Inducing Optimal Scheduling with Selfish Users

Paul Enders[1], Anshul Gandhi[2], Varun Gupta[2]

Laurens Debo[3], Mor Harchol-Balter[2], Alan Scheller-Wolf[1]

[1] Tepper School of Business, Carnegie Mellon University,{penders,awolf}@andrew.cmu.edu
[2] School of Computer Science, Carnegie Mellon University,{anshulg,varun+,harchol+}@cs.cmu.edu
[3] Booth School of Business, University of Chicago,laurens.debo@chicagobooth.edu

It is well known that scheduling jobs according to the Shortest-Remaining-Processing-Time (SRPT) policy is optimal for minimizing mean response time in a single-server system with online arrivals. Unfortunately, SRPT scheduling requires users to reveal their job size (service requirement), which is not always realistic. This may be because some users are informed, but *selfish* (or rational): they know their job size but are willing to lie about it if that is to their advantage. Alternatively, users may be uninformed – they may genuinely not know their size. Complicating matters further, the system administrator may not be able to differentiate between users of these two types. This adds significant complications to the management of such systems, as threats that are credible to force informed users to reveal their size may be unfair to uninformed users. This situation is common in supercomputing environments, for example.

To cope with such a situation we develop a novel approach for inducing the informed users into self-scheduling themselves according to (an approximation of) SRPT while not, unfairly, penalizing the uninformed users. We achieve this by defining a game that users play, applying priority-boosting tokens to *portions* of their job. For the informed users, we characterize the features of a unique equilibrium self-scheduling policy. We prove that the equilibrium policy is a dominant strategy, i.e. no user has any incentive to follow any other strategy, regardless of other users' behavior. We show that the optimal strategy of uninformed users is highly complex and may lead to multiple, potentially unstable, equilibria. Then, via queueing-theoretic analysis, we evaluate the effectiveness of our game as a function of job size variability, load, and other system parameters. We find that our game results in a near approximation of SRPT scheduling for informed users when the appropriate number of tokens (which we can determine) is distributed.

Key words: Strategic Queueing, Scheduling, SRPT, Equilibrium Analysis, Dominant Strategy, Selfish/Rational Users

Area of review: Stochastic Models and Simulation

---

1. Introduction

We consider the problem of online scheduling of jobs on a single machine. In this setting it is well-known that always running the job with the Shortest-Remaining-Processing-Time (known as

SRPT scheduling), is optimal with respect to minimizing mean response time (Schrage 1968). A job's *response time* is the time from when it arrives until it completes. The optimality of SRPT holds regardless of the arrival times and sizes (service requirements) of jobs – i.e., it holds for any arrival sequence. Thus it is clear that the goal of any single machine scheduler concerned with response time should be to strive to implement SRPT scheduling.

What is less well-understood is what one should do when the job sizes aren't known. Within the domain of operating systems, unknown job sizes (e.g., Linux jobs) are typically handled by Round-Robin (RR) scheduling: jobs queue up to receive one small quantum of the processor, and then move to the back of the queue, where they wait again (Peterson and Silberschatz 2002). While RR has the advantage of not requiring knowledge of job size, in general it is nowhere near as effective as SRPT in minimizing mean response time (Bansal and Gamarnik 2006). When job sizes are unknown, but drawn from a specific type of distribution (decreasing failure rate), working on the job with the least attained service (referred to as Foreground Background, FB) is know to be optimal, (e.g. Yashkov 1978). However, FB is still outperformed by SRPT when job sizes are known.

In our setting, job sizes may be unknown because a user genuinely *does not know* the service requirement of his job (is *uninformed*), or because a user who *does know* his service requirement (is *informed*) is *selfish:* willing to lie about his job size so as to minimize his own response time. (For example, pretending that his job size is small so as to gain advantage under SRPT scheduling.) We assume that both types of users – uninformed & informed selfish – may be present. We are motivated by scheduling for supercomputing centers which often face uninformed and informed selfish users (Lee et al. 2004). This mixture of uninformed and selfish users differentiates us from the growing body of prior work dealing with scheduling under inexact job sizes (e.g., Lu et al. 2004a, Lu et al. 2004b, Wierman and Nuyens 2008). Within this context, we seek to schedule jobs as closely as possible to SRPT with the aim of minimizing mean response time (as supercomputing centers typically do).

One potential solution to our problem would be to just ask users for their job size, in the presence of a credible threat (stop processing a user's job if it exceeds its declared size). This would indeed lead to SRPT performance, for the informed users. But what about the uninformed users? They would have incentive to grossly overestimate their job sizes to avoid penalties. In fact, this occurs in practice: In supercomputer applications it is common to ask users for a bound on their job size and kill their job once this bound is exceeded. This leads to overestimation of job sizes, as documented in the literature (e.g. Mu'alem and Feitelson 2001, Chiang et al. 2002, Snell et al. 2002). We seek

a policy that doesn't penalize users for not knowing their job size, while simultaneously providing SRPT scheduling for the informed but selfish users.

Our solution is very different from those proposed in the past: The users – instead of the system administrator – schedule their own jobs, following rules that the system administrator devises. While some authors have investigated user self-scheduling, their solutions all involve the exchange of "real money" which has utility outside the system (and thus provides an ex-post credible threat to all users): Dolan (1978), Hassin and Haviv (1997), and Mendelson and Whang (1990) all use prices or Clark tax-based mechanisms. In many scheduling applications, e.g., scheduling at supercomputing centers, it is undesirable to use real money since processor-hours are already paid for, e.g. by the NSF (Chun et al. 2005). Thus we devise a system that does not rely on real money. A detailed discussion of this and other prior work is presented in §2.

The key idea in our solution is that each user schedules his own job by following the rules to a particular game we devise. The basic game setting (§3) is as follows: We assume that all jobs are comprised of a number of equal-sized quanta, which must be run sequentially, as shown in Figure 1. There are two queues, one labeled high priority, and the other labeled low priority. Jobs in the high-priority queue are always allowed to run before those in the low-priority queue. We give each user – uninformed & informed – some number, say $D = 2$, of tokens. Upon arrival to the system, and *without any knowledge of the system state*, the user must place these 2 tokens on two of the quanta comprising his job. Quanta that have a token placed on them are allowed access to the high-priority queue, whereas all other quanta are processed in the low-priority queue. In this setting, it will turn out that if each user behaves selfishly, trying to minimize only his own response time, then the strongly dominant strategy for informed users results in scheduling themselves close to SRPT irrespective of the uninformed users' strategy. In fact the resulting schedule for informed users (§4.1) will be arbitrarily close to SRPT if we increase the complexity of the game appropriately. *Thus even with users behaving selfishly, we can induce SRPT scheduling, without eliciting their job sizes or involving real money.*

Due to their lack of knowledge uninformed users face a much more complex problem; we show the may have multiple, potentially unstable, equilibria (§4.2).

We then develop a queueing-theoretic algorithm to calculate the equilibrium response time of our game under the assumption of Poisson arrivals, in §5.1. This algorithm requires a strategy for the uninformed users. Since the equilibrium behavior of the uninformed users is complex, we assume they spend their tokens as soon as possible: This is least wasteful of tokens, and also is in

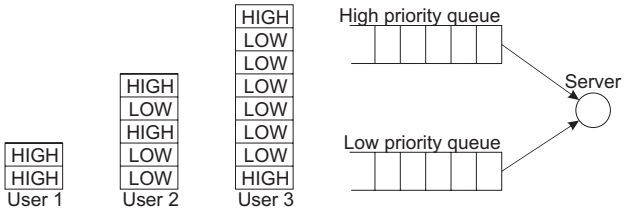


Figure 1 Illustration of 3 users playing the scheduling game. The first user's job consists of 2 quanta, the second user's job consists of 5 quanta and the third of 8 quanta. Users are each given 2 tokens which they place on 2 quanta of their choice, making these "high-priority" quanta. Jobs queue up at the 2-queue system, shown on the right, where they receive only one quantum of service before returning to the back of the line to queue up again.

accordance with FB scheduling. We refer to this strategy as Immediate Gratification (IG)[1]. Given these strategies for informed and uninformed (IG) customers, our algorithm enables us to answer several relevant questions in §5.2, such as how many tokens one should distribute so as to minimize overall mean response time, which factors determine this optimal number of tokens, and how do they affect overall system performance? We find that our game, under optimal token endowment, typically improves performance for both uninformed and informed customers. Specifically, for the informed customers it significantly reduces the gap between the ideal SRPT and Round-Robin (or Processor Sharing, PS, a limiting type of RR) performance. We provide insight into the optimal token selection by identifying the different *pains* and *gains* tokens cause to informed and uninformed users. We also find that in general, the optimal number of tokens increases as job sizes become more variable, the system load increases, or as the fraction of uninformed users decreases.

In §5.3 we use our algorithm to briefly revisit the problem of optimal token placement for uninformed customers. We demonstrate numerically that their optimal strategy may be very complex; for example when users have distributional knowledge about their size, their best response may be to place tokens discontiguously.

As supercomputing centers typically operate as a multiserver system, we consider extending our game to the multiserver setting in §6. We present our conclusions in §7.

2. Prior work

The present work lies at the intersection of the rich work on *size-based scheduling* (i.e. attempting to achieve SRPT via prioritization of small jobs), and the vast amount of work on scheduling with strategic users. Below, we briefly summarize some of the key results in both of these areas.

[1] This assumption is further justified by the fact that if the equilibrium strategy is hard to compute, it is unlikely that uninformed users adopt the equilibrium behavior, and thus may prefer a simple heuristic such as IG.

Centralized Scheduling Policies with Global Knowledge There is a large body of work on analysis of single server scheduling policies where a centralized scheduler takes all the decisions. Examples of such scheduling policies are SRPT, FB, RR, PS (in which the processor capacity is evenly split among all users), and two-level PS (the processor is equally split between users that have attained less than a threshold amount of service). The optimality of SRPT for minimizing the mean response time was proven as early as 1968 by Schrage. However, SRPT assumes that the exact service times of all jobs are known to the scheduler. The performance of n-level SRPT, where one can only differentiate between n levels of job size, is compared to full SRPT via implementation in Harchol-Balter et al. (2003) and via analysis in Wierman and Nuyens (2008). Both works conclude that 2-level SRPT or 3-level SRPT already achieves most of the performance benefits of full-SRPT.

For the case where the scheduler does not know the job sizes, Yashkov (1978) (in Russian) proves that FB is optimal when the job size distribution belongs to the class of decreasing hazard rate (DHR) distributions. Righter and Shantikumar (1989) show that under DHR, FB is optimal in the stochastic sense, i.e., the tail of the distribution of number of jobs in queue is minimized, for general arrival processes. Feng and Misra (2003) show that under Poisson arrival, FB also minimizes the mean slowdown (also known as stretch, and defined to be the ratio of the response time to the size) of the jobs. Age-based scheduling has also been considered in the context of network flows. In particular, a two-level processor sharing (TLPS) model has been considered by Aalto et al. (2004) and Avrachenkov et al. (2007). Avrachenkov et al. (2007) present analytic results for mean delay under TLPS, Aalto et al. (2004) show that PS is better than TLPS under an increasing hazard rate.

Strategic Users - Equilibrium Strategies We now turn to the case where users possess private information, which the scheduler is not aware of. Most of the literature focuses on *waiting costs* as private information, and the goal of the scheduler is to design pricing schemes (also known as mechanisms) so as to elicit the users' private information and thereby maximize its revenue, or the social utility. Users then join the system based on the cost of service and the queue lengths the observe. As joining decisions of one user may impact the delay and thus costs of other users, with strategic users, equilibrium strategies must be characterized: Users select a strategy that maximizes their expected utility, assuming that other users likewise maximize their own expected utilities.

Hassin and Haviv (1997) introduce a queueing mechanism with a high and a low-priority queue. Upon arrival, a user must select to which priority queue to route his job; users pay a premium *price* to join the high-priority queue. Hassin and Haviv show that threshold policies of the type "Buy priority when, upon arrival, you see more than n jobs in the system" are equilibrium strategies

when arriving users observe the system state and it is more expensive to join the high-priority queue than joining the low-priority queue. Adiri and Yechiali (1974) consider a more general setting than Hassin and Haviv. An iterative process is suggested in which the service station sets prices for priorities and then users decide which priority to buy. The service station may then change the prices to maximize its income. Again, threshold policies are shown to be equilibria in this general case. In both of these works, the job sizes of users are independent and identically distributed as an exponential random variable, and the users do not know their exact job sizes. However, in this paper, we are concerned with general job size distributions where some users know their exact job sizes.

Dolan (1978) considers the case in where users reveal their true size to the scheduler, but possess private knowledge about their delay costs and may not truthfully reveal this information. All users are assumed to be present at $t = 0$. Dolan proposes a Clarke tax based mechanism: It is shown that a user can maximize his utility by revealing his true delay cost if the price that the system charges equals the marginal delay cost imposed on other users. Some approximations are given for online arrivals.

Mendelson and Whang (1990) consider a decentralized capacity allocation system via prices where strategic users have private information about their delay costs and their mean job size (users belong to one of several classes, where the jobs of each class are independent and identically distributed as exponential random variable with some mean), and in which users do not observe the system state upon arrival. The authors design a pricing mechanism in which users that have truly high delay costs and larger mean job size need to pay a higher price to obtain priority. This incentive compatible mechanism also optimizes the net value of the system. There is much other related work; an excellent review can be found in Hassin and Haviv (2003). As opposed to the work of Dolan (1978) and Mendelson and Whang (1990), in our setting, some users know their exact job size, while some users may know their job size distribution, neither of which is revealed to the scheduler.

In most of the above research, each user arrives with one, indivisible job to the service facility. Furthermore, and perhaps more importantly, the focus is on eliciting heterogeneous private delay costs. Hence, prices (or "real money") are part of the coordination mechanism designed by the system administrator. In our context, a job is divisible into smaller pieces that must be processed in sequence. This provides more opportunities to minimize the expected delay and expands the space of implementable policies, enabling preemption, Round Robin, and PS. In addition, we devise a mechanism in which the exchange of real money is not required. Such an exchange is

often not realistic (as also noted by Chun et al. 2005) as the processing hours are often not being sold, but granted. For such a situation we introduce "tokens" as a tool for eliciting users' private information.

3. The Game Set-up

In this section, we formally introduce the set-up of our game: The set of players (or users), their job sizes and the arrival stream. Then we describe the strategy space for the players, as well as their utility. To establish the players' strategies under the game set-up, we define the conditions for the players' strategies to be an equilibrium. Finally, we introduce the notion of a strongly dominant strategy. We have chosen to present a self-contained measure-theoretic definition of the game so that a reader not familiar with game theory can easily follow our treatment.

We first begin with an informal description of our game to provide intuition behind our formal definition. We consider N players, each of whom has a job which requires processing. The size of the job of each player is an integral multiple of a minimum processing unit, called a *quantum*. Further, the job size of each player is a random variable with an associated distribution. The job size distributions of the N players need not be equal, but are assumed common knowledge among the players. An arrival sequence, $\boldsymbol{\omega}$, of the N players is generated according to some stochastic process. The measure induced on the path space by this stochastic process is also assumed common knowledge. The arrival sequence gives the arrival instant of each player into the queueing system which is operated by a system administrator.

The role of the system administrator is to enforce the rules of the games as follows: he maintains one high-priority and one low-priority queue, always depleting the high-priority queue before proceeding with the low-priority queue. Upon the arrival of a new player, the system administrator distributes D tokens to him; each token can be used to route one quantum of the player's job to the high-priority queue. For the service of his j^{th} quantum, the player joins the back of the high or the low-priority queue – depending on the token placement – only after his $(j-1)^{st}$ quantum has been serviced.

When the ith player arrives into the system, he observes some partial information, $f_i(\boldsymbol{\omega})$. It is important to note that this partial information is a function only of the arrival instants of the N players (we give examples of $f_i(\boldsymbol{\omega})$ below). Based on this partial information alone, player i designates at most D quanta of his job to receive high priority. The delay of the ith player depends on the arrival sequence realized, the strategies adopted by the other players, and the job size realizations of all the players. Thus player i chooses his strategy so as to minimize his expected delay, while considering the strategic behavior of the other players. We make this precise below.

Players: There are N players, each of whom has a job of certain size that needs processing. Player i's job is composed of a number of quanta – a discrete random variable $S_i \in \mathbb{N}$ – that each take one unit of time on the server. We denote the distribution of S_i by F_i. Let $\mathbf{F} = \{F_1, \ldots, F_N\}$; for ease of analytical exposition, we assume here that $\mathbf{F}$ is common knowledge. This assumption is *not* required for our analytical results of §4. We call a player i *informed* if F_i is deterministic, otherwise, we call player i *uninformed*. We assume that the job sizes of the players are generated independent of each other (and of the arrival sequence). We will also use $\mathbf{F}$ to denote the measure induced on $\mathbb{N}^N$ by generating job sizes independently of each other according to $\mathbf{F}$.

Arrival process: We denote the space of all arrival sequences of N players by $\Omega = \{\mathbb{R}^+\}^N$. For an $\boldsymbol{\omega} = \{\omega_1, \omega_2, \ldots, \omega_N\} \in \Omega$, ω_i denotes the arrival time of player i on the sample path $\boldsymbol{\omega}$. The measure on the space Ω according to which the sample paths are generated is denoted by $\mathcal{M}$, which is common knowledge.

Signal functions: To each player i, we associate a *signal function*, $f_i : \Omega \to X$ where X is some topological space with an associated Borel σ-algebra $\mathcal{X}$. Let $\mathcal{A}_i$ be the σ-algebra *induced* on Ω by the function $f_i(\cdot)$. Let $\mathbf{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$, and is assumed common knowledge. We will use the signal function $f_i(\omega)$ to restrict the information about the actual realization of the sample path revealed to the ith player. For example, if each player only observes his arrival instant, then $f_i(\boldsymbol{\omega}) = \omega_i$, with $X = \mathbb{R}$ and $\mathcal{X}$ the Borel σ-algebra on $\mathbb{R}$. Another possible signal function is $f_i(\boldsymbol{\omega}) = \sum_j \mathbf{1}_{\omega_j < \omega_i}$, which denotes the number of arrivals into the system before player i. Note that the notion of informed and uninformed players is based entirely on the job size distribution of the respective players, and has nothing to do with the partial information revealed by the signal function.

As a concrete example, let $N = 3$, and let $\Omega = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_3\}$ consist of the following three arrival sequences, each with a probability of $\frac{1}{3}$:

$$\boldsymbol{\omega}_1 = \{0, 1, 2\}, \quad \boldsymbol{\omega}_2 = \{2, 1, 0\}, \quad \boldsymbol{\omega}_3 = \{1, 2, 0\}.$$

That is, on arrival sequence $\boldsymbol{\omega}_2$, the arrival times of players 1, 2 and 3 are $t = 2, 1$ and 0, respectively. Let the arrival sequence generated be $\boldsymbol{\omega}_2$ and let the signal player for ith player be $f_i(\boldsymbol{\omega}) = \omega_i$. That is, the players only observe their own arrival instants. In this case, $\mathcal{A}_1 = 2^\Omega$, while $\mathcal{A}_2 = \{\emptyset, \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2\}, \{\boldsymbol{\omega}_3\}, \Omega\}$, and $\mathcal{A}_3 = \{\emptyset, \{\boldsymbol{\omega}_1\}, \{\boldsymbol{\omega}_2, \boldsymbol{\omega}_3\}, \Omega\}$. On the realization of $\boldsymbol{\omega}_2$, player 1 can perfectly deduce the exact realization of sample path (but not the job sizes) based on the information revealed to him. However, player 2 can not distinguish between $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$, while player 3 cannot distinguish between $\boldsymbol{\omega}_2$ and $\boldsymbol{\omega}_3$.

Strategy space: The space of pure strategies for each player is $\mathcal{S} = \mathbb{N}^D$, where $\mathbf{s} = \{s_1, s_2, \ldots, s_D\} \in \mathcal{S}$ specifies which of the D quanta the player assigns priority-boosting tokens to (if $S_i < D$, at most S_i tokens can be used). The space of mixed strategies of a player is denoted by $m(\mathcal{S})$, the space of all probability measures on $\mathcal{S}$. Player i's strategy is a $\mathcal{A}_i$-measurable mapping from Ω to $m(\mathcal{S})$. We denote this mapping by $p_i(\cdot)$. The restriction that p_i be $\mathcal{A}_i$-measurable enforces the requirement that the strategy of player i is only a function of the partial information about the arrival sequence revealed to the player, and not of the strategies of other players. Let $\boldsymbol{p} = \{p_1, \ldots, p_N\}$ be the *strategy profile* of the N players.

The queueing system: The queueing system consists of two queues: high-priority and low-priority. The server always processes one quantum of a job at a time, and depletes the high-priority queue before proceeding with the low-priority queue, without preemption. When the server finishes service of the j^{th} quantum of a player i, the player joins the back of the high or the low-priority queue – depending on the token placement on his $(j+1)$st quantum. The server then picks the next job to serve from the head of the high-priority queue if it is non-empty; otherwise the next job to be served is picked from the head of the low-priority queue.

Utility: For a given sample path ω, and a strategy profile $\boldsymbol{p} = \{p_1, \ldots, p_N\}$ for the N players, define $\mathbb{E}[W_i|\omega, \boldsymbol{p}]$ to be the the mean delay of the ith user on sample path $\boldsymbol{\omega}$ of arrival instants under the product measure $\mathbf{F} \times \boldsymbol{p}$ on job sizes and token placements of the N players. Let $W^i_{\boldsymbol{p}}(\cdot) = \mathbb{E}[W_i|\cdot, \boldsymbol{p}]$. Therefore, $W^i_{\boldsymbol{p}} : \Omega \to \mathbb{R}^+$ is a random variable from Ω to $\mathbb{R}^+$.

In Game theory parlance, we have defined the following *simultaneous game*: At time $t = 0$, Nature generates a sample path $\boldsymbol{\omega}$ of arrival instants. The information sets of all players are then created based on their individual signal functions. The information set, I_i, for player i is given by $I_i = \{\boldsymbol{\omega}' : f_i(\boldsymbol{\omega}') = f_i(\boldsymbol{\omega})\}$. All the players are then required to simultaneously place their D tokens on D quanta. Finally, Nature moves to resolve the remaining randomness in job sizes. The queueing system is simulated and player i receives as penalty the mean delay experienced by his job.

Definition 1 (Best Response). For a player i, we call the $\mathcal{A}_i$-measurable mapping p_i^* a best response to the set of strategies $\boldsymbol{p}_{-i} = \{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_N\}$ (where each p_j is $\mathcal{A}_j$-measurable) if

$$p_i^* = BR(\boldsymbol{p}_{-i}) \triangleq \arg\min_{p_i} \mathbb{E}\Big[W^i_{p_i, \boldsymbol{p}_{-i}} | \mathcal{A}_i\Big].$$

Definition 2 (Equilibrium). A strategy vector $\boldsymbol{p}^* = \{p_1^*, \ldots, p_N^*\}$ is called an equilibrium if for each i, $p_i^* = BR(\boldsymbol{p}^*_{-i})$.

Definition 3 (Strongly Dominant Strategy). If $\hat{p}_i \in \arg\min_{p_i} \mathbb{E}\big[W_i|\boldsymbol{\omega}, (p_i, \boldsymbol{p}_{-i})\big]$, for any $\boldsymbol{p}_{-i}$ and $\boldsymbol{\omega}$, then $\hat{p}_i$ is called a strongly dominant strategy for player i.

A strongly dominant strategy minimizes a player's mean delay, irrespective of the strategy of all other players, *on every sample path*. If a strongly dominant strategy exists for each player, then the corresponding strategy profile is an equilibrium for a game with any measure $\mathcal{M}$ on arrival sample paths; it follows immediately that a strongly dominant strategy is a best response to any strategy and hence an equilibrium according to Definition 2.

4. Analysis of the Game

Having defined the game set-up, we now analyze the equilibrium behavior of users. In §4.1, we characterize a strongly dominant strategy for informed users. Since the informed users have a strongly dominant strategy, that is, an optimal strategy irrespective of the strategy adopted by uninformed users, this allows us to decouple the analysis of uninformed users and informed users. In §4.2 we then address the question of finding equilibrium strategies for uninformed users in the presence of informed users playing their dominant strategy, and find that even in the simplest system, the equilibrium behavior of uninformed users is quite complex.

4.1. Equilibrium analysis for informed users

In this section we characterize the strongly dominant strategy for informed users. We will, in fact, prove a stronger result: We will show that a single pure strategy minimizes an informed user's delay on any given arrival sequence, for any deterministic assignment of job sizes to other users, for any strategy profile for other users, and even if the numbers of tokens given to different users are not the same. This in turn proves that the only requirement for our proposed strategy to be dominant is that the arrival sequence, the job sizes, the token allotment to the users, and the actions (token placement) of other users are not influenced by the action of the tagged informed user. These conditions are certainly met in the game set-up described in §3. In addition some assumptions of §3, such as the job size distributions being common knowledge, can be relaxed without influencing the result presented in this section.

We begin by defining a specific pure strategy for informed users:

Definition 4 (Delayed Gratification). For an informed player with job size n and D tokens, Delayed Gratification (DG) is the pure strategy which chooses high priority for the last D quanta, i.e.

$$p(\{(n-D+1)^+,(n-D)^+,\ldots,(n-1)^+,n\}) = 1.$$

Theorem 1. *DG is a strongly dominant strategy for informed users.*

Before proceeding to the proof of Theorem 1, we will illustrate the intuition behind it on a small example. We first define another pure strategy for all users, which is in some sense the exact opposite of DG.

Definition 5 (Immediate Gratification). Immediate Gratification (IG) is the pure strategy which chooses high priority for the first D quanta, i.e.

$$p(\{1,2,\ldots,\min(n,D)\}) = 1.$$

Consider two users A and B who arrive at time $t^A = 0$ and $t^B = \epsilon$ $(0 < \epsilon < 1)$, respectively. Both users have a job of size two quanta and $D = 1$ token. The system is empty at $t = 0^-$ and there are no further arrivals. There are only two possible pure strategies for each user: place the token on the first quantum (IG) or on the second (DG). The delay (cost) matrix for this example is shown in Figure 2:

		User B: DG	User B: IG
User A	DG	$(2, 4-\epsilon)$	$(3, 4-\epsilon)$
	IG	$(4, 3-\epsilon)$	$(3, 4-\epsilon)$

Figure 2 The cost matrix for the two player example. An entry (a,b) denotes a cost of a to user A, and cost of b to user B under the indicated strategy profile for each user.

We see that DG is the dominant strategy for both users. The benefits of delayed gratification are two fold: *(i)* minimize overtake by later-arriving users in the L queue, as seen by comparing user A's strategies when user B adopts DG, and *(ii)* possibly overtake the L quanta of earlier-arriving users who are not following DG, as seen by comparing user B's strategies when user A adopts IG.

Proof of Theorem 1 Recall that proving that DG is a dominant strategy for informed users entails proving that on every sample path of arrivals, the response time of a tagged informed user is minimized by adopting DG, irrespective of the strategies of other users, under the assumption that the strategies of users are independent of the actions taken by other users.

Consider a fixed sample path of arrivals and a tagged arrival U of size n. We will represent a pure strategy for user U as a string in $\{H, L\}^n$, where the ith character is H iff under the pure strategy, the ith quantum is assigned a token. Consider two possible pure strategies $\mathbf{P}_{HL}$ and $\mathbf{P}_{LH}$ for U. The strategies $\mathbf{P}_{HL}$ and $\mathbf{P}_{LH}$ are identical except that one consecutive HL pair in $\mathbf{P}_{HL}$ is flipped to LH to obtain $\mathbf{P}_{LH}$. We will prove that the response time of U under strategy $\mathbf{P}_{LH}$ is at most the response time under strategy $\mathbf{P}_{HL}$. This will prove the theorem since we can keep flipping HL to LH until we obtain the strategy corresponding to DG.

Let k be the index at which $\mathbf{P}_{HL}$ and $\mathbf{P}_{LH}$ first differ. Let $t = 0^-$ be the time at which U finishes

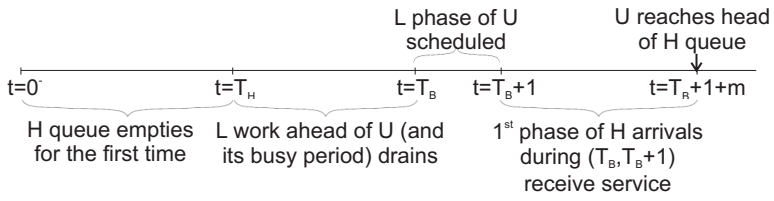


Figure 3 Illustration of the scheduling under $\mathbf{P}_{LH}$ strategy.

its $(k-1)$th phase and chooses to follow L under $\mathbf{P}_{LH}$ and H under $\mathbf{P}_{HL}$. Clearly, at $t=0^-$ the state of the system in both cases is identical. Let the state description at $t=0^-$ is given by:

$n_H =$ number of users in H queue (excluding U)

$n_L =$ number of users in L queue (excluding U)

$u_i^H \equiv$ user at the ith position in the H queue, $i \in \{1,\ldots,n_H\}$

$u_i^L \equiv$ user at the ith position in the L queue, $i \in \{1,\ldots,n_L\}$

$u^S \equiv$ user at the server

$\epsilon =$ excess of the user at server (this allows $k=1$)

$h_i^H =$ the length of the H streak (series of consecutive high-priority quanta) in u_i^H's strategy (including and following the H phase it is waiting on)

$h_i^L =$ the length of the H streak in u_i^L's strategy following the L phase its waiting on

$h^S =$ the length of the H streak in u^S's strategy following the phase its serving at $t=0^+$

Define $B^H(W,t)$ to be the length of the busy period initiated at time t by (high) work of length W, which ends when the H queue becomes empty. That is, the busy period is the time to serve W, and the initial H streak of all the arrivals during the busy period.

Define $N_A^H(t_1,t_2)$ to be the number of external arrivals to the H queue in the time interval $[t_1,t_2)$, and $N_A^L(t_1,t_2)$ to be the number of external arrivals to the L queue in the time interval $[t_1,t_2)$.

Let $T_H = B^H(\epsilon + h^S + \sum_{i=1}^{n_H} h_i^H, 0)$

Let $T_B = B^H(\epsilon + h^S + \sum_{i=1}^{n_H} h_i^H + \sum_{j=1}^{n_L}(1+h_j^L), T_H)$.

Let $m = N_A^H(T_B, T_B+1)$.

Note that $t = T_H^+$ is the time at which under $\mathbf{P}_{LH}$, the first low-priority job starts receiving service. Also, $t=T_B^+$ is the time at which under $\mathbf{P}_{LH}$, U starts receiving service for the L phase of interest. Further, $t = T_B + 1 + m$ is the time at which U starts receiving service for its following H phase under $\mathbf{P}_{LH}$ strategy. This is illustrated in Figure 3.

The following proposition compares the state of the system at $t = T_B + 1 + m$ under strategies $\mathbf{P}_{LH}$ and $\mathbf{P}_{HL}$:

PROPOSITION 1. *Under strategies $\boldsymbol{P}_{HL}$ and $\boldsymbol{P}_{LH}$, at time $t = T_B + 1 + m$:*

1. *Under $\boldsymbol{P}_{LH}$, U is at the head of the H queue.*
2. *Under $\boldsymbol{P}_{HL}$, U is in the L queue.*
3. *The set of users in the H queues in the two systems are identical. Further, these users are waiting on the same phases in the two systems.*
4. *The set of users in the L queues in the two systems are identical. Further, these users are waiting on the same phase in the two systems.*

Using Proposition 1, it is easy to complete the proof of the Theorem. At $t = T_B + m + 1$, the state of the systems is identical except that under $\mathbf{P}_{LH}$, U is at the head of the H queue and under $\mathbf{P}_{HL}$, U is in the L queue (the order of the other users within the queues is immaterial). Further, the arrival sequence to the two systems as well as the subsequent strategy followed by U is also identical in the two systems. This is true since in our game, users choose their strategies before observing the system state and hence user U's actions do not influence the strategies of users arriving after U. Therefore, under $\mathbf{P}_{LH}$, U leaves the system no later than under $\mathbf{P}_{HL}$.

Note that we did not impose the condition that users know the sizes or job size distributions of other users, or that all users get the same quantity of tokens, and hence Delayed Gratification remains a strongly dominant strategy for informed users even in a more relaxed version of our game. □

Proof of Proposition 1 The proof is illustrated in Figure 4. Define $B^H = B^H(1,t) - 1$ to be the busy period, of only high-priority jobs, initiated at time t, by execution of a single job (of size 1).

Points 1 and 2 of Proposition 1 are relatively straightforward. Point 1 follows from the definition of T_B and m, whereas point 2 follows from the fact that, under $\mathbf{P}_{HL}$, we can lower bound the time for U to reach the head of the L queue by ignoring any external arrivals to the L queue after $t = 0$. In that case user U reaches the head of the L queue at precisely $t = T + B + 1 + m$. As there may be external arrivals, under $\mathbf{P}_{HL}$ U will reach the head of the L queue no sooner than U would reach the head of the H queue under $\mathbf{P}_{LH}$, which is exactly at $t = T_B + 1 + m$.

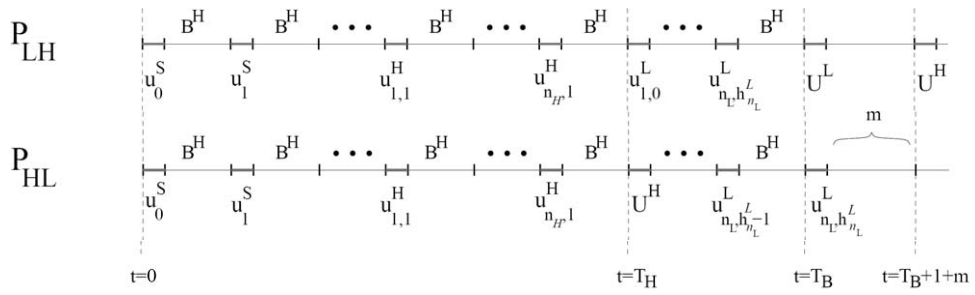


Figure 4 **Illustration of the equivalence of state under P_{LH} and P_{HL} via reordering.**

To prove points 3 and 4, it will be helpful to consider a reordered scheduling of jobs in the two systems, as depicted in Figure 4. While this reordered scheduling changes the order of users in the queue in the final state, the aim of the construction is to show the equivalence of the states of a user under $\mathbf{P}_{HL}$ and $\mathbf{P}_{LH}$ strategies followed by user U. We use u_0^S to denote the phase of the job that is at server at $t=0$ (note that this can be an L phase). We use u_j^S $(j \geq 1)$ to denote the jth H phase (relative to the state at $t=0^-$) of u^S (if there are any). By $u_{i,j}^H$, we mean the the jth H phase (relative to the state at $t=0^-$) of user u_i^H. For the users present in the L queue at $t=0^-$, $u_{i,0}^L$ denotes the L phase u_i^L is waiting on at $t=0^-$, and $u_{i,j}^L$ for $j \geq 1$ denotes the jth H phase following the L phase.

The reordered scheduling is as follows: We start by processing the phase of the user already at the server at time $t=0^-$. In Figure 4, this is depicted by u_0^S. When this phase ends, we do not schedule the next job until the busy period of incoming H jobs is finished. This is depicted by the B^H following u_0^S, and includes the leading H streaks of the arrivals to the H queue during u_0^S and the new arrivals to the H queue during B^H, which will subsequently move to the L queue by the end of this busy period. Then we pick the next phase of u^S (u_1^S) or the first scheduled phase of the next user, if scheduled before T_B+1+m, $(u_{1,1}^H,\ldots,u_{n_H,1}^H$; or $u_{1,0}^L,\ldots,u_{n_L,h_{n_L}^L}^L$ if no jobs are waiting in the high queue) and continue. The reordering preserves the final state (at $t=T_B+1+m$) of the jobs present in the system at $t=0^-$, and of the incoming jobs. Further, it is clear that for any user, the final state (queue, phase) under the reordering shown in Figure 4 are identical. In particular, refer to Figure 4, at $t=T_B+1+m$,

1. All the n_H users and u^S present in the system at $t=0^-$ finish their H streak and are in the L queue (if they have an L following). Else, by definition they would have been served.

2. All the arrivals into the H queue during $[0,T_B)$ finish their initial H streak and are in the L queue (if they have an L following). Else, by definition they would have been served.

3. All the arrivals into the H queue during $[T_B,T_B+1)$ finish their first H phase and are waiting in the queue corresponding to their second phase.

4. All the arrivals into the H queue during $[T_B+1,T_B+1+m)$ are in the H queue waiting for their first H phase.

5. All the n_L users in the L queue at $t=0^-$ finish their L phase and the following H streak and are back in the L queue (if they have an L following).

6. All the arrivals into the L queue during $[0,T_B+1+m)$ are in the L waiting for their first L phase.

The order of jobs after reordering the scheduling can be easily determined, although for proving Theorem 1, points 1 and 2 in the statement of the proposition about the position of U suffice. $\square$

It is somewhat surprising that saving tokens for the last quanta has such strong appeal to the *selfish* informed users. Furthermore, as DG is a strongly dominant strategy, it remains an equilibrium strategy for informed users for any arrival measure $\mathcal{M}$ for informed and uninformed users and any strategy choice of other users independent of the informed user's strategy. Hence, DG is a very compelling strategy to play.

4.2. Equilibrium analysis for uninformed users

In §4.1, we proved that Delayed Gratification is a strongly dominant strategy for informed users. We now consider the question of equilibrium behavior of uninformed users by numerically examining a very simple system.

Example 1. We consider a system with both informed and uninformed users. The job size distribution of all uninformed users is identical: the job size is 1 with probability p_1, or 2 with probability $p_2 = 1 - p_1$. We have two classes of informed users: those with job size 1, and those with job size 2. The uninformed users arrive according to a Poisson process with rate $\lambda\alpha$. Informed users of size 1 arrive according to an independent Poisson process with rate $\lambda(1-\alpha)p_1$, and informed users of size 2 arrive according to another independent Poisson process with rate $\lambda(1-\alpha)p_2$. Alternately, we can view the arrival process as a Poisson process with rate λ, where each user has a job size of 1 with probability p_1, or 2 with probability p_2. On arrival, each user is classified as uninformed with probability α (and hence does not see his job size), or classified as informed with probability $(1-\alpha)$ (and hence knows his job size). Each user gets $D = 1$ token. We use $\rho = \lambda(1 \cdot p_1 + 2 \cdot p_2)$ to denote the load of the system, and consider the equilibrium behavior under the assumption of stationarity.

Although the game set-up in §3 was defined for a finite number of players, we can show that set of equilibria the stationary game of Example 1 is an appropriately defined limit of the equilibria of a sequence of games, parametrized by the number of players N, as $N \to \infty$. We present a formal statement of this claim and proof in Appendix A.

We now specialize our definitions from §3 to Example 1. Under the above setup, there are two pure strategies for users: placing the token on the first quantum, which we denote as HL, and placing token on the second quantum, which we denote as LH. From Theorem 1, we know that all informed users of size 1 follow HL, and all informed users of size 2 follow LH. It now remains to identify the strategy for uninformed users.

We characterize the mixed strategy of an uninformed user by p_{HL}, which denotes the probability that the uninformed user places his token on the first quantum. The strategy $p_{HL}=1$ is equivalent to the pure strategy - HL, and the strategy $p_{HL}=0$ is equivalent to the pure strategy LH. Note that since $\mathcal{A}_i=\{\emptyset,\Omega\}$, the strategy p_{HL} is a constant, and not a function of the arrival sequence. Consider a tagged uninformed user U who chooses strategy p^U_{HL}, while all the other uninformed users choose strategy p^u_{HL}. Let $\mathbb{E}\left[D^{uninf}_{p^U_{HL},p^u_{HL}}\right]$ denote the mean delay experienced by the tagged uninformed user U in the given scenario. Upon observing p^u_{HL}, user U's best response is given by:

$$BR(p^u_{HL})=\arg\min_{p^U_{HL}}\mathbb{E}\left[D^{uninf}_{p^U_{HL},p^u_{HL}}\right]$$

Since all uninformed users are identical, the Nash equilibrium of the uninformed users will be a symmetric equilibrium, p^*_{HL} such that $p^*_{HL}=BR(p^*_{HL})$. For example, $p^*_{HL}=1$ will be an equilibrium if and only if when every uninformed user adopts HL, the mean delay of a tagged uninformed user is minimized by following HL as well. Clearly, $0<p^*_{HL}<1$ can be an equilibrium if and only if the mean delays of a tagged uninformed user under strategies HL and LH are equal. Given the above definitions, all that remains to be done to compute the equilibria of our game is to compute the function $\mathbb{E}\left[D^{uninf}_{p^U_{HL},p^u_{HL}}\right]$. We do this next.

Performance (Queueing) Analysis or Example 1 Our goal here is to derive the expression for the expected delay of a tagged uninformed user under the stationary game, described in Example 1, as a function of the tagged user's strategy, p^U_{HL}, and the symmetric strategy profile of other uninformed players, which we will shorthand as p_{HL}.

To begin, we define two classes of users: HL and LH. The HL class is made up of the p_{HL} fraction of uninformed users who place their token on first quantum, and the p_1 fraction of informed users who are of size 1. The class LH is made up of the $(1-p_{HL})$ fraction of uninformed users who place their token on the second quantum and p_2 fraction of informed users who are of size 2.

Define:

$$D^t_\ell=\text{expected delay of a type } t\in\{HL,LH\} \text{ user of size 2 in phase } \ell\in\{H,L\},$$
$$Q^t_\ell=\text{expected number of type } t\in\{HL,LH\} \text{ users in } \ell\in\{H,L\} \text{ priority queue}$$

Given the above quantities the expected delay of the tagged uninformed user can be written as:

$$\mathbb{E}\left[D^{uninf}_{p^U_{HL},p_{HL}}\right]=p^U_{HL}(D^{HL}_H+p_2D^{HL}_L)+(1-p^U_{HL})(D^{LH}_L+p_2D^{LH}_H).$$

Further, we can express the expected delay of the informed, and untagged uninformed users as:

$$\mathbb{E}\left[D^{inf}\right] = p_1 D_H^{HL} + p_2(D_L^{LH} + D_H^{LH})$$
$$\mathbb{E}\left[D^{uninf}\right] = p_{HL}(D_H^{HL} + p_2 D_L^{HL}) + (1-p_{HL})(D_L^{LH} + p_2 D_H^{LH}).$$

We will now derive expressions for D_ℓ^t, $t \in \{HL, LH\}, \ell \in \{H, L\}$. To start, we have the following relations from Little's law:

$$Q_L^{LH} = \lambda(\alpha(1-p_{HL}) + (1-\alpha)p_2)D_L^{LH}$$
$$Q_H^{LH} = \lambda(\alpha(1-p_{HL})p_2 + (1-\alpha)p_2)D_H^{LH}$$
$$Q_L^{HL} = \lambda(\alpha p_{HL} p_2)D_L^{HL}$$
$$Q_H^{HL} = \lambda(\alpha p_{HL} + (1-\alpha)p_1)D_H^{HL}$$

Finding the mean delay by conditioning on the state seen by a tagged user, and then unconditioning by taking expectation over the initial state, we obtain the following relations:

$$D_L^{LH} = \frac{\frac{\rho}{2} + Q_H^{HL} + Q_H^{LH} + \left(1 + \frac{\alpha(1-p_{HL})p_2 + (1-\alpha)p_2}{\alpha(1-p_{HL}) + (1-\alpha)p_2}\right) Q_L^{LH} + Q_L^{HL}}{1 - \lambda(\alpha p_{HL} + (1-\alpha)p_1)} \tag{1}$$

$$D_H^{LH} = \lambda(\alpha p_{HL} + (1-\alpha)p_1) \tag{2}$$

$$D_L^{HL} = \frac{\left[\begin{array}{l}\left(1 + \frac{\alpha(1-p_{HL})p_2 + (1-\alpha)p_2}{\alpha(1-p_{HL}) + (1-\alpha)p_2}\right) Q_L^{LH} + Q_L^{HL} + \frac{\alpha p_{HL} p_2}{\alpha p_{HL} + (1-\alpha)p_1} Q_H^{HL} \\ \quad + \lambda\left\{(\alpha p_{HL} + (1-\alpha)p_1) + (\alpha(1-p_{HL})(1+p_2) + 2(1-\alpha)p_2)\right\}(D_H^{HL} + 1)\end{array}\right]}{1 - \lambda(\alpha p_{HL} + (1-\alpha)p_1)} \tag{3}$$

$$D_H^{HL} = \frac{\rho}{2} + Q_H^{HL} + Q_H^{LH} \tag{4}$$

We provide an explanation for the expressions for D_H^{LH} and D_L^{HL} below. The other expressions follow the same line of reasoning. We use a similar procedure (but with significantly more state variables) for our numerical evaluations in §5.

Intuition for D_H^{LH}: Recall that D_H^{LH} denotes the mean delay in the H queue for a tagged user of size 2 who places his token on the second quantum. Observe that whenever an LH customer enters service for the first time, the high-priority queue is necessarily empty. Once the L quantum of this LH user is served, the user joins the back of the H queue. Thus the only jobs that delay the tagged user while he is in the high-priority queue are the HL arrivals during the service time of the tagged user's first (L) quantum. This is given by $\lambda(\alpha p_{HL} + (1-\alpha)p_1)$.

Intuition for D_L^{HL}: Recall that D_L^{HL} denotes the mean delay in the L queue for a tagged user of size 2 who places his token on the first quantum. By stationarity and PASTA, the mean number of type $t \in \{HL, LH\}$ jobs that this tagged user sees in queue $\ell \in \{H, L\}$ is Q_ℓ^t. All the Q_H^{LH} users

have left the system by the time the tagged user reaches the L priority queue. The Q_L^{LH} users are still in the L queue ahead of the tagged user. These are made up of both the informed users of size 2 and the uninformed users who follow LH. Of these Q_L^{LH} users, $\frac{\alpha(1-p_{HL})p_2+(1-\alpha)p_2}{\alpha(1-p_{HL})+(1-\alpha)p_2}$ fraction of users have a second quantum and will subsequently join the H queue, delaying the tagged user twice. The Q_L^{HL} users are also ahead of the tagged user in the L queue and each delays the tagged user by one unit. Of the Q_H^{HL} users that the tagged user saw in the H queue on arrival, $\frac{\alpha p_{HL}p_2}{\alpha p_{HL}+(1-\alpha)p_1}$ users had a second L quantum and are also in front of the tagged user in L queue. Moreover, while the tagged user was waiting (and receiving service for the first quantum) in the H queue for $D_H^{HL}+1$ units, the new arrivals into the H and L queue also create additional delays, since the former are now in the H queue, and the latter are in the L queue ahead of the tagged user. Finally, since the tagged user is in the L queue, all new HL arrivals that happen while users that arrive while the above users receive service also delay the tagged user. Thus the overall delay is given by the busy period of new HL arrivals (due only to the first H quantum) caused by the jobs in the H queue and in the L queue ahead of the tagged user when the tagged user joins the back of the L queue.

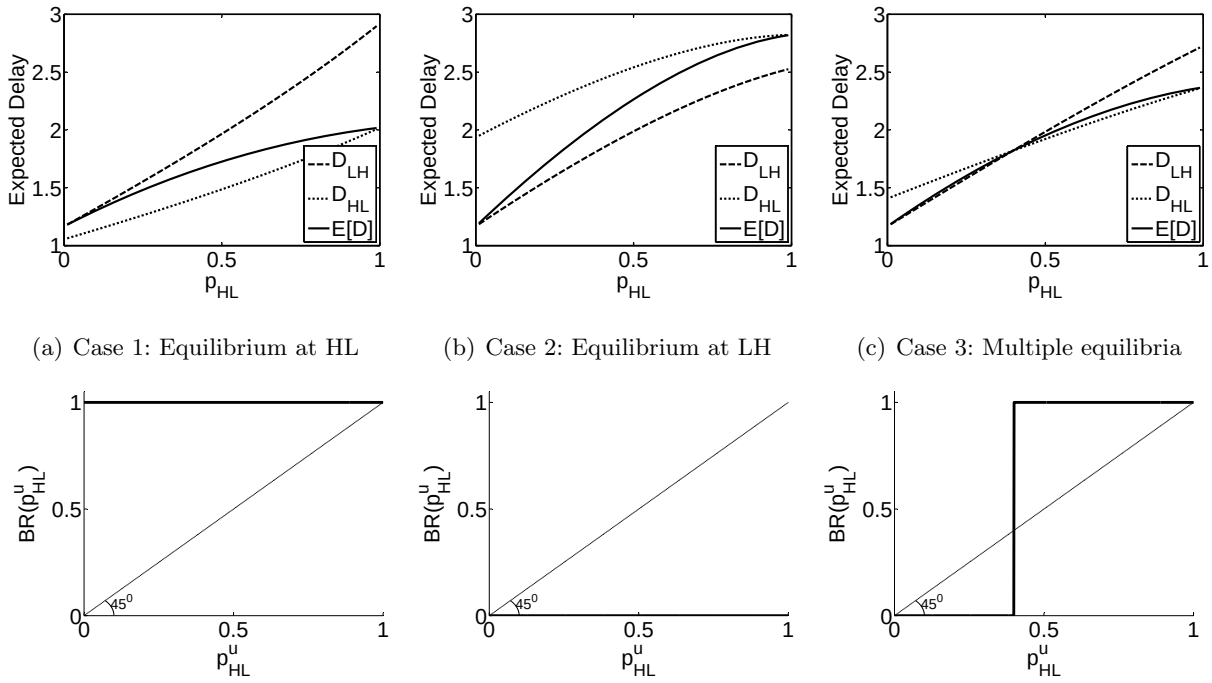
4.2.1. Equilibrium Behavior for Example 1 Figure 5 shows the potential behavior of uninformed users. We fixed $\alpha = 1$ for the results in Figure 5, that is, there are only uninformed users in the system. In each of the subfigures, the D_{LH} curve shows the mean delay of an arriving uninformed user who follows LH while every other uninformed user places their token on the first quantum (HL) with probability p_{HL}. The D_{HL} curve shows the mean delay of an arriving uninformed user who follows HL while every other uninformed user places their token on the first quantum with probability p_{HL}. The $\mathbb{E}[D]$ curve shows the overall mean delay of users in the system when all uninformed users follow strategy p_{HL}. These delay measures are calculated using Equations (1)-(4).

We observe the following:

1. There are three possible types of equilibria:

(a) Unique equilibrium at $p_{HL} = 1$, i.e. use HL as displayed in Figure 5(a). Note that the socially optimal policy for the uninformed users is to use LH which also reduces their own mean delay. Thus this equilibrium is an example of Braess' paradox (Braess 1969), whereby if the users are forced to play a degenerate game by only allowing them the option of LH, then users achieve optimal delays. However, if we add the option of HL and allow users to place tokens (in their own best interest), then the mean delay of all users becomes worse.

(b) Unique equilibrium at $p_{HL} = 0$, i.e. use LH as displayed in Figure 5(b).



(a) Case 1: Equilibrium at HL (b) Case 2: Equilibrium at LH (c) Case 3: Multiple equilibria

(d) Case 1: Equilibrium at HL (e) Case 2: Equilibrium at LH (f) Case 3: Multiple equilibria

Figure 5 Possible equilibria, as a function of p^u_{HL}, the strategy adopted by other uninformed users. Figures (a)-(c) show the mean delay of a tagged uninformed user for the two pure strategies, LH and HL, as a function of the strategy adopted by other users. The dashed curve represents the mean delay under the pure strategy LH, and the dotted curve shows the mean delay under the pure strategy HL. The solid curve shows the expected delay of all users as a function of p_{HL}. Figures (d)-(f) show the best response function $BR(p^u_{HL})$ corresponding to Figures (a)-(c), respectively. For each of the curves, we fixed $\rho = 0.7$ and $\alpha = 1$. The value of p_1 for Figures (a)-(c) are 0.6, 0.1 and 0.4, respectively.

(c) Three equilibria: at $p_{HL} = 0, p^*_{HL}, 1$, see Figure 5(c). The equilibrium at $p_{HL} = p^*_{HL}$ is unstable. If a positive fraction of users deviates to LH, then the system converges to $p_{HL} = 0$ and hence social optimality. If a positive fraction of users deviate towards HL, the system converges to $p_{HL} = 1$[2].

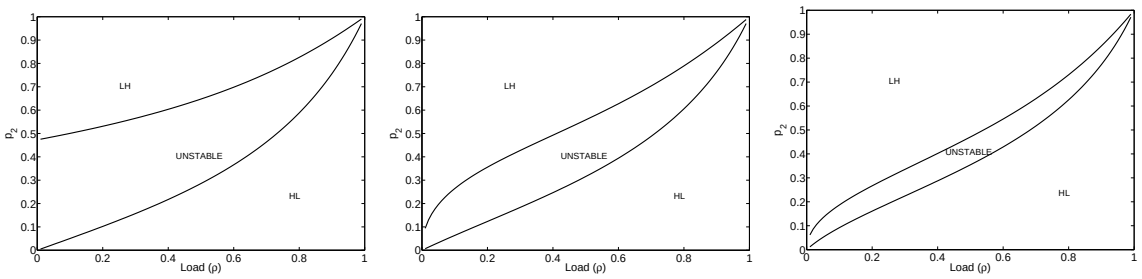
2. The socially optimal policy always seems to be $p_{HL} = 0$, i.e. use LH. This can be seen from $\mathbb{E}[D]$ being minimized at $p_{HL} = 0$. While Figure 5 only shows the case $\alpha = 1$, we have observed this behavior across the entire range of α.

Finally, we explore how the existence of the aforementioned equilibria depends on system characteristics. Figure 6 illustrates how the existence of equilibria depends on load, p_2, and α. Begin by observing the case $\alpha = 1$. When the load is approaching 1, the system is highly utilized and hence

[2] This is a follow-the-crowd situation (see e.g. §1.5.1 of Hassin and Haviv 2003).

jobs observe long delays. In this case, independent of p_1, going to the high-priority queue first, and then leaving the system with probability p_1 is favored over saving the token for the second quantum and possibly wasting it. For lower loads, all equilibria exist. As p_2 increases users switch from using HL, to the unstable equilibrium, to LH which in the limit $p_2 \to 1$ is equivalent to DG, as minimizing overtaking (using DG) becomes increasingly important. When α decreases due to addition of informed users, the region where LH is the unique equilibrium increases. This is because informed users follow DG and have the potential to overtake uninformed size 2 jobs who are suboptimally following HL. Therefore, adding informed users to the system makes the uninformed users "better behaved," leading to socially optimal behavior for a larger range of system parameters.

Thus, even in the simple setting of Example 1, the equilibrium behavior of the informed users is complex. Not only is the equilibrium sensitive to the system load, the job size distribution and the fraction of informed users, but there are also settings where there are multiple equilibria, with the possibility of some of them being unstable. This complexity will only increase further for other job size distributions and it is extremely likely that faced with the task of placing tokens in a real setting, an uninformed user will likely follow a heuristic - such as Immediate Gratification which does not waste any tokens - rather than compute the equilibrium of a game with a high dimensional strategy space. We will make such an assumption in the next section.



(a) $\alpha = 1$ (b) $\alpha = 0.75$ (c) $\alpha = 0.4$

Figure 6 Illustration of the regions of possible equilibria as a function of system load ρ and p_2 for different values of fraction of uninformed users.

5. Performance (Queueing) Analysis

The analytical results presented in §4 enable us to establish the strategies of the $(1-\alpha)$ proportion of users who know the sizes of their jobs: Delayed Gratification. Further, we saw that for the uninformed users, the equilibrium behavior can be quite complex, and difficult to determine. Therefore,

in this section we will assume that the α fraction of users who do *not* know their job size utilize Immediate Gratification[3]. In support of this assumption we note that:

- We have already seen an example in §4.2 where Immediate Gratification is the unique equilibrium;
- Immediate Gratification closely resembles the popular scheduling discipline of FB (also known as Least Attained Service), in which jobs that have received the least service are given priority. FB is known to be optimal in certain cases when the exact job sizes are not available (i.e. customers are uninformed), Yashkov (1978); and
- From an uninformed user's point of view, rather than spending significant computational effort to find the optimal policy, it might be more reasonable to spend tokens on the initial quanta to ensure none are wasted (saved only to have the job complete before they are used).

Thus, results of §4.1 coupled with our assumption that the uninformed users follow Immediate Gratification completely specifies the behavior of the users in the system. We can now explore questions such as:

1. What performance will users experience?
2. How many tokens should be distributed to each user so as to optimize performance?
3. How effective is our game at approximating SRPT-like performance?
4. Which factors most affect the optimal token distribution and performance of our system?

To answer these questions, we develop a queueing theory based algorithm to calculate mean system delay experienced by DG and IG users, as a function of α, system load, the coefficient of variation of the job sizes, and the token endowment. This algorithm is a significant extension of the algorithm from §4.2 and the simpler algorithm provided by Wolff (1970) to analyze round-robin scheduling; compared to Wolff (1970) we track significantly more state variables and distinguish between different types of jobs, which change type while passing through the system.

We briefly describe our algorithm in §5.1, before presenting graphs derived from the application of our algorithm, in §5.2, which answer the questions posed above. In §5.3, we use our algorithm to generate additional insights into the uninformed users' strategy.

5.1. Algorithm for Calculating System Performance

We now briefly outline the queueing-theoretic algorithm we devise to calculate mean customer delay. We assume users arrive according to a Poisson process; each user brings a work requirement which may be broken up into a discrete, finite number of quanta; and the number of quanta of

[3] A similar analysis could be carried out if we assumed uninformed users utilized an alternate heuristic policy, such as putting all tokens at or before the expected job size.

work that each user brings is i.i.d. Leveraging these assumptions, we first decompose a user's total time in queue into the sum of the mean delay of *each quanta* of the user's job. This decomposition is dependent upon three factors: (i) whether the user is an informed or uninformed user (playing DG or IG); (ii) the user's total job size; and (iii) the size of the token endowment.

Given this decomposition, we calculate the expected delay of each individual quanta by solving a linear system of equations. These equations are derived by conditioning on the system state that a "tagged user" sees upon arrival, as well as the characteristics of all users who will arrive while the tagged user is in queue. As in §4.2, these arrivals complicate the analysis significantly: If the tagged user is currently waiting in the low-priority queue and a new user arrives into the high-priority queue, the tagged user must wait through the entire *busy period* initiated by this new user's high-priority quanta. Here we will describe and provide intuition for how to derive the expected waiting time for the two types of quanta for informed users, and then relate these waiting times to the number in system. More details, i.e. the approach for uninformed users and the precise equations that need to be solved, can be found in Appendix B. For tractability we need a bounded probability distribution. Let, the size of the job, x be the realization of $X_i \in \mathbb{N} \leq M$, where M is the maximum support of the probability distribution.

5.1.1. Delay for a DG user in High-priority queue We begin with the simplest case, i.e. obtaining $\mathbb{E}[W^{LH}_{i,DG}]$, where $W^{LH}_{i,DG}$ is the (random variable for the) time spent in the high-priority queue by a DG user, waiting for the service of the i^{th} high-priority quantum (that is, the $(x-D+i)^{th}$ quantum overall). Note that this is for a job with original size $x > D$ ($x \leq D$ is the next case we cover). Therefore, this job has gone through the low-priority queue $x-D$ times for its first $x-D$ quanta, and then migrated to the high-priority queue. Let S be the generic random variable denoting size of other users in the system. The following types of jobs contribute to $W^{LH}_{i,DG}$ (see Appendix B.1):

1. All DG jobs with $S \leq D$ (and hence those jobs that enter the H queue on arrival), that arrived after the tagged job started its last low-priority quantum (at that point the high-priority queue was empty) *and* that are still around when the tagged job starts waiting on its i^{th} H quantum to be processed. More precisely: DG users of size $1 \leq S \leq D$ that arrive during a time interval of length $W^{LH}_{i-1,DG}+1$, DG users of size $2 \leq S \leq D$ that arrive during a time interval of length $W^{LH}_{i-2,DG}+1$, and so on up to all DG users of size $i \leq S \leq D$ that arrive while the last low-priority quanta of the tagged job was processed ($W^{LH}_{0,DG}+1$, where $W^{LH}_{0,DG}=0$).

2. All IG jobs that arrived after the tagged job started its last low-priority quantum *and* that are still around when the tagged job starts waiting on its i^{th} quantum to be processed. More precisely:

IG users of size $S \geq 1$ that arrive during a time interval of length $W^{LH}_{i-1,DG} + 1$, IG users of size $S \geq 2$ that arrive during a time interval of length $W^{LH}_{i-2,DG} + 1$, and so on up to all IG users of size $S \geq i$ that arrive during a time interval of length $W^{LH}_{0,DG} + 1$.

We now consider the slightly more involved case for $x \leq D$, and see how to obtain $\mathbb{E}[W^{H}_{i,DG}]$, where $W^{H}_{i,DG}$ is (the random variable for) the time spent in the high-priority queue by a DG user with original size $x \leq D$, waiting for service of the i^{th} quantum. The following types of jobs contribute to $W^{H}_{i,DG}$ (see Appendix B.2):

1. All jobs present in the high-priority queue upon arrival of the tagged job into the system, that had i or more quanta to go in the high-priority queue. These include IG users, and DG users with remaining size larger than or equal to i and either original size $S \leq D$, or $S > D$ but remaining size less than or equal to D.

2. All DG jobs with $S \leq D$, that arrived after the tagged job first entered the system *and* that are still around when the tagged job starts waiting on its i^{th} quantum to be processed. More precisely: DG users of size $1 \leq S \leq D$ that arrive during a time interval of length $W^{H}_{i-1,DG} + 1$, DG users of size $2 \leq S \leq D$ that arrive during a time interval of length $W^{H}_{i-2,DG} + 1$, and so on up to DG users of size $i - 1 \leq S \leq D$ that arrive during $W^{H}_{1,DG} + 1$.

3. All IG jobs that arrived after the tagged job first entered the system *and* that are still around when the tagged job starts waiting on its i^{th} quantum to be processed. More precisely: IG users of size $S \geq 1$ that arrive during a time interval of length $W^{H}_{i-1,DG} + 1$, IG users of size $S \geq 2$ that arrive during a time interval of length $W^{H}_{i-2,DG} + 1$, and so on up to all IG users of size $S \geq i - 1$ that arrive during $W^{H}_{1,DG} + 1$.

4. If there was a job J in service when the tagged user arrived, the job J can contribute to $W^{H}_{i,DG}$ if it is one of the following types:

   (a) A DG job processing its last low-priority quantum, and thus migrating to the high-priority queue to process the last D quanta.

   (b) A DG job with original size larger than D, and remaining size (number of quanta of the job at the server that have not been completely processed yet) less than or equal to D but larger than $i - 1$, thus processing a high-priority quantum upon arrival of the tagged job.

   (c) A DG job with original size less than or equal to D, but remaining size larger than $i - 1$.

   (d) An IG job with more than $i - 1$ quanta in the high-priority queue to go.

So far we have focused on DG jobs. The analysis of delay of uninformed users, that is those using IG, in the High-priority queue is exactly the same as that for a DG user with original size $x \leq D$. The analysis of delay of DG and IG users in the Low-priority queue is conceptually similar, but more tedious.

5.1.2. Relating waiting time to number in system The delay of any given quantum of a tagged job depends on the system state observed by the job when it first arrives to the system. Following the outline of §5.1.1, we can express the expected delay for each quanta of the tagged job conditioned on the system state (number of users of each type (DG/IG, original size, remaining size) in each queue). The expected delay conditioned on the state seen on arrival turns out to be a linear function of the state. Thus, by linearity of expectation, by unconditioning on the arrival state, we obtain a system of equations relating the mean delays of the quanta of a tagged user, and the expected values of the state variables seen by a tagged user on arrival.

As job arrivals follow a Poisson process we can invoke PASTA (Poisson Arrivals See Time Averages) to deduce that the expected values of the state variables seen by an arbitrary arrival are the same as the time average values of these state variables. Next, we use Little's law to relate the expected delays of the quanta of each type of user to the time average number in system of each type of user, and hence the time average values of the state variables. We thus obtain a linear system of equations which can be solved to obtain the expected delay for each quanta of a tagged user of any size or class (DG/IG).

To obtain the mean delay of all users, we uncondition again by integrating over the characteristics of the tagged user (IG or DG and his total job size) which finally yields the mean delay of a "typical" arrival to the system. See Appendix B for details.

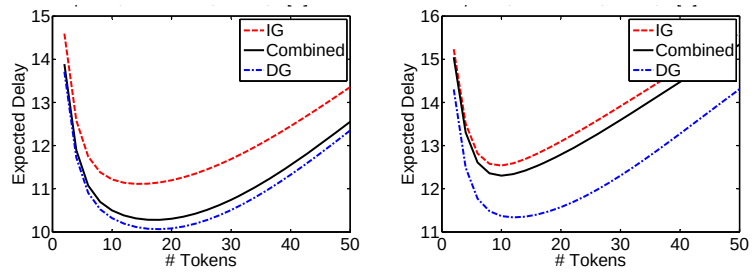
For the instances that we consider, the algorithm takes no more than a couple of seconds. The computational complexity is at most $O(M^3)$ as we could use Gaussian elimination to solve our system of $2M$ equations.

5.2. System Performance and Insights

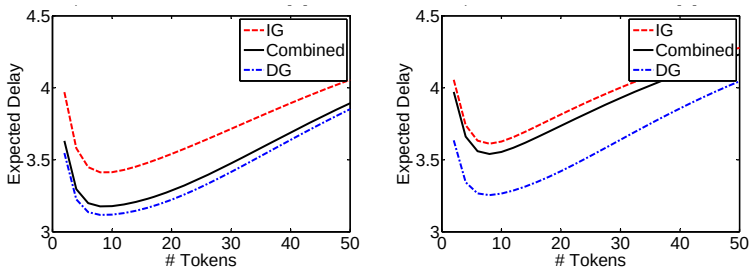
In our calculations, we assume that the users' number of quanta follows a mixture of truncated geometric distributions; these allow us to model a variety of C^2 values parsimoniously. We stress though that our algorithm can be used with any discrete distribution with finite support.

Figures 7(a) - 7(d) show how the token endowment (horizontal axis) influences system delay of the DG and IG customers individually, and in aggregate (vertical axis), for a representative problem instance. Here $C^2 = 4.36$, while both ρ and α are varied to change population characteristics.

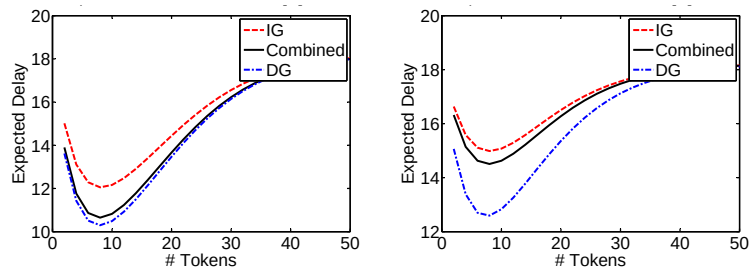
As expected, the delay of the DG customers is strictly less than the delay of the IG customers. Note that the mean delay of each customer type (and in aggregate) decreases rapidly as the number of tokens is increased toward optimal, before increasing more gradually as the token endowment surpasses optimal. This illustrates the fundamental driver of behavior with respect to tokens, which we call the *token trade-off*: An increase in tokens shifts some of each user's (of size $> D$) quanta from



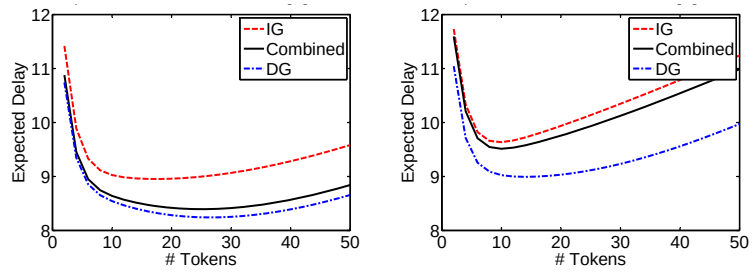
(a) $\rho = 0.8$,$C^2 = 4.36$, $\alpha = 0.2$. (b) $\rho = 0.8$, $C^2 = 4.36$, $\alpha = 0.8$.



(c) $\rho = 0.5$, $C^2 = 4.36$, $\alpha = 0.2$. (d) $\rho = 0.5$, $C^2 = 4.36$, $\alpha = 0.8$.



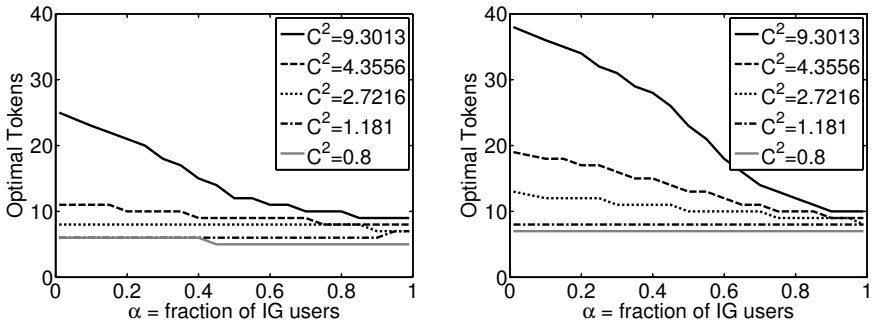
(e) $\rho = 0.8$, $C^2 = 1.18$, $\alpha = 0.2$. (f) $\rho = 0.8$, $C^2 = 1.18$, $\alpha = 0.8$.



(g) $\rho = 0.8$, $C^2 = 7.05$, $\alpha = 0.2$. (h) $\rho = 0.8$, $C^2 = 7.05$, $\alpha = 0.8$.

Figure 7 Expected delay vs Number of tokens.

the low-priority to the high-priority queue. This *increases* the delay of those customers already in the high-priority queue by interfering with their processing, but *decreases* the delay of the customers who are shifted. Initially, as the endowment increases, the *gain* for the shifted customers outweighs the *pain* inflicted on those customers already in the high-priority queue as this queue is relatively sparsely populated. But as the number of tokens grows, more and more quanta are included within



(a) $\rho = 0.6$. (b) $\rho = 0.8$.

Figure 8 Optimal number of tokens vs α.

the high-priority queue, and congestion in the high-priority queue increases to the point that the pain outweighs the gains of increasing endowment.

Comparing Figures 7(a) and 7(c) with Figures 7(b) and 7(d) we see that system improvement grows with the proportion of DG customers: The DG users' job-size knowledge allows them to utilize their tokens more efficiently – the *token trade-off* is better for DG customers than for IG customers, as only those arriving DG customers with no more quanta than tokens are immediately granted high priority, but under IG *all* arriving customers initially assume high priority. Thus large DG arrivals only interfere with small jobs (size $< D$) when the system is "empty", i.e. when they move from the low to the high queue, but IG users interfere with all jobs immediately upon arrival. Thus the greater the proportion of DG users, the less relative *pain* for the the same *gain*. Similarly, the DG users' delay is minimized at a slightly larger allocation than the IG users'; as their *token trade-off* is better it would be in the interest of the system to grant them larger allotments.

Next, comparing Figures 7(a) and 7(b) with the pairs of Figures 7(e) and 7(f) and Figures 7(g) and 7(h). We observe that the optimal number of tokens seems to increase in C^2. We will further investigate this in Figure 8.

Note finally that with zero or very large numbers of tokens prioritization vanishes (all customers are the same priority) and the scheduling policy becomes Round Robin. Compared to RR (or PS), the optimal token allocation can reduce delays by 15%-25%, for our examples: Tokens can be an effective tool at reducing system delays, *assuming the system manager selects the optimal number of tokens.* We examine the sensitivity of optimal token allocation to system parameters next.

Figures 8(a) and 8(b) show how the optimal number of tokens, (vertical axis) decreases as the proportion of IG customers grows (horizontal axis) for different values of job size C^2, for $\rho = 0.6$, and $\rho = 0.8$, respectively. These figures also show that as system traffic increases, or as the system variability grows, the optimal token allocation grows larger, which is in agreement with the cutoffs

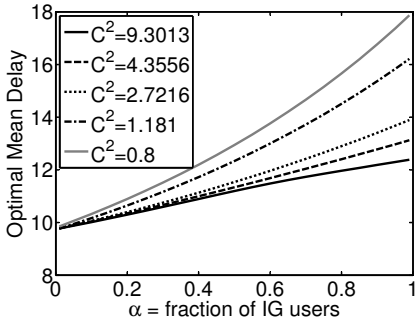
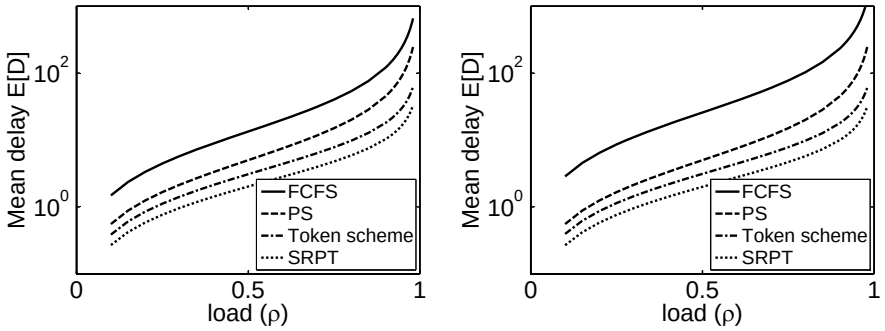


Figure 9 **Optimal delay vs** α**,** $\rho = 0.8$**.**

for job segmentation proposed in Bansal and Gamarnik (2006). Both of these effects spring again from the the *token trade-off*: As load or variability grow, congestion at the low-priority queue grows more rapidly than at the high-priority queue. In particular, jobs that immediately go to the high-priority queue delay jobs in both the high and the low-priority queue, whereas jobs that go to the low-priority queue will only increase congestion at the low-priority queue. Hence, all arriving jobs increase congestion at the low-priority queue and only some increase congestion at the high-priority queue. Thus shifting more quanta to the high-priority queue yields a net benefit; the relative *gain* increases faster than the relative *pain*, and optimal endowment consequently increases.

Figure 9 builds upon the previous two, showing how the mean system delay given an *optimal* token allocation changes (vertical axis), versus the proportion of IG customers (horizontal axis), for different values of ρ and C^2. Once again we see that systems with larger variability and larger proportions of DG customers are more amenable to prioritization via tokens, and thus enjoy comparably smaller delays. In the case of $\alpha = 0$, when all users utilize DG, our scheduling strategy becomes identical with what Harchol-Balter et al. (2003) and Wierman and Nuyens (2008) define as *Two-level SRPT*: These authors consider two queues; jobs of remaining size less than a certain cutoff (the endowment) go to the high-priority queue, and all other jobs go to the low-priority queue. But how effective is this Two-level SRPT as a scheduling policy?

From Figure 9 we see that it reduces the effects of job-size variability dramatically: As α approaches zero and hence all users start using DG, the mean delay for all of the C^2 values converge to values very close to one another, between 9.77 and 9.85. As expected, our policy is not as effective as running pure SRPT, which would lead to mean delay values between 5.73 (for $C^2 = 9.3013$) and 6.78 (for $C^2 = 0.8$). But when compared to PS, which would yield a mean delay of 20 for all values of C^2, we see that our game can claim between 71.70% and 76.76% of the possible improvements of SRPT over PS, *using only two queues*, and *in the presence of selfish users.* By increasing the number of priority level (and accompanying currencies) our game performance grows arbitrarily tight to that of SRPT.



(a) $\mathbb{E}[X] = 5$, $C^2 = 4.3555$. (b) $\mathbb{E}[X] = 5$, $C^2 = 9.3013$.

Figure 10 Comparison of $\mathbb{E}[D]$ across varying scheduling policies.

We conclude with a more general comparison of several scheduling policies, in Figure 10: First Come First Served (FCFS), PS, our Token Scheme in the presence of only informed customers and the optimal number of tokens, and pure SRPT. Here we observe that, as load increases, while FCFS and PS lead to skyrocketing delays (i.e. though the vertical distance from SRPT remains roughly the same, note that the y-axis is on a log scale) the delay of the optimal token scheme remains relatively close to the delay of SRPT.

5.3. Further Investigation of uninformed users' strategies

To obtain the foregoing numerical results, we assumed that the uninformed users place their tokens according to Immediate Gratification (IG). From the perspective of the system administrator an analogue policy to IG would be Foreground-Background (FB also known as Least-Attained-Service). Under FB, jobs which have received the least service are given priority, and hence the priority of quanta decreases monotonically over time (this policy is known to be optimal in some settings, see e.g. Yashkov 1978, 1987, Righter and Shantikumar 1989, Feng and Misra 2003). In this section we will investigate plausible strategies for users who have partial, distributional, knowledge about their job size in the presence of fully informed users.

Let us assume that a customer is at least certain that his job size lies within the range of $[a, b]$. If $D < a$ the user can delay gratification to quantum $a - D$ without fear of letting tokens unnecessarily go to waste, but could it be optimal to delay gratification further and risk wasting tokens? We show that this may indeed be the case by comparing IG with the most delayed strategy, DG, assuming job size b and seeing that for some uninformed customers DG is better than IG.

We examine this question in Figure 11, which compares the relative merits of a pure IG versus a pure DG policy, showing mean delay under both IG and DG (vertical axis) against the user's job-size (horizontal axis) for a representative example under the assumption $D = 35$ (the optimal

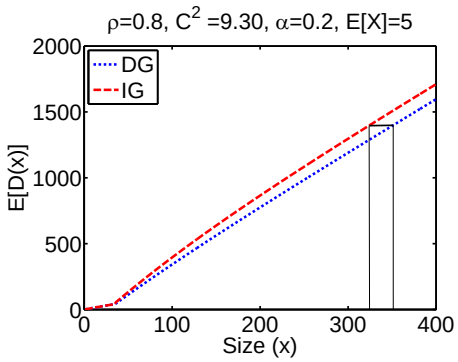


Figure 11 **Expected delay vs size.**

allocation for this example). This figure assumes that the 80% informed users play DG, while the uninformed users play IG.

Recall that the use of DG presumes the user knows his job size, whereas IG does not. Nevertheless, for job sizes no larger than D the delays are identical, as both sets of users use all of their tokens immediately. As job sizes grow DG begins to outperform IG. What is crucial for our discussion is the *horizontal gap* between these two policies – see the line segment marked on the figure, linking the delay of an IG user of size 325 with the equivalent delay of a DG customer of size 352.

Let us consider a tagged *uninformed* user who knows that his job size is between 325 and 352. In this case, for this user, acting as though his job size were in fact 352 and using DG would lead to a (weakly) *smaller* mean delay than using IG, as his delay is upper bounded by that of a DG job of size 352. (If his job size is less than 352 he will just terminate earlier than he expects, wasting tokens but also reducing his delay.) Thus it is beneficial for him to utilize DG, and potentially waste a large proportion of his tokens. Note that DG is just one potential policy, there likely exist other, more complex, policies for uninformed users that are better still. This leads us to our next question:

With partial knowledge, can we be certain tokens that should be placed contiguously?

We now briefly consider the case where users have partial knowledge about their job size and analyze how this affects their token placement.

LEMMA 1. *Users that know that their job size is drawn from a 2 point distribution should place their tokens in at most 2 contiguous blocks, preceding their sizes with positive support.*

Proof of Lemma 1 Consider a job of size a with probability $0 < p_a < 1$ and size $b > a$ with probability $1 - p_a$, and an allocation of $D < b$ tokens. We proceed by counterexample. If D^a tokens are placed on quanta $a - D^a + 1, a - D^a, \ldots, a$, and D^b tokens are placed on quanta $b - D^b + 1, b - D^b, \ldots, b$, and $D^a + D^b < D$ there must exist a third block of tokens that is placed ahead of one of

the two blocks (of size D^a and D^b). By Theorem 1, using these tokens should optimally be delayed and hence these tokens should be placed adjacent to the blocks of either D^a or D^b. □

6. Extension to multiserver setting

Since one of the settings in which we envision our scheme of being useful is in scheduling supercomputing centers, it is natural to ask whether and to what extent our scheme extends to multiserver scenarios. It is indeed straightforward to extend our token-based scheme when there are more than one server: Whenever a server becomes free after serving a quantum, it picks the next job to serve from the the head of the highest priority queue that is non-empty. In this setting Theorem 1 can be easily extended to prove that DG is a strongly dominant strategy even in a multiserver setting. Further, as we add more priority levels, our token based scheme is once again a better approximation of SRPT under multiserver setting. However, SRPT is not necessarily the optimal scheduling policy when the number of servers exceeds 1. In particular, it has been shown in Leonardi and Raz (1997) that there exist sample paths of arrivals on which the mean response time under SRPT can be a factor $\log(P)$ times the optimal mean response time, where P is the ratio of the largest to the smallest job size in the arrival sequence. But, the authors also show that mean response time under SRPT can be at worst a factor of $4+\log(P)$ away from the optimal mean response time. Further, no on-line scheduling policy, that is a scheduling policy that makes its decision based only on the arrivals so far and can't look into the future, has a better worst case performance guarantee than SRPT within a constant factor. (Thus within a constant factor, SRPT is indeed the optimal on-line scheduling policy.) The optimality and performance guarantee of SRPT under a stochastic arrival process is still open. However, we believe that even in a multiserver setting, SRPT is the correct scheduling policy, and hence our token-based scheme will guarantee competitive performance.

7. Conclusions

In this paper we consider the problem of online single-machine scheduling. When job sizes are private information of all users, mechanism design has been used (primarily Clarke taxes) to induce incentive compatibility and obtain job size information as well as job urgency. In this paper we consider for the first time a mix of users: Users who *know* their job size and users who *genuinely don't*. We derive a system that will not unfairly punish the users who have no knowledge about their job size, as one would want to do with users who behave strategically and lie about their job size.

The mechanism developed in this paper motivates users with private information about their job size to provide truthful signals about their true remaining size. These signals are revealed through

the placement of a fixed number of tokens distributed to each user by the system administrator. For the informed users, we show that there exists a strongly dominant strategy whereby these users place tokens to "boost" priority for the final portion of their job. We refer to this strategy as Delayed Gratification (DG). Intuitively, by choosing to prioritize the final portion of their job (i.e. when remaining size is small), the users self-schedule themselves according to an approximation of the optimal schedule: SRPT. We also show that strategies for uninformed users are complex and remain an open question.

Our mechanism differs in at least two ways from the mechanisms proposed in the literature. First, our mechanism allows for the fair treatment of both informed and uninformed users. Second, we use a virtual currency that is of no value outside the system. Some existing papers also use virtual currency, but rarely consider the question how much money (tokens) to endow users with, and only have 1 class of users.

Using queueing-theoretic analysis, we calculate the expected waiting time and the optimal token endowment given the system parameters. We find that the optimal token endowment increases with the variance of job size distribution and the system load, but decreases as the fraction of users who don't know their job size increases. These are consequences of what we term the token tradeoff, which we identify as the driving factor behind the optimal token allocation.

We hope our work will find application in areas such as supercomputing, where the job size information is often private or nonexistent, and the use of real money to signal priority is undesirable.

While in this paper we focused on users with either exact, or absolutely no knowledge of the job sizes, in many scenarios users may have a "reasonable" estimate of their size. Further, these estimates will become tighter as the job is processed. One way to utilize this knowledge is instead of asking users to place tokens on arrival. to allow them to place tokens in an as-you-go manner. Developing and analyzing mechanisms to handle such users is significantly more complex, and planned for future work.

References

Aalto, S., U. Ayesta, E. Nyberg-Oksanen. 2004. Two-level processor-sharing scheduling disciplines: mean delay analysis. *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on measurement and modeling of computer systems* 97–105.

Adiri, I., U. Yechiali. 1974. Optimal priority-purchasing and pricing decisions in nonmonopoly and monopoly queues. *Oper. Res.* **22** 1051–1066.

Avrachenkov, K., P. Brown, N. Osipova. 2007. Optimal choice of threshold in two level processor sharing. *ArXiv e-prints* **706**.

Bansal, N., D. Gamarnik. 2006. Handling load with less stress. *Queueing Systems* **54**(1) 45–54.

Braess, D. 1969. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung* **12** 258–268.

Chiang, S.H., A. Arpaci-Dusseau, M.K. Vernon. 2002. The impact of more accurate requested runtimes on production job scheduling performance. Feitelson et al. (2002).

Chun, B.N., P. Buonadonna, A. AuYoung, Chaki Ng, D.C. Parkes, J. Shneidman, A.C. Snoeren, A. Vahdat. 2005. Mirage: A microeconomic resource allocation system for sensornet testbeds. *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on* 19–28.

Dolan, R.J. 1978. Incentive mechanisms for priority queuing problems. *Bell Journal of Econ.* **9**(2) 421–436.

Feitelson, D.G., L. Rudolph, U. Schweigelsohn. 2002. *Job Scheduling Strategies for Parallel Processing*. Springer Verlag, Lect. Notes in Comput. Sci. vol. 2537.

Feng, H., V. Misra. 2003. Mixed scheduling disciplines for network flows. *ACM SIGMETRICS Perform. Eval. Rev. Special issue on MAMA 2003* **31**(2) 36–39.

Harchol-Balter, M., B. Schroeder, N. Bansal, M. Agrawal. 2003. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems* **21** 207–233.

Hassin, R., M. Haviv. 1997. Equilibrium threshold strategies: The case of queues with priorities. *Oper. Res.* **45**(6) 966–973.

Hassin, R., M. Haviv. 2003. *To Queue or not to Queue*. Kluwer, Norwell, Ma, USA.

Lee, C.B., Y. Schwartzman, J. Hardy, A. Snavely. 2004. Are user runtime estimates inherently inaccurate? *10th Workshop on Job Scheduling Strategies for Parallel Processing* .

Leonardi, S., D. Raz. 1997. Approximating total flow time on parallel machines. *Proceedings of the Symposium on Theory of Computation* .

Lu, D., P. Dinda, Y. Qiao, H. Sheng, F. Bustamante. 2004a. Applications of srpt scheduling with inaccurate information. *Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS* .

Lu, D., H. Sheng, P. Dinda. 2004b. Size-based scheduling policies with inaccurate scheduling information.

Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 31–38.

Mendelson, H., S. Whang. 1990. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Oper. Res.* **38**(5) 870–883.

Mu'alem, A.W., D.G. Feitelson. 2001. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *IEEE Trans. Prallel & Distributed Syst.* **12**(6) 529–543.

Peterson, J.L., A. Silberschatz. 2002. *Operating systems concepts*. John Wiley & Sons Inc.

Righter, R., J.G. Shantikumar. 1989. Scheduling multiclass single server queueing systems to stocastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences* **3** 323–333.

Schrage, L.E. 1968. A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res.* **16**(3) 687–690.

Snell, Q.O., M.J. Clement, D.B. Jackson. 2002. Preemption based backfill. Feitelson et al. (2002).

Wierman, A., M. Nuyens. 2008. Near-optimal scheduling despite inexact job-size information. *SIGMETRICS '08: Proceedings of the international conference on Measurement and modeling of computer systems*.

Wolff, R.W. 1970. Time sharing with priorities. *SIAM J. Appl. Math.* 566–574.

Yashkov, S.F. 1978. On feedback sharing a processor among jobs with minimal serviced length (in Russian). *Technika Sredstv Svyazi. Ser. ASU* (2) 51–62.

Yashkov, S.F. 1987. Processor-sharing queues: Some progress in analysis. *Queueing Systems* **2** 1–17.

Appendix A: Convergence of finite game to stationary game

The goal of this section is to show that the equilibria of the stationary game defined in Example 1 are an appropriately defined limit of a sequence of games with finite N number of players, as $N \to \infty$.

Consider a game with N players of whom $\lceil \alpha \cdot N \rceil$ are uninformed with the job size distribution defined in Example 1 (1 quantum with probability p_1, 2 quantum with probability $p_2 = 1 - p_1$), $\lceil (1-\alpha) \cdot p_1 \cdot N \rceil$ are informed with job size of 1 quantum, and the rest are informed with a job size of 2 quanta. The arrival instants of the uninformed users are generated according to a Poisson process with rate $\lambda\alpha$ starting at $t = 0$, and the uninformed users are mapped to the $N\alpha$ arrival instants at random (i.e., all permutations are equally likely). The arrival instants of the informed users are generated similarly, by independent Poisson processes with appropriate rates. Thus we have defined $\mathcal{M}$, the measure on $\Omega = \{\mathbb{R}^+\}^N$, the space of arrival sequences of the N players. The signal functions of all players are independent of the arrival sequence, and hence no information about the realized sample path is available to the players. That is, $\mathcal{A}_i = \{\emptyset, \Omega\}$ for all players.

Let $\mathcal{E}_{sta}$, $\mathcal{E}_{N,\epsilon}$, $\widetilde{\mathcal{E}}$, $\mathcal{E}_N$ and $\widehat{\mathcal{E}}$ be defined as follows:

$$\mathcal{E}_{sta} \doteq \text{Set of Nash equilibria of the stationary game of Example 1}$$

$$\mathcal{E}_N \doteq \text{Set of Nash equilibria of the finite game with } N \text{ players}$$

$$\mathcal{E}_{N,\epsilon} \doteq \text{Set of } \epsilon\text{-Nash equilibria}^4 \text{ of the finite game with } N \text{ players}$$

$$\widehat{\mathcal{E}} \doteq \lim_{N\to\infty} \mathcal{E}_N = \{\mathbf{p}^* | \lim_{N\to\infty} \mathbf{1}_{\mathbf{p}^* \in \mathcal{E}_N} \text{ exists and equals } 1\}$$

$$\widetilde{\mathcal{E}} \doteq \lim_{\epsilon\to 0} \lim_{N\to\infty} \mathcal{E}_{N,\epsilon}$$

PROPOSITION 2. *The equilibria of the stationary game of Example 1 and the sequence of finite games converge in the following sense:*

$$\mathcal{E}_{sta} = \widetilde{\mathcal{E}}.$$

Proof: At $t = 0$, our queueing system is idle. We can now split the time interval $[0, \infty)$ into cycles and a residual time period – $C_1, C_2, \ldots, C_m, R$ – as follows: The cycle C_1 begins at $t = 0$ and ends when the first busy period ends, and thus includes one idle period and one busy period. Cycle C_2 begins at the end of cycle C_1 and ends when the first busy period following C_1 ends. The sequence of cycles ends when we

[4] An ϵ-Nash equilibrium is defined to be a strategy profile where no player can improve his/her utility by more than ϵ by deviating unilaterally.

run out of either the uninformed, or informed users of either size. That is, C_m denotes the last cycle such that there is at least one player of each class that hasn't arrived by the end of C_m. The remaining time period, from the end of C_m to $+\infty$ is denoted by R. The cycles $C_1, \ldots, C_m$ are thus i.i.d. renewal cycles of a stationary queueing system (the definitions of these cycles is independent of the strategies adopted by users since we have a work conserving queueing system). As $N \to \infty$, the probability that any given player arrives during a cycle approaches 1. This is because each permutation of the players is equally likely, and hence the probability that a player arrives during a renewal cycle is the same as the expected fraction of players (of his class) that arrive in some cycle. As $N \to \infty$, $m \to \infty$, while the number of users that arrive during R remains stochastically bounded. Therefore the probability that a user arrives during a renewal cycle approaches 1 as $1 - O(1/N)$.

For a fixed N, consider a tagged player U of one of the three types. Let $\mathbf{p} = \{p^{uninf}, p^{inf_1}, p^{inf_2}, p^U\}$ denote a strategy profile, where p^{uninf} denotes the (symmetric) strategy profile of the untagged uninformed users, p^{inf_1} denotes the (symmetric) strategy profile of the untagged informed users of size 1, p^{inf_2} denotes the (symmetric) strategy profile of the untagged informed users of size 2, and p^U denotes the strategy of the tagged player. For each N and for the stationary game, the delay (cost) of the tagged player is a uniformly continuous function in each component of $\mathbf{p}$ (with sup norm on the measures) since it is a continuous function with compact support. Further, as $N \to \infty$, the delay (cost) under the finite game converges uniformly to the (delay) cost under the stationary game. To see this, note that the utility (equivalently cost) of a player arriving in the residual period R is stochastically bounded above by an integrable random variable, uniformly in the strategy adopted by the player. Thus as $N \to \infty$, the utility of the tagged user U is dominated by his utility given he arrives in one of the renewal cycles. This utility, in turn, is the same as if the tagged user arrived in a stationary game where the strategy profile of the other players is given by $\{p^{uninf}, p^{inf_1}, p^{inf_2}\}$ and the tagged user adopt strategy p^U.

Thus as $N \to \infty$, the best response function of the tagged player converges point-wise to the best response function of the tagged player under the stationary game at points where the best response under the stationary game is unique. However, at points where the stationary game has multiple best responses, we may not even have convergence of the best response of the N player game as $N \to \infty$. Further, we do not necessarily have $\mathcal{E}_{sta} = \widehat{\mathcal{E}}$ (indeed, consider a hypothetical case where $\mathcal{E}_{sta} = \{1\}$, while $\mathcal{E}_N = \{1 - 1/N\}$ in which case $\widehat{\mathcal{E}} = \emptyset$). To get around this problem, we use the notion of ϵ-Nash equilibria. Due to uniform convergence of

delays, for all ϵ, there exists an N_ϵ such that $\forall\ n \geq N_\epsilon\ \mathcal{E}_{sta} \subseteq \mathcal{E}_{n,\epsilon}$. Further, for any $\mathbf{p}^* \notin \mathcal{E}_{sta}$, there is some ϵ^*, such that for all $\epsilon < \epsilon^*$, there exists an N_ϵ, such that $\mathbf{p}^* \notin \mathcal{E}_{n,\epsilon}\ \forall n > N_\epsilon$. Thus, $\mathcal{E}_{sta} = \lim_{\epsilon\to 0}\lim_{N\to\infty}\mathcal{E}_{N,\epsilon} = \widetilde{\mathcal{E}}$. □

Appendix B: Performance (Queueing) Analysis (from Section 5)

The equations presented in this Appendix are a non-trivial adaptation of Wolff (1970). The reader intending to understand the full analysis below is advised to consult Wolff first to see a simpler case. Tables 1 and 2 introduce the necessary notation and basic definitions.

α	Fraction of uninformed (IG) users
D	Number of tokens given to every user
M	Maximum size of a job
p_i	Pr[job needs i quanta]
λ_{DG}	Arrival rate of DG jobs
λ_{IG}	Arrival rate of IG jobs
ρ	Time average probability that the server is busy ($= \lambda \cdot \mathbb{E}$[job size])
$W^H_{i,DG}$	Expected delay for the ith H of a DG job with zero L
$W^{LH}_{i,DG}$	Expected delay for the ith H of a DG job with non-zero L
$W^L_{i,DG}$	Expected delay for the ith L of a DG job with non-zero L
$W^H_{i,IG}$	Expected delay for the ith H of an IG job
$W^L_{i,IG}$	Expected delay for the ith L of an IG job with non-zero L
q^H_{DG}	Given the arrival sees server busy, the server is busy with an H of a DG job
q^{LH}_{DG}	Time average probability that the server is busy with an H of a DG user with non-zero L
q^{L*}_{DG}	Time average probability that the server is busy with the last L of a DG user with non-zero L
q^H_{IG}	Time average probability that the server is busy with an H of an IG user
$n^H_{i,j,DG}$	Number of DG jobs (in H queue) of original size $i \leq D$ waiting for jth H phase
$n^L_{i,j,DG}$	Number of DG jobs (in L queue) of original size $i + D$ waiting for jth L phase
$n^{LH}_{j,DG}$	Number of DG jobs (of original size $> D$ but now in H queue) waiting for jth H phase
$n^H_{i,IG}$	Number of IG jobs (in H queue) waiting for ith H phase
$n^L_{i,IG}$	Number of IG jobs (now in L queue) waiting for ith L phase
$m^H_{k,DG}$	Number of DG jobs in H queue requiring exactly k more H phases (including the phase they are waiting for)
$m^H_{k,IG}$	Number of IG jobs in H queue requiring exactly k more H phases (including the phase they are waiting for)
R^H_{DG}	Number of remainining H quanta of a DG job (possibly including the one in service), conditioned on the server being busy with a DG job
R^L_{DG}	Number of remainining L quanta of a DG job (possibly including the one in service), conditioned on the server being busy with a DG job
R^H_{IG}	Number of remainining H quanta of an IG job (possibly including the one in service), conditioned on the server being busy with an IG job
R^L_{IG}	Number of remainining L quanta of an IG job (possibly including the one in service), conditioned on the server being busy with an IG job

Table 1 Notation

$P^H_{\geq i}$	$=$	$\sum_{j=i}^{D} p_j$
$P_{\geq i}$	$=$	$\sum_{j=i}^{M} p_j$

Table 2 Basic definitions

Let X be the original job size, and i be the quantum of interest for which we seek the delay. We will first analyze the delay of the quanta of DG jobs that get high priority, where the earlier part of the job had low priority, i.e. $X \geq i > D$. Second we will analyze the the delay of the quanta of jobs that go though the high-priority queue *only*, i.e. $X \leq D$. Finally, we will analyze the delay of quanta in the low-priority queue.

B.1. Solving for $W^{LH}_{i,DG}$

Recall that $W^{LH}_{i,DG}$ denotes the expected time spent by a DG job in the H queue while waiting for service of its ith H quantum. The expression for $W^{LH}_{i,DG}$ is given by:

$$W^{LH}_{0,DG} = 0 \tag{5}$$

$$\begin{aligned} W^{LH}_{i,DG} = {} & \lambda_{DG}\left[P^H_{\geq i}(W^{LH}_{0,DG}+1) + \ldots + P^H_{\geq 2}(W^{LH}_{i-2,DG}+1) + P^H_{\geq 1}(W^{LH}_{i-1,DG}+1)\right] \\ & + \lambda_{IG}\left[P_{\geq i}(W^{LH}_{0,DG}+1) + \ldots + P_{\geq 2}(W^{LH}_{i-2,DG}+1) + P_{\geq 1}(W^{LH}_{i-1,DG}+1)\right] \qquad 1 \leq i \leq D \end{aligned} \tag{6}$$

Note that $\lambda_{DG} \cdot P^H_{\geq i-j}(W^{LH}_{j,DG}+1)$ denotes the expected number of DG jobs of size at least $i-j$, but at most D, that arrive while the tagged job was waiting to receive, or receiving, service for its jth H quantum ($j=0$ denotes the last L quantum). Similar interpretation holds for the term $\lambda_{IG} \cdot P_{\geq i-j}(W^{LH}_{j,DG}+1)$. Since the H queue was necessarily empty when the DG job moves to the H queue, the only jobs that delay the tagged user are of the kind mentioned above.

B.2. Solving for $W^H_{i,IG} = W^H_{i,DG}$

Recall that $W^H_{i,IG}$ denotes the expected time spent by an IG job in the H queue while waiting for service of its ith H quantum. As H quanta of an IG job and of a DG job with original size $X \leq D$ are indistinguishable, $W^H_{i,IG} = W^H_{i,DG}$.

Some further definitions are introduced in Table 3. While arriving at the equations in Table 3, we have made use of Little's law to relate the number of jobs of each type in the system, to the average time spent in queue by experienced the jobs while waiting to receive service for different types of quantum. We now note that via PASTA (Poisson Arrivals See Time Average), the quantities $n^{LH}_{j,DG}$, $n^H_{i,j,DG}$, and so on, also denote the expected number of jobs of the corresponding type in the queue, as seen by an arbitrary arrival.

$W^H_{i,DG}$	=	$W^H_{i,IG}$
$n^{LH}_{j,DG}$	=	$\lambda_{DG}\ P_{\geq D+1}\ W^{LH}_{j,DG}$
$n^H_{i,j,DG}$	=	$\lambda_{DG}\ p_i\ W^H_{j,DG}$
$n^H_{j,IG}$	=	$\lambda_{IG}\ P_{\geq j}\ W^H_{j,IG}$
$m^H_{k,DG}$	=	$n^{LH}_{D-k+1,DG} + \sum_{i=k}^{D} n^H_{i,i-k+1,DG}$
$m^H_{k,IG}$	=	$\sum_{i=1}^{D-k} n^H_{i,IG} \frac{p_{k+i-1}}{P_{\geq i}} + n^H_{D-k+1,IG} \frac{P_{\geq D}}{P_{\geq D-k+1}}$
	=	$\lambda_{IG}\left(\sum_{i=1}^{D-k} W^H_{i,IG} p_{k+i-1} + W^H_{D-k+1,IG} P_{\geq D}\right)$
q^H_{DG}	=	$\frac{\lambda_{DG}}{\rho}\left(p_1 + 2\cdot p_2 + \cdots + D\cdot p_D\right)$
q^{LH}_{DG}	=	$\frac{\lambda_{DG}}{\rho} D \cdot P_{\geq D+1}$
q^{L*}_{DG}	=	$\frac{\lambda_{DG}}{\rho} P_{\geq D+1}$
q^H_{IG}	=	$\frac{\lambda_{IG}}{\rho}\left(p_1 + 2\cdot p_2 + \cdots + D\cdot P_{\geq D}\right)$
$\sum_{j=i}^{D} m^H_{j,DG}$	=	$\lambda_{DG}(W^H_{1,DG}P^H_{\geq i} + W^H_{2,DG}P^H_{\geq i+1} + \ldots + W^H_{D-i+1,DG}P^H_{\geq D}) + \lambda_{DG}P_{\geq D+1}(W^{LH}_{1,DG} + \ldots + W^{LH}_{D-i+1,DG})$
$\sum_{j=i}^{D} m^H_{j,IG}$	=	$\lambda_{IG}(W^H_{1,IG}P_{\geq i} + W^H_{2,IG}P_{\geq i+1} + \ldots + W^H_{D-i+1,IG}P_{\geq D})$

Table 3 **Definitions for $W^H_{IG} = W^H_{DG}$**

The above observations allow us to develop the following equations:

$$
\begin{aligned}
W^H_{1,DG} &= \Big(\sum_{j=1}^{D} m^H_{j,DG} + \sum_{j=1}^{D} m^H_{j,IG}\Big) + \frac{\rho}{2} \\
W^H_{2,DG} &= \Big(\sum_{j=2}^{D} m^H_{j,DG} + \sum_{j=2}^{D} m^H_{j,IG}\Big) + \lambda_{DG}P^H_{\geq 1}(W^H_{1,DG}+1) + \lambda_{IG}P_{\geq 1}(W^H_{1,DG}+1) \\
&\quad + \rho\left(q^{L*}_{DG}\cdot 1 + q^{LH}_{DG}\frac{D-1}{D} + q^H_{DG}\mathbf{Pr}\big[R^H_{DG}>1\big] + q^H_{IG}\mathbf{Pr}\big[R^H_{IG}>1\big]\right) \\
W^H_{i,DG} &= \Big(\sum_{j=i}^{D} m^H_{j,DG} + \sum_{j=i}^{D} m^H_{j,IG}\Big) + \lambda_{DG}\left[P^H_{\geq 1}(W^H_{i-1,DG}+1) + \ldots + P^H_{\geq i-1}(W^H_{1,DG}+1)\right] \\
&\quad + \lambda_{IG}\left[P_{\geq 1}(W^H_{i-1,DG}+1) + \ldots + P_{\geq i-1}(W^H_{1,DG}+1)\right] \\
&\quad + \rho\left(q^{L*}_{DG}\cdot 1 + q^{LH}_{DG}\frac{D-i+1}{D} + q^H_{DG}\mathbf{Pr}\big[R^H_{DG}>i-1\big] + q^H_{IG}\mathbf{Pr}\big[R^H_{IG}>i-1\big]\right).
\end{aligned}
$$

We briefly describe what each quantity in the expression for $W^H_{i,DG}$ means. The first paranthesis denotes the delay due to the IG and DG jobs that the tagged arrival sees in the H queue on arrival, and which had at least i remaining H quanta at that time. The second and third terms denote the delay caused by new IG and DG arrivals into the H queue while the tagged job was waiting/receiving service for its first $i-1$ H quanta. The last term denotes the contribution due to the job that was occupying the server when the tagged job first arrived into the system. This job could either be 1) a DG job serving its last L quantum, 2) a DG job of original size larger than D who was now serving an H quantum, 3) a DG job of original size at most D, or 4) and IG job serving its H quantum. While the job at the server could also be an IG job serving its L quantum, or a DG job serving some (but not last) L quantum, these jobs will not interfere in the service of the ith H quantum of the tagged job when $i>1$.

After rearranging the terms we get the following, more useful, forms:

$$W^H_{1,DG} = \lambda_{DG}(P^H_{\geq 1}W^H_{1,DG} + P^H_{\geq 2}W^H_{2,DG} + \ldots + P^H_{\geq D}W^H_{D,DG}) + \lambda_{DG}P_{\geq D+1}(W^{LH}_{1,DG} + \ldots + W^{LH}_{D,DG}) + \\ + \lambda_{IG}(P_{\geq 1}W^H_{1,IG} + P_{\geq 2}W^H_{2,IG} + \ldots + P_{\geq D}W^H_{D,IG}) + \frac{\rho}{2} \quad (7)$$

$$W^H_{i,DG} = \lambda_{DG}(P^H_{\geq i}W^H_{1,DG} + P^H_{\geq i+1}W^H_{2,DG} + \ldots + P^H_{\geq D}W^H_{D-i+1,DG}) + \lambda_{DG}(P^H_{\geq 1}W^H_{i-1,DG} + \ldots + P^H_{\geq i-1}W^H_{1,DG}) \\ + \lambda_{IG}(P_{\geq i}W^H_{1,IG} + P_{\geq i+1}W^H_{2,IG} + \ldots + P_{\geq D}W^H_{D-i+1,IG}) + \lambda_{IG}(P_{\geq 1}W^H_{i-1,DG} + \ldots + P_{\geq i-1}W^H_{1,DG}) \quad (8) \\ + \lambda_{DG}P_{\geq D+1}(W^{LH}_{1,DG} + \ldots + W^{LH}_{D-i+1,DG}) + \rho\left(q^{L*}_{DG} + q^{LH}_{DG}\frac{D-i+1}{D} + q^H_{DG} + q^H_{IG}\right) \\ 2 \leq i \leq D$$

B.3. Solving for $W^L_{i,DG}$ & $W^L_{i,IG}$.

Recall that $W^L_{i,DG}$ denotes the expected time spent by a DG job in the L queue whike waiting for service of its ith L quantum. Similarly, $W^L_{i,IG}$ denotes the expected time spent by a DG job in the L queue while waiting for service of its ith L (and hence $i+D$th overall) quantum.

Solving for $W^L_{i,DG}$.

First, let $\rho^H = \lambda_{DG}(p_1 + 2\cdot p_2 + \ldots + D\cdot p_D) + \lambda_{IG}(p_1 + 2\cdot p_2 + \ldots + (D-1)\cdot p_{D-1} + D\cdot P_{\geq D})$. Note that ρ^H denotes the load constituted by jobs that enter the High-priority queue on arrival, *only* due to their H quanta.

For a DG job in L queue, the following interferences occur:

1. An busy period of jobs in H queue it saw on arrival, and the new DG and IG arrivals into the H queue

2. After the above busy period ends, a number of IG jobs have now come into the L queue after finishing their H. To analyze this, we will need the expression for number of IG jobs of original size *at least* $D+1$ served in a busy period started by workload of size x. This is given by:

$$\frac{x\lambda_{IG}P_{\geq D+1}}{1-\rho^H}$$

3. Now in every round there are jobs finishing one quantum and generating an H busy period, and DG jobs finishing their last L and generating a longer H busy period (started by workload of D+1). All IG jobs of original size at least $D+1$ that arrive during these busy periods are now present in the L queue.

Following the line of reasoning explained above (and in the main text) we develop the following set of equations:

$$W^L_{1,DG} = B^H_0 + \frac{\sum_{j=1}^{M-D}(m^L_{j,DG} + m^L_{j,IG})}{1-\rho^H} + \frac{m^L_{1,DG}\cdot D}{1-\rho^H}$$

$n^L_{i,j,DG}$	$=$	$\lambda_{DG}\ p_{i+D}\ W^L_{j,DG}$
$n^L_{j,IG}$	$=$	$\lambda_{IG}\ P_{\geq j+D}\ W^L_{j,DG}$
$m^L_{k,DG}$	$=$	$\sum_{i=k}^{M-D} n^L_{i,i-k+1,DG}$
	$=$	$\lambda_{DG} \sum_{i=k}^{M-D} p_{i+D} W^L_{i-k+1,DG}$
$m^L_{k,IG}$	$=$	$\sum_{i=1}^{M-D-k+1} n^L_{i,IG} \frac{p_{D+k+i-1}}{P_{\geq D+i}}$
	$=$	$\lambda_{IG} \left(\sum_{i=1}^{M-D-k+1} W^L_{i,IG} p_{D+k+i-1} \right)$
$\sum_{j=i}^{M-D} m^L_{j,DG}$	$=$	$\lambda_{DG}(W^L_{1,DG} P_{\geq i+D} + W^L_{2,DG} P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,DG} P_{\geq M})$
$\sum_{j=i}^{M-D} m^L_{j,IG}$	$=$	$\lambda_{IG}(W^L_{1,IG} P_{\geq i+D} + W^L_{2,IG} P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,IG} P_{\geq M})$
$\mathbb{E}[H_{DG}]$	$=$	$\frac{1}{P^H_{\geq 1}} (p_1 + 2 \cdot p_2 + \cdots + D \cdot p_D)$
$\mathbb{E}[H^2_{DG}]$	$=$	$\frac{1}{P^H_{\geq 1}} (p_1 + 2^2 \cdot p_2 + \cdots + D^2 \cdot p_D)$
$\mathbb{E}[H_{IG}]$	$=$	$(p_1 + 2 \cdot p_2 + \cdots + D \cdot P_{\geq D})$
$\mathbb{E}[H^2_{IG}]$	$=$	$(p_1 + 2^2 \cdot p_2 + \cdots + D^2 \cdot P_{\geq D})$
B^H_0	$=$	$\frac{\rho}{1-\rho^H} \left[\frac{1}{2} + D \cdot q^{L*}_{DG} + \frac{q^H_{DG}}{2} \cdot \left(\frac{\mathbb{E}[H^2_{DG}]}{\mathbb{E}[H_{DG}]} - 1 \right) + q^{LH}_{DG} \cdot \frac{D-1}{2} + \frac{q^H_{IG}}{2} \cdot \left(\frac{\mathbb{E}[H^2_{IG}]}{\mathbb{E}[H_{IG}]} - 1 \right) \right]$
		$+ \frac{1}{1-\rho^H} \left(\sum_{i=1}^{D} i \cdot (m^H_{i,DG} + m^H_{i,IG}) \right)$

Table 4 **Definitions for** $W^L_{IG} = W^L_{DG}$

$$
\begin{aligned}
W^L_{2,DG} = & \frac{\rho^H}{1-\rho^H} + \frac{\rho(q^L_{DG}\mathbf{Pr}[R^L_{DG} > 1] + q^L_{IG}\mathbf{Pr}[R^L_{IG} > 1])}{1-\rho^H} \\
& + \frac{\rho q^L_{DG}\mathbf{Pr}[R^L_{DG} = 2] \cdot D}{1-\rho^H} \\
& + \frac{\rho q^H_{IG}\mathbf{Pr}[\text{IG job in last H}]\mathbf{Pr}[R^L_{IG} \geq 1 | \text{IG job in last H}]}{1-\rho^H} \\
& + \frac{\sum_{j=2}^{M-D}(m^L_{j,DG} + m^L_{j,IG})}{1-\rho^H} + \frac{m^L_{2,DG} \cdot D}{1-\rho^H} \\
& + \frac{\lambda_{DG} P_{\geq D+1}(W^L_{1,DG} + 1)}{1-\rho^H} + \frac{\lambda_{DG} p_{D+1}(W^L_{1,DG} + 1)D}{1-\rho^H} + \frac{\lambda_{IG} P_{\geq D+1}(W^L_{1,DG})}{1-\rho^H} \\
& + \frac{\sum_{i=1}^{D} n^H_{i,IG} \frac{P_{\geq D+1}}{P_{\geq i}}}{1-\rho^H} \\
W^L_{i,DG} = & \frac{\rho^H}{1-\rho^H} + \frac{\rho(q^L_{DG}\mathbf{Pr}[R^L_{DG} > i-1] + q^L_{IG}\mathbf{Pr}[R^L_{IG} > i-1])}{1-\rho^H} \\
& + \frac{\rho q^L_{DG}\mathbf{Pr}[R^L_{DG} = i] \cdot D}{1-\rho^H} \\
& + \frac{\rho q^H_{IG}\mathbf{Pr}[\text{IG job in last H}]\mathbf{Pr}[R^L_{IG} \geq i-1 | \text{IG job in last H}]}{1-\rho^H} \\
& + \frac{\sum_{j=i}^{M-D}(m^L_{j,DG} + m^L_{j,IG})}{1-\rho^H} + \frac{m^L_{i,DG} \cdot D}{1-\rho} \\
& + \frac{\lambda_{DG}}{1-\rho^H} \left[P_{\geq D+1}(W^L_{i-1,DG} + 1) + \ldots + P_{\geq D+i-1}(W^L_{1,DG} + 1) \right] \\
& + \frac{\lambda_{DG} \cdot D}{1-\rho^H} \left[p_{D+1}(W^L_{i-1,DG} + 1) + \ldots + p_{D+i-1}(W^L_{1,DG} + 1) \right] \\
& + \frac{\lambda_{IG}}{1-\rho^H} \left[P_{\geq D+1}(W^L_{i-1,DG} + 1) + \ldots + P_{\geq D+i-2}(W^L_{2,DG} + 1) + P_{\geq D+i-1}(W^L_{1,DG}) \right] \\
& + \frac{\sum_{j=1}^{D} n^H_{j,IG} \frac{P_{\geq D+i-1}}{P_{\geq j}}}{1-\rho^H}.
\end{aligned}
$$

These equations can be rearranged into the following useful form:

$$
\begin{aligned}
W^L_{1,DG} &= B^H_0 + \frac{\lambda_{DG}}{1-\rho^H}(W^L_{1,DG}P_{\geq 1+D} + W^L_{2,DG}P_{\geq 2+D} + \ldots + W^L_{M-D,DG}P_{\geq M}) \\
&+ \frac{\lambda_{IG}}{1-\rho^H}(W^L_{1,IG}P_{\geq 1+D} + W^L_{2,IG}P_{\geq 2+D} + \ldots + W^L_{M-D,IG}P_{\geq M}) \qquad (9) \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}(\sum_{i=1}^{M-D} p_{i+D}W^L_{i,DG}) \\
W^L_{i,DG} &= \frac{\rho^H}{1-\rho^H} + \frac{\rho q^L_{DG}}{1-\rho^H} \\
&+ \frac{\lambda_{IG}(P_{\geq D+i} + \ldots + P_{\geq M})}{1-\rho^H} \\
&+ \frac{\lambda_{DG}\cdot P_{\geq D+i}\cdot D}{1-\rho^H} \\
&+ \frac{\lambda_{IG}\cdot P_{\geq D+i-1}}{1-\rho^H} \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[W^L_{1,DG}P_{\geq i+D} + W^L_{2,DG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,DG}P_{\geq M}\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[W^L_{1,IG}P_{\geq i+D} + W^L_{2,IG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,IG}P_{\geq M}\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[W^L_{1,DG}p_{i+D} + W^L_{2,DG}p_{i+1+D} + \ldots + W^L_{M-D-i+1,DG}p_M\right] \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,DG}) + \ldots + P_{\geq D+i-1}(W^L_{1,DG})\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{i-1,DG}+1) + \ldots + p_{D+i-1}(W^L_{1,DG}+1)\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,DG}+1) + \ldots + P_{\geq D+i-2}(W^L_{2,DG}+1) + P_{\geq D+i-1}(W^L_{1,DG})\right] \\
&+ \frac{\sum_{j=1}^{D}\left(n^H_{j,IG}\frac{P_{\geq D+i-1}}{P_{\geq j}}\right)}{1-\rho^H} \\
&= \frac{\rho^H}{1-\rho^H} + \frac{\rho q^L_{DG}}{1-\rho^H} \\
&+ \frac{\lambda_{IG}(P_{\geq D+1} + \ldots + P_{\geq M})}{1-\rho^H} \\
&+ \frac{\lambda_{DG}\cdot P_{\geq D+1}\cdot D}{1-\rho^H} \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[W^L_{1,DG}P_{\geq i+D} + W^L_{2,DG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,DG}P_{\geq M}\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[W^L_{1,IG}P_{\geq i+D} + W^L_{2,IG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,IG}P_{\geq M}\right] \qquad (10) \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[W^L_{1,DG}p_{i+D} + W^L_{2,DG}p_{i+1+D} + \ldots + W^L_{M-D-i+1,DG}p_M\right] \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,DG}) + \ldots + P_{\geq D+i-1}(W^L_{1,DG})\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{i-1,DG}) + \ldots + p_{D+i-1}(W^L_{1,DG})\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,DG}) + \ldots + P_{\geq D+i-2}(W^L_{2,DG}) + P_{\geq D+i-1}(W^L_{1,DG})\right]
\end{aligned}
$$

$$+\frac{\sum_{j=1}^{D}\left(n_{j,IG}^{H}\frac{P_{\geq D+i-1}}{P_{\geq j}}\right)}{1-\rho^{H}}$$

Solving for $W_{i,IG}^{L}$: When considering the low-priority quanta of an IG job, we know that, when the quantum is about to start serving:

- all the only H DG jobs seen on arrival have left,
- all the IG jobs seen in H queue on arrival have either left or are in the L queue,
- if the job seen on arrival at the server was an IG job serving an H quantum, it has left or is in the L queue,
- if the job seen on arrival at the server was a DG job serving an H quantum, it has left,
- if the job seen on arrival at the server was a DG job serving an L quantum, it has left or is in the L queue, unless it was serving its last L quantum in which case it is in the H queue with one more remaining quantum,
- if the job seen on arrival at the server was an IG job serving an L quantum, it has left or is in the L queue,
- jobs which arrived while the tagged job was in the H queues may still be in the H queue and before the tagged job can begin service in the L queue, these arrivals will cause a busy period.

The initial workload (job at the server, and the H quanta of the jobs in the H queue) that will cause the busy period is

$$\begin{aligned}
B_0^H = {} & \rho q_{DG}^{L*}\cdot 1 \\
& + \lambda_{DG}\left\{D\cdot p_D(W_{D,IG}^H+1)\right. \\
& \quad + (D-1)\cdot[p_{D-1}(W_{D,IG}^H+1)+p_D(W_{D-1,IG}^H+1)]+\ldots \\
& \quad \left. +1\cdot[p_1(W_{D,IG}^H+1)+\ldots+p_D(W_{1,IG}^H+1)]\right\} \\
& + \lambda_{IG}\left\{[1\cdot P_{\geq D}](W_{1,IG}^H+1)\right. \\
& \quad + [1\cdot p_{D-1}+2\cdot P_{\geq D}](W_{2,IG}^H+1)+\ldots \\
& \quad \left. [1\cdot p_1+2\cdot p_2+\ldots+D\cdot P_{\geq D}](W_{D,IG}^H+1)\right\}.
\end{aligned}$$

Using the reasoning from the start of this subsection we get the following equations for $W_{i,IG}^{L}$:

$$W_{0,IG}^{L}=\sum_{j=1}^{D}(W_{j,IG}^{H}+1)-1$$

$$
\begin{aligned}
W^L_{1,IG} = & \frac{B^H_0}{1-\rho^H} + \frac{\rho(q^L_{DG}\mathbf{Pr}[R^L_{DG}>1] + q^L_{IG}\mathbf{Pr}[R^L_{IG}>1])}{1-\rho^H} \\
& + \frac{\rho q^L_{DG}\mathbf{Pr}[R^L_{DG}=2]\cdot D}{1-\rho^H} \\
& + \frac{\rho q^H_{IG}\mathbf{Pr}[R^L_{IG}\geq 1|\text{IG job in H}]}{1-\rho^H} \\
& + \frac{\sum_{j=1}^{M-D}(m^L_{j,DG}+m^L_{j,IG})}{1-\rho^H} \\
& + \frac{m^L_{1,DG}\cdot D}{1-\rho^H} \\
& + \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{0,IG}+1)\right] \\
& + \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{0,IG}+1)\right] \\
& + \frac{\sum_{j=1}^{D} n^H_{j,IG}\frac{P_{\geq D+1}}{P_{\geq j}}}{1-\rho^H} \\
W^L_{i,IG} = & \frac{\rho^H}{1-\rho^H} + \frac{\rho(q^L_{DG}\mathbf{Pr}[R^L_{DG}>i] + q^L_{IG}\mathbf{Pr}[R^L_{IG}>i])}{1-\rho^H} \\
& + \frac{\rho q^L_{DG}\mathbf{Pr}[R^L_{DG}=i+1]\cdot D}{1-\rho^H} \\
& + \frac{\rho q^H_{IG}\mathbf{Pr}[R^L_{IG}\geq i|\text{IG job in H}]}{1-\rho^H} \\
& + \frac{\sum_{j=i}^{M-D}(m^L_{j,DG}+m^L_{j,IG})}{1-\rho^H} \\
& + \frac{m^L_{i,DG}\cdot D}{1-\rho^H} \\
& + \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,IG}+1)+\ldots+P_{\geq D+i}(W^L_{0,IG}+1)\right] \\
& + \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{i-1,IG}+1)+\ldots+p_{D+i}(W^L_{0,IG}+1)\right] \\
& + \frac{\lambda_{IG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,IG}+1)+\ldots+P_{\geq D+i-1}(W^L_{1,IG}+1)+P_{\geq D+i-1}(W^L_{0,IG})\right] \\
& + \frac{\sum_{j=1}^{D}\left(n^H_{j,IG}\frac{P_{\geq D+i}}{P_{\geq j}}\right)}{1-\rho^H}.
\end{aligned}
$$

Which can be rearranged into:

$$
W^L_{0,IG} = \sum_{j=1}^{D}(W^H_{j,IG}+1) - 1 \tag{11}
$$

$$
\begin{aligned}
W^L_{1,IG} = & \frac{B^H_0}{1-\rho^H} + \frac{\rho q^L_{DG}}{1-\rho^H} \\
& + \frac{\lambda_{IG}(P_{\geq D+2}+\ldots+P_{\geq M})}{1-\rho^H} \\
& + \frac{\lambda_{IG}\cdot D\cdot P_{\geq D+1}}{1-\rho^H} \\
& + \frac{\lambda_{DG}P_{\geq D+1}\cdot D}{1-\rho^H} \\
& + \frac{\lambda_{DG}}{1-\rho^H}(W^L_{1,DG}P_{\geq 1+D} + W^L_{2,DG}P_{\geq 2+D} + \ldots + W^L_{M-D,DG}P_{\geq M})
\end{aligned} \tag{12}
$$

$$
\begin{aligned}
&+ \frac{\lambda_{IG}}{1-\rho^H}(W^L_{1,IG}P_{\geq 1+D} + W^L_{2,IG}P_{\geq 2+D} + \ldots + W^L_{M-D,IG}P_{\geq M}) \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}(\sum_{i=1}^{M-D} p_{i+D}W^L_{i,DG}) \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{0,IG}+1)\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{0,IG}+1)\right] \\
&+ \frac{\sum_{j=1}^{D} n^H_{j,IG}\frac{P_{\geq D+1}}{P_{\geq j}}}{1-\rho^H} \\
W^L_{i,IG} = &\frac{\rho^H}{1-\rho^H} + \frac{\rho q^L_{DG}}{1-\rho^H} \\
&+ \frac{\lambda_{IG}(P_{\geq D+i+1} + \ldots + P_{\geq M})}{1-\rho^H} \\
&+ \frac{\lambda_{IG}\cdot D\cdot P_{\geq D+i}}{1-\rho^H} \\
&+ \frac{\lambda_{IG}(P_{\geq D+1} + \ldots + P_{\geq D+i-1})}{1-\rho^H} \\
&+ \frac{\lambda_{DG}P_{\geq D+1}\cdot D}{1-\rho^H} \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[W^L_{1,DG}P_{\geq i+D} + W^L_{2,DG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,DG}P_{\geq M}\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[W^L_{1,IG}P_{\geq i+D} + W^L_{2,IG}P_{\geq i+1+D} + \ldots + W^L_{M-D-i+1,IG}P_{\geq M}\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[W^L_{1,DG}p_{i+D} + W^L_{2,DG}p_{i+1+D} + \ldots + W^L_{M-D-i+1,DG}p_M\right] \\
&+ \frac{\lambda_{DG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,IG}) + \ldots + P_{\geq D+i}(W^L_{0,IG})\right] \\
&+ \frac{\lambda_{DG}\cdot D}{1-\rho^H}\left[p_{D+1}(W^L_{i-1,IG}) + \ldots + p_{D+i}(W^L_{0,IG})\right] \\
&+ \frac{\lambda_{IG}}{1-\rho^H}\left[P_{\geq D+1}(W^L_{i-1,IG}) + \ldots + P_{\geq D+i-1}(W^L_{1,IG}) + P_{\geq D+i-1}(W^L_{0,IG})\right] \\
&+ \frac{\sum_{j=1}^{D}\left(n^H_{j,IG}\frac{P_{\geq D+i}}{P_{\geq j}}\right)}{1-\rho^H}
\end{aligned}
\tag{13}
$$

To get the delay of each quantum we now solve the system of equations as defined by (5) -(13). Then, expectation with respect to the job size distribution, specified by p_i needs to be taken to get the delay of an average user. Thus, the expected delay of a DG user of size x is given by:

$$
W_{DG}(x) = \begin{cases} \sum_{i=1}^{x} W^H_{i,DG} & x \leq D \\ \sum_{i=1}^{x-D} W^L_{i,DG} + \sum_{i=1}^{D} W^{LH}_{i,DG} & x > D, \end{cases}
$$

and the expected delay of an IG user of size x is given by:

$$
W_{IG}(x) = \begin{cases} \sum_{i=1}^{x} W^H_{i,IG} & x \leq D \\ \sum_{i=1}^{D} W^H_{i,IG} + \sum_{i=1}^{x-D} W^L_{i,IG} & x > D. \end{cases}
$$