) blockchain system, these utility functions can be considered as generalized hash functions to replace the currently used random number generators to generate partial nonce values and private keys, i.e., $((p_1, p_2), (q_1, q_2), (c_1, c_2))$.

Example 4.1. For the upper graph case in Figure 7 and by the utility functions in (4.1), we can propose a 2-stage pricing and rate-scheduling policy at each time point $t \in [0, \infty)$ by a Pareto maximal-utility Nash equilibrium point to the following non-zero-sum game problem:

$$\max_{c \in \mathcal{R}} U_j(pq, c) \text{ for each } j \in \{0, 1, 2\} \text{ and a fixed } pq \in R_+^2, \quad (4.2)$$

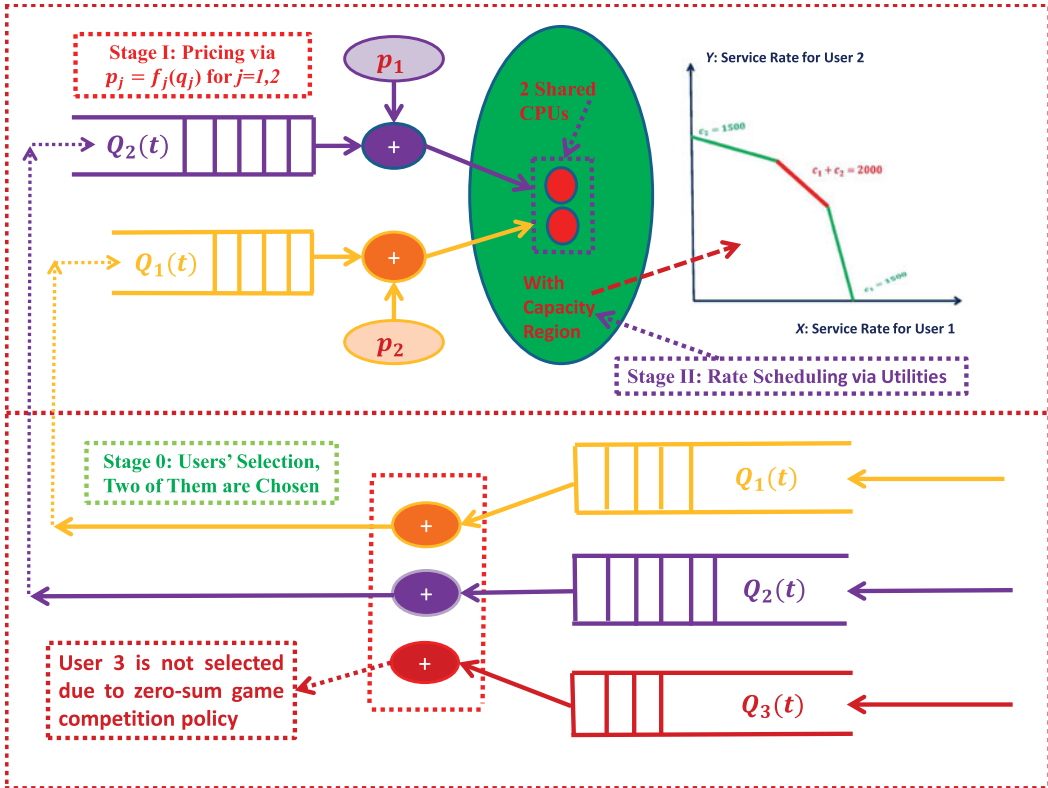


Figure 8. A 3-stage processing flow chart of users-selection, dynamic pricing, and rate scheduling for a single pool service system with 3-users.

where $U_0(pq, c) = U_1(pq, c) + U_2(pq, c)$. To wit, if $c^* = (c_1^*, c_2^*)$ is a solution to the game problem in (4.2), we have that:

$$\begin{cases} U_0(pq, c^*) \geq U_0(pq, c), \\ U_1(pq, c^*) \geq U_1(pq, c_{-1}^*) \quad \text{with } c_{-1}^* = (c_1, c_2^*), \\ U_2(pq, c^*) \geq U_2(pq, c_{-2}^*) \quad \text{with } c_{-2}^* = (c_1^*, c_2). \end{cases} \quad (4.3)$$

Furthermore, it follows from the inequalities in (4.3) that, if a game player’s (i.e., a user’s) rate service policy is unilaterally changed, his utility cannot be improved.

Remark 4.2. Due to the non-degenerate assumption imposed in (3.10)-(3.11), the strictly positive outer normal vector ζ to the facet \mathcal{O}_v with $v = 1$ at the point ρ_v^* exists, which is as shown in Figure 7. Hence, the so-called complete resource pooling condition (CRP) as introduced in Subsubsection 6.1.2 holds. In this case, a so-called fixed point defined as a Pareto minimal Nash equilibrium point to a dual-cost non-zero-sum game problem can be explicitly constructed, which is presented in (5.5). Interested readers are also referred to Dai [8], Ye and Yao [28] for more related discussions.

4.2. The second example

The second example is by adding Stage 0 for users’ selection in Figure 8. Comparing with the first case with $J = 2$, we here consider a 3-user case (i.e., $J = 3$) and add one more user selection layer. At each time point, we choose two of the three users for service according to a zero-sum game competition policy.

When any two users $i, j \in \{1, 2, 3\}$ with $i \neq j$ are selected, they will be served based on a non-zero-sum game competition policy. The capacity upper bound of the corresponding capacity region satisfies $c_i + c_j = 2,000$ as in the first 2-user case. Furthermore, suppose that, at a particular time point, there is a price vector $p = (p_1, p_2, p_3) \in R_+^3$ corresponding to the process $P(t)$ in (2.8) and a queue length vector $q = (q_1, q_2, q_3) \in R_+^3$ corresponding to the process $Q(t)$ defined in (2.7). Then, for each $(c_i, c_j) \in \mathcal{R}$, the corresponding utility functions are taken as in (4.1) if $i, j \in \{1, 2\}$. However, if $i = 3$ or $j = 3$, the corresponding utility function is taken to be the following one,

$$U_3(pq, c) = U_3(p_3q_3, c_3) = p_3q_3 \ln(c_3). \tag{4.4}$$

Example 4.3. For the second case corresponding to both the upper and lower graphs in Figure 8 and by the utility functions in (4.1) and (4.4), we can design a 3-stage users-selection, pricing and rate-scheduling policy myopically at each time point $t \in [0, \infty)$, which involves two steps as follows. First, we choose two users for service by a saddle point policy via the solution to the zero-sum game problem,

$$\max_{c \in \mathcal{R}} U_0(pq, c), \max_{c \in \mathcal{R}} U_{0j}(pq, c), \max_{c \in \mathcal{R}} (-U_{0j_1}(pq, c)), \max_{c \in \mathcal{R}} (-U_{0j_2}(pq, c)), \tag{4.5}$$

for each $j \in \{1, 2, 3\}, j_1 \in \{1, 2, 3\} \setminus \{j\}, j_2 \in \{1, 2, 3\} \setminus \{j, j_1\}$, and a fixed $pq \in R_+^3$, where,

$$\begin{cases} U_0(pq, c) &= U_1(pq, c) + U_2(pq, c) + U_3(pq, c), \\ U_{01}(pq, c) &= U_1(pq, c) + U_2(pq, c), \\ U_{02}(pq, c) &= U_1(pq, c) + U_3(pq, c), \\ U_{03}(pq, c) &= U_2(pq, c) + U_3(pq, c). \end{cases} \tag{4.6}$$

In other words, if $c^* = (c_1^*, c_2^*, c_3^*)$ is a solution to the game problem in (4.5), and if

$$c_{-j}^* = \begin{cases} (c_j, c_{j_1}^*, c_{j_2}^*) & \text{if } j = 1, \\ (c_{j_1}^*, c_j, c_{j_2}^*) & \text{if } j = 2, \\ (c_{j_1}^*, c_{j_2}^*, c_j) & \text{if } j = 3, \end{cases}$$

then, for a fixed $pq \in R_+^3$, we have that:

$$\begin{cases} U_0(pq, c^*) &\geq U_0(pq, c), \\ U_{0j}(pq, c^*) &\geq U_{0j}(pq, c_{-j}^*), \\ -U_{0j_1}(pq, c^*) &\geq -U_{0j_1}(pq, c_{-j_1}^*), \\ -U_{0j_2}(pq, c^*) &\geq -U_{0j_2}(pq, c_{-j_2}^*). \end{cases} \tag{4.7}$$

Second, when two users corresponding to the summation $U_{0j} = U_k + U_l$ for an index $j \in \{1, 2, 3\}$ with two associated indices $k, l \in \{1, 2, 3\}$ as in one of (4.6) are selected, we can propose a 2-stage pricing and rate-scheduling policy at each time point by a Pareto maximal-utility Nash equilibrium point to the non-zero-sum game problem for a fixed $pq \in R_+^3$,

$$\max_{c \in \mathcal{R}} U_{0j}(pq, c), \max_{c \in \mathcal{R}} U_k(pq, c), \max_{c \in \mathcal{R}} U_l(pq, c). \tag{4.8}$$

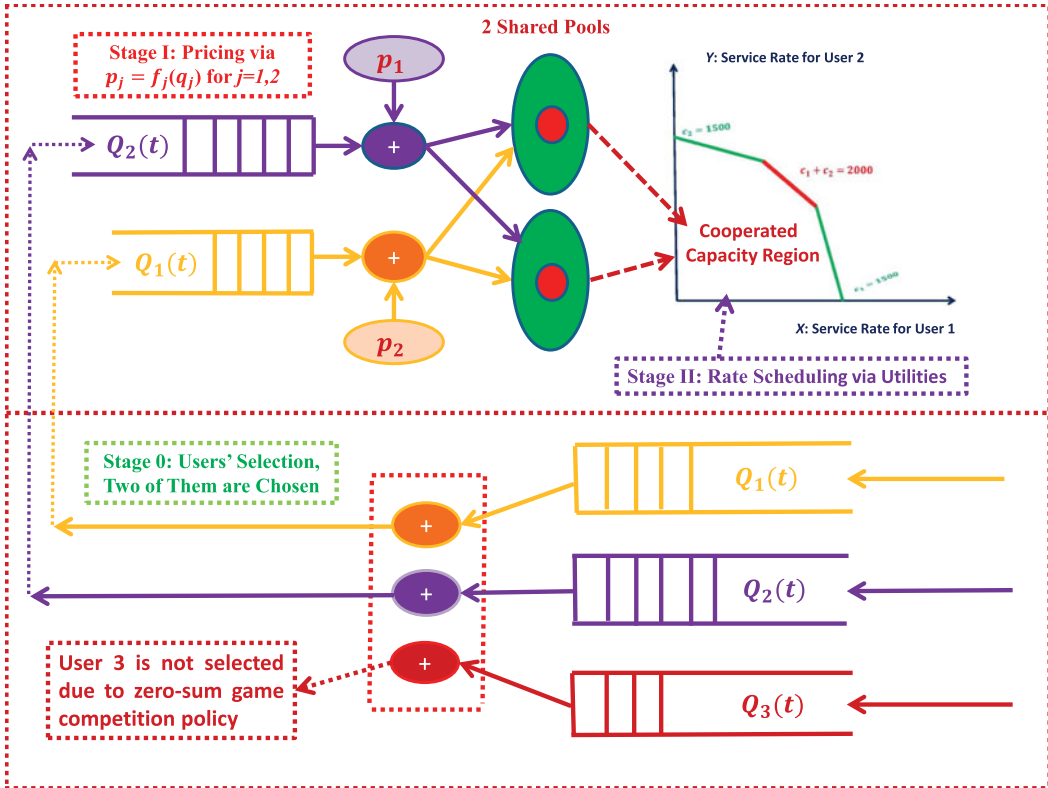


Figure 9. A 3-stage processing flow chart of users-selection, dynamic pricing, and rate scheduling for a service system with 2-pools and 3-users.

To wit, if $c^* = (c_k^*, c_l^*)$ is a solution to the game problem in (4.8), we have that:

$$\begin{cases} U_{0j}(pq, c^*) \geq U_{0j}(pq, c), \\ U_k(pq, c^*) \geq U_k(pq, c_{-k}^*) \quad \text{with } c_{-k}^* = (c_k, c_l^*), \\ U_l(pq, c^*) \geq U_l(pq, c_{-l}^*) \quad \text{with } c_{-l}^* = (c_k^*, c_l). \end{cases} \quad (4.9)$$

4.3. The third example

The third example is corresponding to a case with 2 pools (i.e., $V = 2$) as shown in Figure 9. In reality, this system is corresponding to a MIMO wireless communication system with two base stations. Each of the base station is equipped with a single antenna. However, the two base stations can cooperated each other to form a transmission channel with a capacity region as shown in Figure 9. The rest of the illustration is similar to the one for the second example. Hence, we omit it here.

5. Simulation case studies via RDRS models

In this section, we conduct simulation case studies for Examples 4.1–4.3 presented in Section 4. The simulation for the third example with 2-pools in Section 4 is similar to the one for Example 4.3. Hence, we omit it here. The main point of these simulation studies is to illustrate our policies proposed in the two examples outperform several policies in certain ways. These policies used for the purpose of comparisons include an existing constant pricing policy, an existing 2D-Queue policy, a newly designed randomly users' selection stochastic pooling policy, and an arbitrarily selected dynamic pricing policy.

As mentioned in Section 4, Examples 4.1–4.3 are corresponding to a single-pool system with two-users and three-users, respectively. Thus, we will omit all the related pool index v . In an associated real-world system, the parameter vectors p and q in (4.2) (or (4.5)) are the randomly evolving pricing process $P(t)$ in (2.8) and the queue length process $Q(t)$ in (2.7). Hence, it is our concern of this section about how to employ the RDRS performance model in Definition 3.1 to evaluate the usefulness of our proposed myopic users-selection, dynamic pricing, and scheduling policies globally over the whole time horizon $[0, \infty)$ for Examples 4.1–4.3. To interpret our numerical simulation implementations, we first identify the corresponding dual-cost functions $\{C_j(q, c)\}$ as defined in (3.22) with $j \in \{1, 2, 3\}$ for the associated $U_j(q, c)$ given in (4.1) and (4.4). More precisely,

$$\begin{cases} C_1(p_1q_1, c_1) &= \frac{1}{\mu_1} \int_0^{q_1} \frac{\partial U_1(p_1u, c_1)}{\partial c_1} du = \frac{(p_1q_1)^2}{2\mu_1c_1}, \\ C_2(p_2q_2, c_2) &= \frac{1}{\mu_2} \int_0^{q_2} \frac{\partial U_2(p_2u, c_2)}{\partial c_2} du = \frac{2(p_2q_2)^3}{3\mu_2c_2^3}, \\ C_3(p_3q_3, c_3) &= \frac{1}{\mu_3} \int_0^{q_3} \frac{\partial U_3(p_3u, c_3)}{\partial c_3} du = \frac{(p_3q_3)^2}{2\mu_3c_3}, \end{cases} \tag{5.1}$$

where $1/\mu_j$ for all $j \in \{1, 2, 3\}$ are average quantum packet lengths associated with the three users as explained just before (2.6).

5.1. The simulation for example 4.1

Based on the first two dual-cost functions in (5.1), we can formulate a corresponding 2-stage minimal dual-cost non-zero-sum game problem for a price parameter $p \in R_+^2$ as follows,

$$\min_{q \in R_+^2} C_j(pq, c) \quad \text{subject to} \quad \frac{q_1}{\mu_1} + \frac{q_2}{\mu_2} \geq w, \tag{5.2}$$

for a fixed constant $w > 0$, a fixed $c \in \mathcal{R}$, and all $j \in \{0, 1, 2\}$ with $C_0(pq, c) = C_1(pq, c) + C_2(pq, c)$. Since $C_j(p_jq_j, c_j)$ for each $j \in \{1, 2\}$ is strictly increasing with respect to p_jq_j (or simply q_j), a Pareto minimal dual-cost Nash equilibrium point to the problem in (5.2) must be located on the line where the equality of the constraint inequality is true (i.e., $q_1/\mu_1 + q_2/\mu_2 = w$). Thus, we know that:

$$q_j = \mu_j \left(w - \frac{q_{2-j+1}}{\mu_{2-j+1}} \right) \quad \text{with} \quad j \in \{1, 2\}. \tag{5.3}$$

Hence, it follows from (5.3) that:

$$\bar{f}(q_1) \equiv \sum_{j=1}^2 C_j(p_jq_j, c_j) = \frac{p_1^2q_1^2}{2\mu_1c_1} + \frac{2p_2^3\mu_2^2}{3c_2^3} \left(w - \frac{q_1}{\mu_1} \right)^3. \tag{5.4}$$

Then, by solving the equation $\frac{\partial \bar{f}(q_1)}{\partial q_1} = 0$, we can get the minimal value of the function $\bar{f}(q_1)$ for each $p \in R_+^2$. More precisely, the unique Pareto minimal point $q^*(p, w) = (q_1^*, q_2^*)(p, w)$ to the problem in (5.2) can be explicitly given by:

$$\begin{cases} q_1^*(p, w) &\equiv \bar{g}_1(p, w) = \frac{1}{2} \left(\frac{2w}{\mu_1} + \frac{p_1^2c_2^3}{2p_2^3c_1\mu_2^2} \right) \mu_1^2 - \sqrt{\frac{1}{4} \left(\frac{2w}{\mu_1} + \frac{p_1^2c_2^3}{2p_2^3c_1\mu_2^2} \right)^2 \mu_1^4 - \mu_1^2w^2}, \\ q_2^*(p, w) &\equiv \bar{g}_2(p, q_1^*(p, w), w) = \mu_2 \left(w - \frac{q_1^*(p, w)}{\mu_1} \right). \end{cases} \tag{5.5}$$

From the green curve in the upper-left graph of Figure 10 where $p_1 = p_2 = 1$ and $w = 10,000$, we can see that this point is close to the one corresponding to $q_1 = 0$ and we can consider it as a Pareto

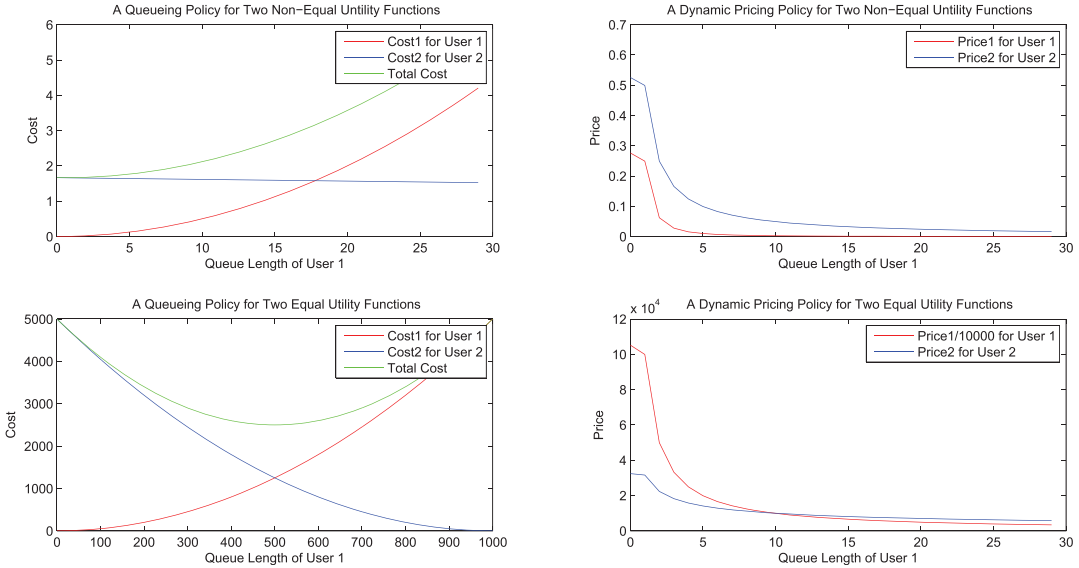


Figure 10. Pareto optimal Nash equilibrium policies with dynamic pricing, where, the Price1/10,000 in the lower-right graph means that Price1 is divided by 10,000.

minimal Nash equilibrium point near boundary. Another Nash equilibrium point is the intersection point of the red and blue curves in the left graph of Figure 10. Obviously, this point is not a minimal total cost point. However, we can use some transformation technique to shift the minimal point to this one and to design a more fairly balanced decision policy. Nevertheless, for the purpose of this research in finding the Pareto utility-maximization Nash equilibrium policy, we use the point in (5.5) as our decision policy. In this case, for the price parameter $p \in R_+^2$, we have:

$$\begin{cases} C_0(pq^*, c) \leq C_0(pq, c), \\ C_1(pq^*, c) \leq C_1(pq_{-1}^*, c) \quad \text{with } q_{-1}^* = (q_1, q_2^*), \\ C_2(pq^*, c) \leq C_2(pq_{-2}^*, c) \quad \text{with } q_{-2}^* = (q_1^*, q_2). \end{cases} \quad (5.6)$$

Then, associated with a given queue length based Pareto minimal Nash equilibrium point in (5.5), we can obtain the relationship between prices p_1 and p_2 as follows,

$$\frac{p_1^2(q_1, w)}{p_2^3(q_1, w)} \equiv \kappa(q_1, w) = \left(\frac{2c_1\mu_2^2}{c_2^3} \right) \left(\frac{\mu_1^2 w^2 + q_1^2}{\mu_1^2 q_1} - \frac{2w}{\mu_1} \right). \quad (5.7)$$

From (5.7), we can see that there are different choices of dynamic pricing policies corresponding to Pareto minimal Nash equilibrium point $q^*(p, w)$ in (5.5). For the current study, we take

$$\begin{cases} p_1(q_1, w) = \kappa^2(q_1, w), \\ p_2(q_1, w) = \kappa(q_1, w), \end{cases} \quad (5.8)$$

whose dynamic evolutions with the queue length q_1 are shown in the upper-right graph of Figure 10.

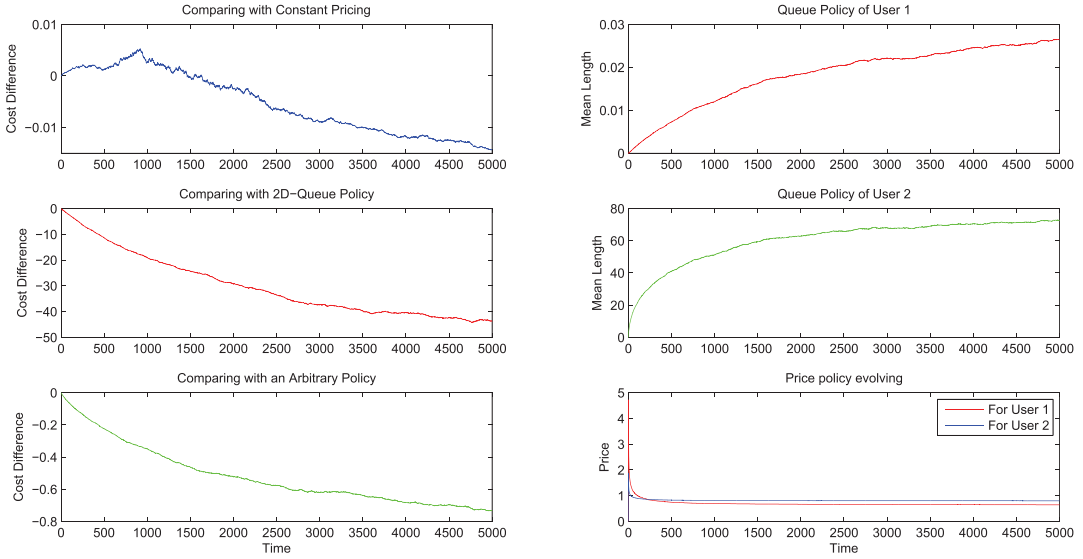


Figure 11. In this simulation, the number of simulation iterative times is $N = 6,000$, the simulation time interval is $[0, T]$ with $T = 200$, which is further divided into $n = 5,000$ subintervals as explained in Subsection 5. Other values of simulation parameters introduced in Definition 3.1 and Subsubsection 4 are as follows: $initialprice1 = 9$, $initialprice2 = 3$, $lowerboundprice1 = 0.64$, $lowerboundprice2 = 0.8$, $\lambda_1 = 10/3$, $\lambda_2 = 5$, $m_1 = 3$, $m_2 = 1$, $\mu_1 = 1/10$, $\mu_2 = 1/20$, $\alpha_1 = \sqrt{10/3}$, $\alpha_2 = \sqrt{20}$, $\beta_1 = \sqrt{10}$, $\beta_2 = \sqrt{20}$, $\zeta_1 = 1$, $\zeta_2 = \sqrt{2}$, $\rho_1 = \rho_2 = 1,000$, $c_1^2 = c_2^1 = 1,500$, $\theta_1 = -1$, $\theta_2 = -1.2$.

Next, by Theorem 3.5, we know that the coefficients of the 1-dimensional RDRS under our dynamic pricing and game-based scheduling policy for the physical workload process \hat{W} can be denoted by:

$$\begin{cases} \hat{b} &= \theta_1/\mu_1 + \theta_2/\mu_2, \quad \hat{\sigma}^E = \hat{\sigma}^S = (1/\mu_1, 1/\mu_2), \quad \hat{R} = 1, \\ \hat{\sigma} &= \sqrt{\left(\sum_{j=1}^2 \hat{\sigma}_j^E \sqrt{\Gamma_{jj}^E}\right)^2 + \left(\sum_{j=1}^2 \hat{\sigma}_j^S \sqrt{\Gamma_{jj}^S}\right)^2}. \end{cases} \tag{5.9}$$

Then, based on \hat{W} , we can get the dynamic queueing policy by (5.5) and its associated dynamic pricing policy through (5.8):

$$\hat{Q}(t) = q^*(\hat{P}(t), \hat{W}(t)), \quad \hat{P}(t) = \hat{P}(\hat{Q}_1(t), \hat{W}(t)). \tag{5.10}$$

After determining the initial prices $\hat{P}(0) = (initialprice1, initialprice2)$, we suppose that $\hat{P}(t)$ has the lower bound price protection functionality, i.e., $\hat{P}(t) \in [lowerboundprice1, \infty) \times [lowerboundprice2, \infty)$. Corresponding to (5.8), this truncated price process still owns the Lipschitz continuity as imposed in (2.8). Then, by combining the policy in (5.10) with the simulation algorithms for RDRSs we can illustrate our policy in (5.10) is cost-effective in comparing with a constant pricing policy, a 2D-Queue policy and an arbitrarily selected dynamic pricing policy. These simulation comparisons are presented in Figure 11 with different parameters. The number N of simulation iterative times for these comparisons is 6,000 and the simulation time interval is $[0, T]$ with $T = 200$, which is further divided into $n = 5,000$ subintervals. The first graph on the left-column in Figure 11 is the mean total cost difference (MTCD) at each time point t_i with $i \in \{0, 1, \dots, 5,000\}$ between our current dynamic

pricing policy in (5.10) and the constant pricing policy with $P_1(t) = P_2(t) = 1$ for all $t \in [0, \infty)$, i.e.,

$$\text{MTCD}(t_i) = \frac{1}{N} \sum_{j=1}^N \left(C_0(\hat{P}(\omega_j, t_i) \hat{Q}(\omega_j, t_i), \rho) - C_0(P(\omega_j, t_i) Q(\omega_j, t_i), \rho) \right), \quad (5.11)$$

where ω_j denotes the j th sample path and the $Q(\omega_j, t_i)$ in (5.11) is the queue length corresponding to the constant pricing policy at each time point t_i . The second graph on the left-column in Figure 11 is the MTCD between our newly designed dynamic pricing policy in (5.10) and a 2D-Queue policy used as an alternative comparison policy in Dai [8]. For this 2D-Queue policy, the constant pricing with $P_1(t) = P_2(t) = 1$ is employed and the associated $Q(t)$ is presented as a two-dimensional RDRS model as in Dai [8]. The third graph on the left-column in Figure 11 is the MTCD between our current dynamic pricing policy in (5.10) and an arbitrarily selected dynamic pricing policy given by:

$$\begin{cases} P_1(t) = \text{lowerboundprice1} + \frac{20}{0.05 + \sqrt{Q_1(t)}}, \\ P_2(t) = \text{lowerboundprice2} + \frac{30}{0.1 + \sqrt{Q_2(t)}} \end{cases} \quad (5.12)$$

with the associated queue policy $Q(t) = \hat{Q}(t)$. The first and second graphs on the right-column in Figure 11 display the dynamics of $\hat{Q}(t)$ for both users. The third graph on the right-column in Figure 11 shows the price evolutions corresponding to two users. From the first graph in Figure 11, we can see that the cost is relatively large if the initial prices are relatively high. All of the other comparisons in Figure 11 show the cost-effectiveness of our policy in (5.10).

5.2. The simulation for example 4.3

Based on the three dual-cost functions in (5.1), we can first select any two of the three users for service by formulating the following minimal dual-cost zero-sum game problem for a price parameter $p \in R_+^3$, a constant $w > 0$, and a fixed $c \in \mathcal{R}$,

$$\begin{cases} \min_{q \in R_+^3} C_0(pq, c), \\ \min_{q \in R_+^3} C_{0j}(pq, c) & \text{subject to } q_1/\mu_1 + q_2/\mu_2 \geq w, \\ \min_{q \in R_+^3} (-C_{0j_1}(pq, c)) & \text{subject to } q_1/\mu_1 + q_3/\mu_3 \geq w, \\ \min_{q \in R_+^3} (-C_{0j_2}(pq, c)) & \text{subject to } q_2/\mu_2 + q_3/\mu_3 \geq w, \end{cases} \quad (5.13)$$

where $j \in \{1, 2, 3\}$, $j_1 \in \{1, 2, 3\} \setminus \{j\}$, and $j_2 \in \{1, 2, 3\} \setminus \{j, j_1\}$, and

$$\begin{cases} C_0(pq, c) = C_1(p_1q_1, c_1) + C_2(p_2q_2, c_2) + C_3(p_3q_3, c_3), \\ C_{01}(pq, c) = C_1(p_1q_1, c_1) + C_2(p_2q_2, c_2), \\ C_{02}(pq, c) = C_1(p_1q_1, c_1) + C_3(p_3q_3, c_3), \\ C_{03}(pq, c) = C_2(p_2q_2, c_2) + C_3(p_3q_3, c_3). \end{cases} \quad (5.14)$$

In other words, if $q^* = (q_1^*, q_2^*, q_3^*)$ is a solution to the game problem in (5.13), and if,

$$q_{-j}^* = \begin{cases} (q_j, q_{j_1}^*, q_{j_2}^*) & \text{if } j = 1, \\ (q_{j_1}^*, q_j, q_{j_2}^*) & \text{if } j = 2, \\ (q_{j_1}^*, q_{j_2}^*, q_j) & \text{if } j = 3, \end{cases} \quad (5.15)$$

then, for any two fixed $p, c \in R_+^3$, we have that:

$$\begin{cases} C_0(pq^*, c) & \leq C_0(pq, c), \\ C_{0j}(pq^*, c) & \leq C_{0j}(pq_{-j}^*, c), \\ -C_{0j_1}(pq^*, c) & \leq -C_{0j_1}(pq_{-j_1}^*, c), \\ -C_{0j_2}(pq^*, c) & \leq -C_{0j_2}(pq_{-j_2}^*, c). \end{cases} \tag{5.16}$$

Furthermore, when two users corresponding to the summation $C_{0j} = C_k + C_l$ for an index $j \in \{1, 2, 3\}$ with two associated indices $k, l \in \{1, 2, 3\}$ as in one of (5.13) are selected, we can propose a 2-stage pricing and queueing policy at each time point by a Pareto minimal dual cost Nash equilibrium point to the non-zero-sum game problem for two fixed $p, c \in R_+^3$,

$$\min_{q \in R_+^3} C_{0j}(pq, c), \quad \min_{q \in R_+^3} C_k(pq, c), \quad \min_{q \in R_+^3} C_l(pq, c). \tag{5.17}$$

To wit, if $q^* = (q_k^*, q_l^*)$ is a solution to the game problem corresponding to the two users, we have that:

$$\begin{cases} C_{0j}(pq^*, c) & \geq C_{0j}(pq, c), \\ C_k(pq^*, c) & \geq C_k(pq_{-k}^*, c) \quad \text{with } q_{-k}^* = (q_k, q_l^*), \\ C_l(pq^*, c) & \geq C_l(pq_{-l}^*, c) \quad \text{with } q_{-l}^* = (q_k^*, q_l). \end{cases} \tag{5.18}$$

Thus, for the price parameter $p \in R_+^3$ and each $w \geq 0$, it follows from (5.13)-(5.16) and (5.17)-(5.18) that our queueing policy $(q_1^*(p, w), q_2^*(p, w), q_3^*(p, w))$ can be designed by:

$$\begin{cases} \begin{cases} q_1^*(p, w) = \bar{g}_1(p_1, p_2, w), \\ q_2^*(p, w) = \bar{g}_2(p_1, p_2, q_1^*, w), \end{cases} & \text{if } C_{01}(pq^*, c) \leq \min \{C_{02}(pq^*, c), C_{03}(pq^*, c)\}, \\ \begin{cases} q_1^*(p, w) = \hat{g}_1(p_1, p_3, w), \\ q_3^*(p, w) = \hat{g}_3(p_1, p_3, q_1^*, w), \end{cases} & \text{if } C_{02}(pq^*, c) \leq \min \{C_{01}(pq^*, c), C_{03}(pq^*, c)\}, \\ \begin{cases} q_2^*(p, w) = \bar{g}_2(p_3, p_2, q_3^*, w), \\ q_3^*(p, w) = \bar{g}_1(p_3, p_2, w), \end{cases} & \text{if } C_{03}(pq^*, c) \leq \min \{C_{01}(pq^*, c), C_{02}(pq^*, c)\}, \end{cases} \tag{5.19}$$

where the function \bar{g}_1 is given in (5.5) and \hat{g}_1 is calculated in a similar way as follows,

$$\begin{cases} \hat{g}_1(p_1, p_3, w) & = \frac{(p_3^2 \mu_3 w) / (\mu_1 c_3)}{(p_1^2 / \mu_1 c_1) + (p_3^2 \mu_3 / \mu_1^2 c_3)}, \\ \hat{g}_3(p_1, p_3, q_1, w) & = \mu_3 (w - (q_1 / \mu_1)). \end{cases} \tag{5.20}$$

The intersection point of \hat{g}_1 and \hat{g}_3 in terms of q_1 is a Pareto optimal Nash equilibrium point as shown in the lower-left graph of Figure 10. Furthermore, based on (5.19)-(5.20), we can inversely determine our pricing policy $p = (p_1, p_2, p_3)$ as follows,

$$\begin{cases} \begin{cases} p_1(q_1^*, w) = \kappa^2(q_1^*, w), \\ p_2(q_1^*, w) = \kappa(q_1^*, w) \end{cases} & \text{if } C_{01}(pq^*, c) \leq \min \{C_{02}(pq^*, c), C_{03}(pq^*, c)\}, \\ \begin{cases} p_1(q_1^*, w) = \varpi(q_1^*) \hat{\kappa}(q_1^*, w), \\ p_3(q_1^*, w) = \varpi(q_1^*) \sqrt{\hat{\kappa}(q_1^*, w)} \end{cases} & \text{if } C_{02}(pq^*, c) \leq \min \{C_{01}(pq^*, c), C_{03}(pq^*, c)\}, \\ \begin{cases} p_2(q_3^*, w) = \kappa(q_3^*, w), \\ p_3(q_3^*, w) = \kappa^2(q_3^*, w) \end{cases} & \text{if } C_{03}(pq^*, c) \leq \min \{C_{01}(pq^*, c), C_{02}(pq^*, c)\}, \end{cases} \tag{5.21}$$

where, κ is defined in (5.7) and $\hat{\kappa}$ can be calculated in the same way as follows,

$$\hat{\kappa}(q_1^*, w) = \frac{\mu_3 c_1}{q_1^* c_3} \left(w - \frac{q_1^*}{\mu_1} \right). \tag{5.22}$$

Furthermore, $\varpi(q_1^*)$ in (5.21) is a Non-negative function in terms of q_1^* and it is taken to be the unity in the drawing of dynamic pricing evolving in the lower-graph of Figure 10 with $w = 10,000$.

To show the cost-effectiveness of our queueing policy in (5.19) with its associated pricing policy in (5.21), we present an arbitrarily selected stochastic pooling policy for the purpose of comparisons as follows,

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} q_1^*(p, w) = \bar{g}_1(p_1, p_2, w), \\ q_2^*(p, w) = \bar{g}_2(p_1, p_2, q_1^*, w) \end{array} \right. \quad \text{if } u \in \left[0, \frac{1}{3}\right), \\ \left\{ \begin{array}{l} q_1^*(p, w) = \hat{g}_1(p_1, p_3, w), \\ q_3^*(p, w) = \hat{g}_3(p_1, p_3, q_1^*, w) \end{array} \right. \quad \text{if } u \in \left[\frac{1}{3}, \frac{2}{3}\right), \\ \left\{ \begin{array}{l} q_2^*(p, w) = \bar{g}_2(p_3, p_2, q_3^*, w), \\ q_3^*(p, w) = \bar{g}_1(p_3, p_2, w) \end{array} \right. \quad \text{if } u \in \left[\frac{2}{3}, 1\right], \end{array} \right. \tag{5.23}$$

where u is a uniformly distributed random number.

After determining the initial price vector $\hat{P}(0) = (\text{initialprice1}, \text{initialprice2}, \text{initialprice3})$, we suppose that $\hat{P}(t)$ has the lower bound price protection and the upper bound constraint functionalities, i.e., $\hat{P}(t) \in [\text{lowerboundprice1}, \text{upperboundprice1}] \times [\text{lowerboundprice2}, \text{upperboundprice2}] \times [\text{lowerboundprice3}, \text{upperboundprice3}]$. Corresponding to (5.21), this truncated price process still own the Lipschitz continuity as imposed in (2.8). Then, by the similar explanations used for (5.11), we can conduct the corresponding simulation comparisons for this example as shown in Figures 4–5. The cost value evolution based on our queueing policy in (5.19) with its associated pricing policy in (5.21) is shown in the first graph of the left-column in each of Figures 4–5. Its MTCD in (5.11) compared with the arbitrarily selected stochastic pooling policy in (5.23) is displayed in the first graph of the right-column in each of Figures 4–5. The cost value evolution based on our queueing policy in (5.19) with constant pricing (i.e., $p_1 = p_2 = p_3$) is shown in the second graph of the left-column in each of Figures 4–5. In this constant pricing case, its MTCD in (5.11) compared with the arbitrarily selected stochastic pooling policy in (5.23) is displayed in the second graph of the right-column in each of Figures 4–5. The MTCD based on our queueing policy in (5.19) with its associated pricing policy in (5.21) and with the constant pricing policy is shown in the third graph of the left-column in each of Figures 4–5. The price evolutions for the three users are shown in the third graph of the right-column in each of Figures 4–5. In the special case with parameters as shown in Figure 5, the three price evolutions are the same. Furthermore, the MTCD between our dynamic pricing policy in (5.21) and the constant pricing policy is the number 0 as shown in the third graph of the left-column in Figure 5.

6. Justification of RDRS modeling

In this section, we theoretically prove the correctness of our RDRS modeling presented in Theorem 3.5. To be convenient for readers, we first outline the proof for Theorem 3.5, which is a technical generalization of the corresponding proofs in existing discussions in Dai [8, 9]. The major breakthrough in this generalization is to incorporate the dynamic pricing functions in (2.8)-(2.9) in terms of both the queue length $Q_j(t)$ with $j \in \{1, \dots, J\}$ and the random environment $\alpha(t)$ into the proof. In Dai [8], we prove a corresponding theorem for a generic game platform where the game-theoretic oriented resource scheduling policy is a solely “win-win” fairly sharing non-zero-sum game oriented one, which is designed for multiple resources-sharing and fairly competing users. The studied platform in Dai [8] consists of multiple intelligent (quantum) cloud-computing pools and parallel-queues. The arrival data streams

associated with all the users are modeled by TSRRPs. Every user in the system can be served at the same time by multiple service pools and in the meantime every service pool consisting of parallel-servers can also provide services to multi-users simultaneously. The associated RDRS performance model is established under the “win-win” fairly resources-sharing scheduling policy via diffusion approximation. The study in Dai [8] is extended to the case for a “win-lose & win-win” 2-stage zero-sum and non-zero-sum mixed game-theoretic scheduling policy based platform in Dai [9]. However, the study in Dai [9] is still a 2-stage resources-competition and resources-sharing oriented one, which does not consider the dynamic pricing issue. Therefore, in the discussions of the following two subsections for proving our current Theorem 3.5, we incorporate the dynamic pricing functions in (2.8)-(2.9) into consideration, which is along the line of the corresponding proof in Dai [9].

6.1. The required conditions

In this subsection, we present the required conditions in proving our RDRS modeling.

6.1.1. Conditions on utility functions

The utility functions can be either simply taken as the well-known proportionally fair and minimal potential delay allocations as used in (4.1) for Example 5 or generally taken such that the existence of a utility based 2-stage game-theoretic policy corresponding to the game problem in (3.20)-(3.21) is guaranteed. More precisely, for each given $p \in \mathbb{R}_+^L$, we can assume that $U_{vj}(p_j q_j, c_{vj})$ for each $j \in \mathcal{J}(v)$ and $v \in \mathcal{V}(j)$ is defined on \mathbb{R}_+^L . It is second-order differentiable and satisfies:

$$\left\{ \begin{array}{l} U_{vj}(0, c_{vj}) = 0, \\ U_{vj}(p_j q_j, c_{vj}) = \Phi_{vj}(p_j q_j) \Psi_v(c_{vj}) \text{ is strictly increasing/concave in } c_{vj} \text{ for } p_j q_j > 0, \\ \Psi_v(v_j c_{vj}) = \Psi_v(v_j) \Psi_v(c_{vj}) \text{ or } \Psi_v(v_j c_{vj}) = \Psi_v(v_j) + \Psi_v(c_{vj}) \text{ for constant } v_j \geq 0, \\ \frac{\partial U_{vj}(p_j q_j, c_{vj})}{\partial c_{vj}} \text{ is strictly increasing in } p_j q_j \geq 0, \\ \frac{\partial U_{vj}(0, c_{vj})}{\partial c_{vj}} = 0 \text{ and } \lim_{q_j \rightarrow \infty} \frac{\partial U_{vj}(p_j q_j, c_{vj})}{\partial c_{vj}} = +\infty \text{ for each } c_{vj} > 0. \end{array} \right. \quad (6.1)$$

Furthermore, we suppose that $\{U_{vj}(p_j q_j, c_{vj}), j \in \mathcal{J}(v), v \in \mathcal{V}(j)\}$ satisfies the radial homogeneity condition at each given time point $t \in [0, \infty)$. In other words, for any scalar $a > 0$, each $q > 0$, $i \in \mathcal{K}$, $v \in \mathcal{V}$, and each $j_l \in \mathcal{M}(i, v, t)$ with $l \in \{1, \dots, M_v\}$, its Pareto maximal utility Nash equilibrium point for the game has the radial homogeneity:

$$c_{vj_l}(apq, i) = c_{vj_l}(pq, i). \quad (6.2)$$

6.1.2. Complete resource pooling condition

Complete resource pooling (CRP) condition is commonly used in queueing network scheduling literature (see e.g., Stolyar [26], Ye and Yao [28]). Roughly speaking, under this condition, the network service resources can completely be shared in certain way by all allowed users. There are different (but essentially equivalent) ways to describe CRP condition (see e.g., Stolyar [26], Ye and Yao [28]). Here, we adopt the way in Stolyar [26] to present our CRP condition. More precisely, let $\bar{\mathcal{R}}_v(i)$ for $i \in \mathcal{K}$ and $v \in \mathcal{V}$ denote the boundary of $\bar{\mathcal{R}}_v(i)$ in (3.10). Moreover, let $\bar{\mathcal{R}}_v^*(i)$ for $i \in \mathcal{K}$ and $v \in \mathcal{V}$ denote the outer (“north-east”) boundary of $\bar{\mathcal{R}}_v(i)$. Then, we have the following concepts.

A vector $\rho_v^*(i)$ with $i \in \mathcal{K}$ and $v \in \mathcal{V}$ is said to satisfy resource pooling (RP) condition if $\rho_v^*(i) \in \bar{\mathcal{R}}_v^*(i)$ and the outer normal vector ζ to $\bar{\mathcal{R}}_v(i)$ at $\rho_v^*(i)$ is unique (up to a scaling). In other words, the RP condition holds if $\rho_v^*(i)$ lies in the (relative) interior of one of curved facets of $\bar{\mathcal{R}}_v^*(i)$. Furthermore, $\rho_v^*(i)$ is said to satisfy the CRP condition if it satisfies the RP condition and all components of the corresponding normal vector ζ are strictly positive.

6.1.3. Heavy traffic condition

In addition, we introduce a sequence of independent Markov processes indexed by $r \in \mathcal{R}$, i.e., $\{\alpha^r(\cdot), r \in \mathcal{R}\}$. These systems all have the same basic structure as presented in the last section except the arrival rates $\lambda_{j_l}^r(i)$ and the holding time rates $\gamma^r(i)$ for all $i \in \mathcal{K}$, which may vary with $r \in \mathcal{R}$. Here, we suppose that they satisfy the heavy traffic condition:

$$r \left(\lambda_{j_l}^r(i) - \lambda_{j_l}(i) \right) m_{j_l}(i) \rightarrow \theta_{j_l}(i) \text{ as } r \rightarrow \infty, \quad \gamma^r(i) = \frac{\gamma(i)}{r^2}, \tag{6.3}$$

where, $\theta_{j_l}(i) \in R$ is some constant for each $i \in \mathcal{K}$. Moreover, we suppose that the nominal arrival rate $\lambda_{j_l}(i)$ is given by:

$$\lambda_{j_l}(i) m_{j_l}(i) \equiv \mu_{j_l} \rho_{j_l}(i), \tag{6.4}$$

and $\rho_{j_l}(i)$ in (6.4) for $j_l \in \mathcal{M}(i, v, t)$ with $l \in \{1, \dots, M_v\}$ is the nominal throughput determined by:

$$\rho_{j_l}(i) = \sum_{v \in \mathcal{V}(j_l)} \rho_{vj_l}(i) \text{ and } \rho_{vj_l}(i) = v_{vj_l} \bar{\rho}_{vj_l}(i), \tag{6.5}$$

with $\rho_{v \cdot}(i) \in \mathcal{O}_v(i)$ that is corresponding to the dimension M_v . In addition, $v_{v \cdot}$ and $\bar{\rho}_{v \cdot}(i)$ are a J_v -dimensional constant vector and a reference service rate vector, respectively, at service pool v , satisfying:

$$\sum_{j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}(v)} v_{j_l} = J_v, \quad v_{j_l} \geq 0 \text{ are constants for all } j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}(v), \tag{6.6}$$

$$\sum_{j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}(v)} \bar{\rho}_{vj_l}(i) = C_{U_v}(i) \text{ and } \bar{\rho}_{vj_l}(i) = \bar{\rho}_{vj_l}(i) \text{ for all } j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}(v). \tag{6.7}$$

Remark 6.1. By (3.11), $\bar{\rho}_{v \cdot}(i)$ for each $i \in \mathcal{K}$ and $v \in \mathcal{V}(j_l)$ can indeed be selected, which satisfy the second condition in (6.7). Thus, the CRP condition is true. Hence, the nominal throughput $\rho(i)$ in (6.4) can be determined. One simple example that satisfies these conditions is to take $v_{vj_l} = 1$ for all $j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}(v)$ and $v \in \mathcal{V}(j_l)$. Thus, the conditions in (6.4)-(6.7) mean that the system manager wishes to maximally and fairly allocate capacity to all users. Moreover, the design parameters $\lambda_{j_l}(i)$ for all $j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}$ and each $i \in \mathcal{K}$ can be determined by (6.4).

Next, we assume that the inter-arrival time associated with the k th arriving job batch to the system indexed by $r \in \mathcal{R}$ is given by:

$$u_{j_l}^r(k, i) = \frac{\hat{u}_{j_l}(k)}{\lambda_{j_l}^r(i)} \text{ for each } j_l \in \mathcal{M}(i, v, t) \cap \mathcal{J}, \quad k \in \{1, 2, \dots\}, \quad i \in \mathcal{K}, \tag{6.8}$$

where the $\hat{u}_{j_l}(k)$ does not depend on r and i . Moreover, it has mean one and finite squared coefficient of variation $\alpha_{j_l}^2$. In addition, the number of packets, $w_{j_l}(k)$, and the packet length $v_{j_l}(k)$ are assumed not to change with r . Thus, it follows from the heavy traffic condition in (6.3) for the r th environmental state process $\alpha^r(\cdot)$ with $r \in \mathcal{R}$ that $\alpha^r(r^2 \cdot)$ and $\alpha(\cdot)$ equal to each other in distribution since they own the same generator matrix (see e.g., the definition in pages 384–388 of Resnick [24]). Therefore, under the sense of distribution, all of the systems indexed by $r \in \mathcal{R}$ in (3.1) has the same random environment over any time interval $[0, t]$.

6.2. Proof of Theorem 3.5

First, it follows from the second condition in (6.3) that the processes $\alpha^r(r^2 \cdot)$ for each $r \in \mathcal{R}$ and $\alpha(\cdot)$ are equal in distribution. Hence, without loss of generality, we can assume that:

$$\alpha^r(r^2 t) = \alpha(t) \quad \text{for each } r \in \mathcal{R} \quad \text{and } t \in [0, \infty). \tag{6.9}$$

Thus, for each $j \in \mathcal{J}$, $r \in \mathcal{R}$ and by the radial homogeneity of $\Lambda(pq, i)$ of the policy in (6.2), we can define the fluid and diffusion scaled processes as follows,

$$E_j^r(\cdot) \equiv A_j^r(r^2 \cdot), \tag{6.10}$$

$$\bar{T}_j^r(\cdot) \equiv \int_0^\cdot \Lambda_j(\bar{P}^r(s)\bar{Q}^r(s), \alpha(s), s) ds = \frac{1}{r^2} T_j^r(r^2 \cdot), \tag{6.11}$$

$$\bar{Q}_j^r(t) \equiv \frac{1}{r^2} Q_j^r(r^2 t), \tag{6.12}$$

$$\bar{P}_j^r(t) = f_j(\bar{Q}_j^r(t), \alpha(t)), \tag{6.13}$$

$$\bar{E}_j^r(t) \equiv \frac{1}{r^2} E_j^r(t), \tag{6.14}$$

$$\bar{S}_j^r(t) \equiv \frac{1}{r^2} S_j^r(r^2 t). \tag{6.15}$$

Then, it follows from (2.7), (6.9), the assumptions among arrival and service processes that

$$\hat{Q}_j^r(\cdot) = \frac{1}{r} E_j^r(\cdot) - \frac{1}{r} S_j^r(\bar{T}_j^r(\cdot)). \tag{6.16}$$

Furthermore, for each $j \in \mathcal{J}$, let

$$\hat{E}^r(\cdot) = (\hat{E}_1^r(\cdot), \dots, \hat{E}_j^r(\cdot))' \quad \text{with } \hat{E}_j^r(\cdot) = \frac{1}{r} (A_j^r(r^2 \cdot) - r^2 \bar{\lambda}_j^r(\cdot)), \tag{6.17}$$

$$\hat{S}^r(\cdot) = (\hat{S}_1^r(\cdot), \dots, \hat{S}_j^r(\cdot))' \quad \text{with } \hat{S}_j^r(\cdot) = \frac{1}{r} (S_j^r(r^2 \cdot) - \mu_j r^2 \cdot), \tag{6.18}$$

where

$$\bar{\lambda}_j^r(\cdot) \equiv \int_0^\cdot m_j(\alpha(s), s) \lambda_j^r(\alpha(s), s) ds = \frac{1}{r^2} \int_0^{r^2 \cdot} m_j(\alpha^r(s), r^2 s) \lambda_j^r(\alpha^r(s), r^2 s) ds. \tag{6.19}$$

For convenience, we define

$$\bar{\lambda}^r(\cdot) = (\bar{\lambda}_1^r(\cdot), \dots, \bar{\lambda}_j^r(\cdot))'. \tag{6.20}$$

In addition, we let $\bar{Q}^r(\cdot)$, $\bar{E}^r(\cdot)$, $\bar{S}^r(\cdot)$, and $\bar{T}^r(\cdot)$ be the associated vector processes. Then, for the processes in (6.10)-(6.16), we define the corresponding fluid limit related processes,

$$\bar{Q}_j(t) = \bar{Q}_j(0) + \bar{\lambda}_j(t, \zeta_t(\cdot)) - \mu_j \bar{T}_j(t) \quad \text{for each } j \in \mathcal{J}, \tag{6.21}$$

where $\zeta_t(\cdot)$ denotes a process depending on the external environment, i.e.,

$$\bar{\lambda}(t) = (\bar{\lambda}_1(t), \dots, \bar{\lambda}_J(t))', \quad \bar{\lambda}_j(t) \equiv \int_0^t m_j \lambda_j(\alpha(s), s) ds. \tag{6.22}$$

Furthermore, we have that

$$\bar{T}_j(t) = \int_0^t \bar{\Lambda}_j(\bar{P}(s)\bar{Q}(s), \alpha(s), s) ds, \tag{6.23}$$

$$\bar{P}(t) = f(\bar{Q}(t), \alpha(t)), \tag{6.24}$$

where for each $i \in \mathcal{K}$ and $t \in [0, \infty)$, we have that,

$$\bar{\Lambda}_j(pq, i, t) = \begin{cases} \Lambda_j(pq, i, t) & \text{if } q_j > 0, j \in \bigcup_{v \in \mathcal{V}} \mathcal{M}(i, v, t), \\ \rho_j(i, t) & \text{if } q_j > 0, j \notin \bigcup_{v \in \mathcal{V}} \mathcal{M}(i, v, t), \\ \rho_j(i, t) & \text{if } q_j = 0. \end{cases} \tag{6.25}$$

Then, we have the following lemma concerning the weak convergence to a stochastic fluid limit process under our game-competition based dynamic pricing and scheduling strategy.

Lemma 6.2. *Assume that the initial queue length $\bar{Q}^r(0) \Rightarrow \bar{Q}(0)$ along $r \in \mathcal{R}$. Then, the joint convergence in distribution along a subsequence of \mathcal{R} is true under our game-competition based dynamic pricing and scheduling strategy in (3.20) and (3.28) with the conditions required by Theorem 3.5,*

$$(\bar{E}^r(\cdot), \bar{S}^r(\cdot), \bar{T}^r(\cdot), \bar{Q}^r(\cdot)) \Rightarrow (\bar{E}(\cdot), \bar{S}(\cdot), \bar{T}(\cdot), \bar{Q}(\cdot)). \tag{6.26}$$

In addition, if $\bar{Q}(0) = 0$, the convergence is true along the whole \mathcal{R} and the limit satisfies:

$$\bar{E}(\cdot) = \bar{\lambda}(\cdot), \quad \bar{S}(\cdot) = \mu(\cdot), \quad \bar{T}(\cdot) = \bar{c}(\cdot), \quad \bar{Q}(\cdot) = 0, \tag{6.27}$$

where $\bar{\lambda}(\cdot)$ is defined in (6.22), $\mu(\cdot) \equiv (\mu_1, \dots, \mu_J)'$, and $\bar{c}(\cdot)$ is defined by:

$$\bar{c}(t) = (\bar{c}_1(t), \dots, \bar{c}_J(t))' \quad \text{and} \quad \bar{c}_j(t) \equiv \int_0^t \rho_j(\alpha(s), s) ds \quad \text{for each } j \in \mathcal{J}. \tag{6.28}$$

Proof. First, by the proof of Lemma 1 in Dai [6] and the implicit function theorem, we can show that the pricing function f constructed through (3.22), (3.20), and (3.28) can be assumed to be Lipschitz continuous. Then, by extending the proof of Lemma 3 in Dai [6] and under the conditions in (6.1)-(6.2) and the just illustrated Lipschitz continuity for f , we know that, if $\Lambda(pq, i) \in F_{\mathcal{Q}}(i)$ for each $i \in \mathcal{K}$ is a given utility based 2-stage game-theoretic policy corresponding to the game problem in (3.20) and $\{p^l q^l, l \in \mathcal{R}\}$ is a sequence of valued queue lengths, which satisfies $p^l q^l \rightarrow pq \in R_+^J$ as $l \rightarrow \infty$. Then, for each $j \in \mathcal{J} \setminus \mathcal{Q}(q)$ and $v \in \mathcal{V}(j)$, we have that:

$$\Lambda_{vj}(p^l q^l, i) \rightarrow \Lambda_{vj}(pq, i) \quad \text{as } l \rightarrow \infty. \tag{6.29}$$

Second, due to the proof of Lemma 7 in Dai [6], we only need to prove that a weak fluid limit on the RHS of (6.26) satisfies (6.28). In doing so, we suppose that the weak fluid limit on the RHS of (6.26)

corresponds to a subsequence of the RHS of (6.26), which is indexed by $r_l \in \mathcal{R}$ with $l \in \{1, 2, \dots\}$. Furthermore, it follows from (6.11), (2.11), and the discussion in the proof of Lemma 7 of Dai [6] that the fluid limit process on the right-hand side of (6.26) is uniformly Lipschitz continuous almost surely. Thus, our discussion can focus on a fixed sample path and each regular point $t > 0$ over an interval (τ_{n-1}, τ_n) with $n \in \{1, 2, \dots\}$ for \bar{T}_j with $j \in \mathcal{J}$. More precisely, it follows from (6.21) that \bar{Q} is differential at t and satisfies:

$$\frac{d\bar{Q}_j(t)}{dt} = m_j \lambda_j(\alpha(t), t) - \mu_j \frac{d\bar{T}_j(t)}{dt} \tag{6.30}$$

for each $j \in \mathcal{J}$. If $\bar{Q}_j(t) = 0$ for some $j \in \mathcal{J}$, then it follows from $\bar{Q}_j(\cdot) \geq 0$ that

$$\frac{d\bar{Q}_j(t)}{dt} = 0 \text{ which implies that } \frac{d\bar{T}_j(t)}{dt} = \frac{m_j \lambda_j(\alpha(t), t)}{\mu_j} = \rho_j(\alpha(t), t). \tag{6.31}$$

If $\bar{Q}_j(t) > 0$ for the $j \in \mathcal{J}$, there is a finite interval $(a, b) \in [0, \infty)$ containing t in it such that $\bar{Q}_j(s) > 0$ for all $s \in (a, b)$ and hence we can take sufficiently small $\delta > 0$ such that $\bar{Q}_j(t + s) > 0$ with $s \in (0, \delta)$. Furthermore, by (2.8), $P_j(t + s) > 0$. Now, let r_l with $l \in \mathcal{R}$ be the subsequence \mathcal{R} and let $\delta_l \in (0, \delta]$ be a sequence such that $\delta_l \rightarrow 0$ as $l \rightarrow \infty$ while Λ_j determined by a same group of users over $(0, \delta_l]$. Then, it follows from (6.11) that:

$$\begin{aligned} & \left| \frac{1}{\delta_l} \left(\bar{T}_j^{r_l}(t + \delta_l) - \bar{T}_j^{r_l}(t) \right) - \Lambda_j(\bar{P}(t)\bar{Q}(t), \alpha(t), t) \right| \tag{6.32} \\ & \leq \frac{1}{\delta_l} \int_0^{\delta_l} \left| \Lambda_j(\bar{P}^{r_l}(t + s)\bar{Q}^{r_l}(t + s), \alpha(t + s), t + s) \right. \\ & \quad \left. - \Lambda_j(\bar{P}(t + s)\bar{Q}(t + s), \alpha(t + s), t + s) \right| ds \\ & \quad + \frac{1}{\delta_l} \int_0^{\delta_l} \left| \Lambda_j(\bar{P}(t + s)\bar{Q}(t + s), \alpha(t + s), t + s) - \Lambda_j(\bar{P}(t)\bar{Q}(t), \alpha(t), t) \right| ds \\ & \rightarrow 0 \text{ as } l \rightarrow \infty, \end{aligned}$$

where, the last claim in (6.32) follows from the Lebesgue dominated convergence theorem, the right-continuity of $\alpha(\cdot)$, the Lipschitz continuity of $\bar{Q}(\cdot)$, and the fact in (6.29). Since t is a regular point of \bar{T} , it follows from (6.32) that:

$$\frac{d\bar{T}_j(t)}{dt} = \frac{d\bar{T}_j(t^+)}{dt} = \bar{\Lambda}_j(\bar{Q}(t), \alpha(t), t) \text{ for each } j \in \mathcal{J}, \tag{6.33}$$

which implies that the claims in (6.23)-(6.25) are true.

Along the line of the proofs for Lemma 4.2 in Dai [8], Lemma 4.1 in Dai [9], and Lemma 7 in Dai [6], it suffices to prove the claim that $\bar{Q}(\cdot) = 0$ in (6.27) holds for the purpose of our current paper. In fact, for each $i \in \mathcal{K}$ and $l \in \{1, \dots, M_v\}$, we define

$$\psi(pq, i) \equiv \sum_{v \in \mathcal{V}} \psi_v(pq, i) = \sum_{v \in \mathcal{V}} \sum_{j_l \in \mathcal{M}(i, v) \cap \mathcal{J}(v)} C_{v j_l} (p_{j_l} q_{j_l}, \rho_{v j_l}(i)). \tag{6.34}$$

Then, at each regular time $t \geq 0$ of $\bar{Q}(t)$ over time interval (τ_{n-1}, τ_n) with a given $n \in \{1, 2, \dots\}$, we have that:

$$\begin{aligned} & \frac{d\psi(\bar{P}(t)\bar{Q}(t), \alpha(t))}{dt} \tag{6.35} \\ &= \sum_{v \in \mathcal{V}} \sum_{j_i \in \mathcal{M}(i,v,t) \cap \mathcal{J}(v)} \left(\rho_{vj_i}(\alpha(t), t) - \Lambda_{vj_i}(\bar{P}(t)\bar{Q}(t), \alpha(t), t) \right) \\ & \quad \frac{\partial U_{vj_i}(\bar{P}(t)\bar{Q}_{j_i}(t), \rho_{vj_i}(\alpha(t), t))}{\partial \rho_{vj_i}(\alpha(t), t)} I_{\{\bar{P}_{j_i}(t)\bar{Q}_{j_i}(t) > 0\}} \\ & \leq 0. \end{aligned}$$

Note that, the second equality in (6.35) follows from the concavity of the utility functions and the fact that $\Lambda_{vj_i}(\bar{P}(t)\bar{Q}(t), \alpha(t), t)$ is the Pareto maximal Nash equilibrium policy to the utility-maximal game problem in (3.20) when the system is in a particular state. Thus, for any given $n \in \{0, 1, 2, \dots\}$ and each $t \in [\tau_n, \tau_{n+1})$,

$$\begin{aligned} 0 & \leq \psi(\bar{P}(t)\bar{Q}(t), \alpha(t)) \tag{6.36} \\ & \leq \psi(\bar{P}(\tau_n)\bar{Q}(\tau_n), \alpha(\tau_n)) \\ & = \sum_{v \in \mathcal{V}} \sum_{j_i \in \mathcal{M}(i,v,\tau_n) \cap \mathcal{J}(v)} \frac{1}{\mu_{j_i}} \int_0^{\bar{Q}_{j_i}(\tau_n)} \frac{\partial U_{vj_i}(\bar{P}(\tau_n)u, \rho_{vj_i}(\alpha(\tau_n)))}{\partial C_{vj_i}} du \\ & = \sum_{v \in \mathcal{V}} \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(\alpha(\tau_n)))}{dc_{vj_i}} \right) \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(\alpha(\tau_{n-1})))}{dc_{vj_i}} \right)^{-1} \psi_v(\bar{P}(\tau_n)\bar{Q}(\tau_n), \alpha(\tau_{n-1})) \\ & \dots \\ & \leq \sum_{v \in \mathcal{V}} \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(\alpha(\tau_n)))}{dc_{vj_i}} \right) \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(\alpha(\tau_0)))}{dc_{vj_i}} \right)^{-1} \psi_v(\bar{P}(0)\bar{Q}(0), \alpha(0)) \\ & \leq \kappa \psi(\bar{P}(0)\bar{Q}(0), \alpha(0)), \end{aligned}$$

where κ is a positive constant, i.e.,

$$\kappa = \max_{v \in \mathcal{V}} \max_{i,j \in \mathcal{K}} \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(i))}{dc_{vj_i}} \right) \left(\frac{d\psi_v(\bar{\rho}_{vj_i}(j))}{dc_{vj_i}} \right)^{-1}.$$

Then, by the fact in (6.36), we know that $\bar{Q}(t) = 0$ for all $t \geq 0$. Therefore, we complete the proof of the lemma. □

In the end, by considering a specific state $i \in \mathcal{K}$ and by the index way as used in the proof of Lemma 6.2, we can extend the proofs for Lemma 4.3 to Lemma 4.5 in Dai [8] to the current setting. Then, by using the results in these lemmas to the proof for Theorem 1 in [6], we can reach a proof for Theorem 3.5 of this paper. □

7. Conclusion

In this paper, we study 2-stage game-theoretic problem oriented 3-stage service policy computing, CNN based algorithm design, and simulation for a blockchained buffering system with federated learning. More precisely, based on the game-theoretic problem consisting of both “win-lose” and “win-win” 2-stage competitions, we derive a 3-stage dynamical service policy via a saddle point to a zero-sum

game problem and a Nash equilibrium point to a non-zero-sum game problem. This policy is concerning users-selection, dynamic pricing, and online rate resource allocation via stable digital currency for the system. The main focus is on the design and analysis of the joint 3-stage service policy for given queue/environment state dependent pricing and utility functions. The asymptotic optimality and fairness of this dynamic service policy is justified by diffusion modeling with approximation theory. A general CNN based policy computing algorithm flow chart along the line of the so-called *big model* framework is presented. Simulation case studies are conducted for the system with three users, where only two of the three users can be selected into the service by a zero-sum dual cost game competition policy at a time point. Then, the selected two users get into service and share the system rate service resource through a non-zero-sum dual cost game competition policy. Applications of our policy in the future blockchain based Internet (e.g., metaverse and web3.0) and supply chain finance are also briefly illustrated.

Acknowledgments. The project is funded by National Natural Science Foundation of China with Grant No. 11771006.

Competing interests. The author declares that he has no conflict of interest.

References

- [1] Applebaum, D. (2005). *Lévy Processes and Stochastic Calculus*. Cambridge: Cambridge University Press.
- [2] Ayaz, F., Sheng, Z., Tian, D., & Guan, Y.L. (2022). A blockchain based federated learning for message dissemination in vehicular networks. *IEEE Transactions on Vehicular Technology* 71(2): 1927–1940.
- [3] Bramson, M. (1998). State space collapse with application to heavy traffic limits for multiclass queueing networks. *Queueing Systems* 30(1-2): 89–148.
- [4] Buterin, V. (2013). Ethereum: a next-generation smart contract and decentralized application platform. <http://ethereum.org/ethereum.html>.
- [5] Choudhury, G.L., Mandelbaum, A., Reiman, M.I., & Whitt, W. (1997). Fluid and diffusion limits for queues in slowly changing environment. *Stochastic Models* 13(1): 121–146.
- [6] Dai, W. (2013). Optimal rate scheduling via utility-maximization for J -user MIMO Markov fading wireless channels with cooperation. *Operations Research* 61(6): 1450–1462.
- [7] Dai, W. (2018). A unified system of FB-SDEs with Levy jumps and double completely-S skew reflections. *Communications in Mathematical Sciences* 16(3): 659–704.
- [8] Dai, W. (2018). Platform modelling and scheduling game with multiple intelligent cloud-computing pools for big data. *Mathematical and Computer Modelling of Dynamical Systems* 24(5): 506–552.
- [9] Dai, W. (2019). Quantum-computing with AI & blockchain: modelling, fault tolerance and capacity scheduling. *Mathematical and Computer Modeling of Dynamical Systems* 25(6): 523–559.
- [10] Dai, W. (2022). Convolutional neural network based simulation and analysis for backward stochastic partial differential equations. *Computers and Mathematics With Applications* 119: 21–58.
- [11] Dai, W. (2022). Optimality policy computing for blockchain based smart contracts via federated learning. *Operational Research* 22: 5817–5844.
- [12] Dai, W. (2023). n -qubit operations on sphere and queueing scaling limits for programmable quantum computer. *Quantum Information Processing* 22(122): 1–42.
- [13] Dai, W. & Jiang, Q. (2007). Stochastic optimal control of ATO systems with batch arrivals via diffusion approximation. *Probability in the Engineering and Informational Sciences* 21(3): 477–495.
- [14] Demertzis, K., Iliadis, L., Pimenidis, E., Tziritas, N., Koziri, M., & Kikiras, P. (2021). Blockchain adaptive federated auto MetaLearning BigData and DevOps CyberSecurity architecture in Industry 4.0. In *Proceedings of the 22nd Engineering Applications of Neural Networks Conference*: Halkidiki, Greece. pp. 345–363.
- [15] Elwalid, A.I. & Mitra, D. (1991). Analysis and design of rate-based congestion control of high speed networks, I: Stochastic fluid models, access regulation. *Queueing Systems* 9(1-2): 29–64.
- [16] Iansiti, M. & Lakehani, K.R. (January–February 2017). The truth about Blockchain. *Harvard Business Review*.
- [17] Little, J. (1961). A proof of the queueing formula: $L = \lambda W$. *Operations Research* 9(3): 383–387.
- [18] Maker. (2019). Looking ahead: how to upgrade to multi-collateral Dai from single-collateral Dai. <https://blog.makerdao.com/looking-ahead-how-to-upgrade-to-multi-collateral-dai>.
- [19] Marchi, E. (1967). On the concept of saddle point in zero-sum two-person generalized games. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 9: 26–35.
- [20] Nakamoto, S. (2008). A peer-to-peer electronic cash system. metzdowd.com.
- [21] Nash, J.F. (1950). Equilibrium Points in N -person Games. *Proceedings of the National Academy of Sciences* 36(1): 48–49.

- [22] Qu, Y., Pokhrel, S.R., Garg, S., Gao, L., & Xiang, Y. (2021). A blockchain federated learning framework for cognitive computing in Industrial 4.0 networks. *IEEE Transactions on Industrial Informatics* 17(4): 1964–2973.
- [23] Rajan, D. & Visser, M. (2018). Quantum Blockchain using entanglement in time. <https://arxiv.org/abs/1804.05979>.
- [24] Resnick, S.I. (1992). *Adventures in Stochastic Processes*. Boston: Birkhäuser.
- [25] Rosen, J.R. (1965). Existence and uniqueness of equilibrium points for concave N -person games. *Econometrica* 33(3): 520–534.
- [26] Stolyar, A.L. (2004). MaxWeight scheduling in a generalized switch: state space collapse and workload minimization in heavy traffic. *The Annals of Applied Probability* 14(1): 1–53.
- [27] Wang, H.S. & Moayeri, N. (1995). Finite-state Markov channel – a useful model for radio communication channels. *IEEE Transactions on Vehicular Technology* 44(1): 163–171.
- [28] Ye, H. & Yao, D.D. (2008). Heavy traffic optimality of a stochastic network under utility-maximizing resource control. *Operations Research* 56(2): 453–470.