Minimizing the worst slowdown: off-line and on-line

Hervé Moulin
Rice University

February 2005

Abstract

Minimizing the slowdown (expected sojourn time divided by job size) is a key concern of fairness in scheduling and queuing problems where job sizes are very heterogeneous. We look for protocols (service disciplines) capping the worst slowdown a job may face no matter how large (or small) the other jobs are. We call this worst slowdown the *liability* of the job in question.

In the **scheduling** problem (all jobs released at the same time),we show that allowing the server to randomize the order of service cuts almost in half the liability profiles feasible under deterministic protocols. The same statement holds if cash transfers are feasible and users have linear waiting costs.

In a **queuing** problem (release times of jobs are arbitrary), a deterministic **on-line** (non anticipative) protocol may guarantee the liability θ_r to job i, where r is the number of jobs in the queue when i was released, *if and only if* $\sum_1^\infty \frac{1}{\theta_r} \leq 1$. When the arrival of new jobs is Poisson with rate λ, the liability of a job of size x may not be smaller than its slowdown when all other jobs are equally long, namely $\frac{1}{1-\lambda \cdot x}$. We conjecture that this liability is feasible on-line, and identify a probabilistic protocol achieving the liability $\frac{1.45}{1-\lambda \cdot x}$.

Key words : scheduling, queuing, slowdown, probabilistic protocols

Acknowledgement 1 *Thanks to Eric Friedman,Jay Sethuraman and seminar participants at Columbia and the University of Guanajuato*

1 Individual Guarantees

Several agents share resources according to a mechanical rule of which the input is the profile of individual characteristics. The *guarantee* of a particular agent is the smallest welfare/utility level she will reach, under the worst possible configuration of other agents' characteristics. This level only depends upon this agent's own characteristics, the resources to be shared, and the number of other agents.

To a participant with no information about the actual agents with whom resources have to be shared, the guarantee is a simple measure of her downside risk, influencing both her willingness to participate in the mechanism and her perception of its fairness. Therefore an important design criteria is to improve guarantees as much as permitted by the nature of the resources being allocated. This criteria is as old as the fair division literature, and inspires for instance the familiar "I-Divide-You-Choose" mechanism[1]. More discussion in a variety of micro-economic allocation problems can be found in [5], [11], [12], [18], [20].

Here we apply the idea of maximizing individual guarantees to general scheduling and queuing problems. A single server is the shared resource. An agent's characteristics are the size (processing time) of the job he submits to the server and its release date (in the case of queuing). The expected sojourn time, from release of the job until completion by the server, is the agent's disutility.

A central debate on the management of queues bears on congestion control in the presence of "ill-behaved sources" ([5]), namely queues where the service time may vary wildly across different users. Two conflicting normative goals inspire the discussion. The first goal is utilitarian: to minimize the sum of individual sojourn times. The second is to equalize the *slowdown* (sojourn time divided by service time) of the different users (e.g., [3] [5], [7], [22]).

The familiar protocols *Shortest Job First* (SJF) [2], and *Shortest Remaining Job First* (SRJF)[3] are utilitarian optimal, but it is often argued that they are too harsh on long jobs: the profile of slowdowns is lexicographically minimal when users are ordered according to the size of their jobs. See [1],

[1]If utilities are additive over the pieces of a cake, Divide and Choose guarantees to each agent a piece worth at least half of the entire cake.

[2]in a *scheduling* problem, where all jobs are released at the same time.

[3]in a *queuing* problem, where release dates vary.

[2], [6], [8] and references therein. An exciting stream of recent research uncovers service protocols achieving a reasonable compromise between the two conflicting goals. In an M/G/1 queue the familiar *Processor Sharing* (PS) (serving all active jobs at the same rate) equalizes (expected) slowdown across all users. Not only is PS very far from utilitarian optimal, it also uses the server inefficiently given that partially completed jobs are useless. It is not hard however, to design an efficient protocol Pareto superior to PS: this is the *Fair Sojourn Processing* (FSP) formally introduced in [6] (see also [5]). FSP achieves a nearly optimal total sojourn time, while guaranteeing to every user a smaller slowdown than PS ([6], [7], [21]).[4]

In this paper we take a different route, and compare various service disciplines by the guarantees they offer to the users. That is, we focus on the worst slowdown - we call it the *liability*- that a given user may experience, where the minimum is taken over all possible distributions of job sizes and release dates for other users. We submit that in many real-life queues involving heterogenous users, such as the internet, ignorance of other users' characteristics is the norm rather than the exception. Many human queues are subject to unpredictable bursts and lapses, and the service time may differ widely across users. Insisting that the service protocol minimizes individual liabilities is the simplest way to protect individual users against the unknown, potentially very large, heterogeneity of individual demands.

We are looking for protocols guaranteeing a bounded liability, under the most parsimonious informational assumptions.

In a scheduling problem (sections 3,4) individual liability will only depend upon the number of other users, or their name. We compute the minimal feasible liability profile under a deterministic protocol, then when randomization is allowed (and users care about the expected sojourn time), and finally when cash transfers are feasible (and users have linear waiting costs). We find that randomization (Theorem 1), *or* cash transfers (Theorem 2), cut nearly in half the liability profiles feasible under deterministic protocols.

In a queuing problem (sections5,6,7), we consider on-line (non anticipative) protocols of two different types. If nothing at all is known about future jobs (number, sizes and release dates), we construct in Theorem 3 a deterministic protocol offering to any job a finite liability that only depends upon

[4]These features of FSP hold in any queue, no matter how the successive jobs are released. Yet when the arrival of new jobs is not Poisson, FSP may not result in an egalitarian, or nearly egalitarian, profile of slowdowns.

the number of live (unfinished) jobs at release time. In particular, other live jobs can be arbitrarily large, or small. An alternative assumption is that the arrival of future jobs follows a Poisson process. In the steady-state of this process, the liability only depends upon own job size and the arrival rate of jobs. We compute a lower bound for any feasible liability, conjecture that this lower bound is feasible, and offer a protocol of which the liability is at most 45% larger: Theorem 4.

2 Overview of the results

We start with a scheduling problem involving a set N of users, and illustrate the concept of liability for two benchmark protocols. Consider the *Random Order* (RO) protocol, selecting all service orderings with equal probability. If the service time of job i,$i \in N$, is x_i, the expected sojourn of job i is $x_i + \frac{1}{2}\sum_{j \neq i} x_j$. As x_j can be arbitrarily larger than x_i, the slowdown of job i is unbounded (its liability is infinite) . By contrast SJF offers the liability $n = |N|$ to every user: the worst case is when all jobs $j, j \neq i$, are barely shorter than x_i, implying that i is served last and her slowdown is n. If it is desirable to treat users unequally, we implement similarly the profile of liabilities $(\theta_i, i \in N)$ by the *Shortest Deadline First* (SDF) protocol, serving the jobs in the order of their respective "deadlines" $\theta_i x_i$. This is feasible if and only if $\sum_N \frac{1}{\theta_i} \leq 1$: Proposition 1. These bounds cannot be improved by any deterministic protocol, but they can be cut nearly in half for a *probabilistic* protocol, namely one where the server randomizes the processing order.

To see why randomization is useful here, consider n jobs of size 1. If one of these jobs is deterministically scheduled last, its slowdown is n. But if all orderings are equally probable, the common slowdown is $\frac{n+1}{2}$. More generally, the profile of liabilities $(\theta_i, i \in N)$ is probabilistically feasible if and only if $\sum_N \frac{1}{2\theta_i - 1} \leq 1$: Theorem 1. We offer a couple of simple protocols to implement these liabilities: their key property is that the expected delay imposed by job j on job i (i.e., the probability that j is scheduled before i, multiplied by x_j) never exceeds $\frac{1}{2}\frac{2\theta_i - 1}{2\theta_j - 1} x_i$. See Proposition 2.

Randomization is the easiest way to restore fairness when jobs must be processed whole. An alternative device is cash transfers: it is easy to implement, at least when each users' waiting cost per unit of time is constant and known to the server[5]. The net cost of a user is the sum of the cost of his

[5]See [4], [9], [10], [19], for a discussion of fair and incentive-compatible cash compensa-

service time and a cash transfer, and his slowdown is the ratio of net cost to the "stand alone" cost of service (when he has the server all to himself). The model with cash transfers and linear waiting costs is profoundly different from the probabilistic one. In the latter all probability distributions on the service ordering yield efficient (Pareto optimal) profiles of expected service time, whereas in the former efficiency of the profile of net costs essentially determines the service ordering. Thus it comes as a surprise that the feasible liability profiles ($\theta_i, i \in N$) in the two models are precisely the same: Theorem 2.

We turn to the queuing problem. The case of an *off-line* service protocol, where the server knows at time 0 the size and release date of all present and future jobs, is a simple extension of the scheduling model, to which our Theorems 1 and 2 extend word for word. But in most real-life queues, in particular when the users have very little information about the entire demand as assumed here, it is unlikely that the server would have perfect foresight of all future jobs (if he had, why not share this information with the users?). Then the relevant protocol must be *on-line*, i.e.,non-anticipative. We have something to say about the smallest feasible liability in two important cases.

Assume first that the server - and the users- know absolutely nothing (number, size or release date) about future jobs. None of the familiar on-line protocols discussed in the literature achieves a finite liability in that case. For instance SRJF offers no protection against the release of an arbitrarily large number of jobs shorter than my own; this is the argument, mentioned above, that while helping small jobs, SRJF unfairly penalizes large jobs [6]. The same observation applies to PS (or under its FSP improvement). For instance in the M/G/1 queue, the common slowdown under PS is unbounded if the queue is unstable[7]. And under *First Come First Serve* (FCFS), a newly-released job must wait until all live jobs are completed, no matter how large these could be. Finally under *Last Come First Serve*(LCFS), a very long job released just after me, or a burst of relatively small jobs, will increase my service time without bounds, just like under SRJF.

Using a weighted version of the FSP protocol ([6],[7]), we can nevertheless

tions in that context

[6]See [1], [22], and the recent work estimating the resulting slowdowns under several assumptions on the sizes and release dates of new jobs.

[7]With an arrival rate λ and mean job size $\overline{x}$, $\lambda\overline{x} < 1$ ensures stability and in this case the common slowdown is $\frac{1}{1-\lambda\overline{x}}$ (e.g., [23]).

cap the liability of any user as a function only of the size r of the queue (the number of live jobs) when his job is released. If $\{\theta_1, \theta_2, ...\}$ is a sequence of positive numbers such that $\sum_1^\infty \frac{1}{\theta_r} \leq 1$, our protocol guarantees the liability θ_r to any user who is released in a queue of size r. Conversely, this is only possible when the above inequality holds: Theorem 3. Taking r as a proxy of the congestion in the queue, we conclude that the liability must increase more than linearly in r, in sharp constrast with the scheduling context.

Finally we consider the much discussed context (often called the M/G/1 queue; see [23]), where the release of new jobs follows a Poisson process with rate λ known to the server (and users). We look for a finite liability that only depends upon the arrival rate λ and own job size x (we normalize the service rate to 1). As jobs can be arbitrarily large, the queue may well be unstable. If all jobs in the queue are of identical size x, the queue is stable if and only if $\lambda x < 1$, and in this case the expected slowdown of each job under a fair protocol [8] is $\frac{1}{1-\lambda x}$. Two observations follow. To a job larger than $\frac{1}{\lambda}$, we cannot offer a finite liability (depending only upon λ and x). For a job smaller than $\frac{1}{\lambda}$, the liability cannot be less than $\frac{1}{1-\lambda x}$. I conjecture that there exists a probabilistic on-line protocol achieving precisely this liability for every job $x, 0 \leq x < \frac{1}{\lambda}$. Theorem 4 describes one such protocol achieving the liability $\frac{1{\cdot}45}{1-\lambda x}$ for all x, $0 \leq x < \frac{1}{\lambda}$.

3 Probabilistic scheduling

There is a single server, with service rate normalized to 1, and the processing time of a job is deterministic[9]. A *scheduling problem* is a pair (N, x), where N is a finite set of users, and $x = (x_i, i \in N)$ is a profile of (strictly) positive job sizes. Users care only about their sojourn time, namely the date of completion of their own job (a partially completed job is useless). Therefore preemption is inefficient, jobs must be served whole. An *probabilistic* (resp. *deterministic) protocol* π associates to every problem (N, x) a random ordering $\sigma = \pi(N, x)$ of N namely a probability distribution on the set of orderings of N (resp. an ordering of N). Denoting $P(i, \sigma)$ the (random) set of agents preceding i in σ (including i itself), the expected sojourn time of

[8] Such as FCFS, LCFS, RO, or any strong and work conserving discipline treating two jobs equally if they face the same queue, see [23].

[9] All our results are preserved if the processing time of a job is random, provided the time it takes to process different jobs are stochastically independent.

job i is thus $y_i = E_\sigma[\sum_{P(i,\sigma)} x_j]$.

Given a protocol, and a problem (N,x), the expected *slowdown* of job i is $s_i(N,x) = \frac{y_i}{x_i}$. Given N, a profile $\theta = (\theta_i, i \in N)$ is a *feasible liability profile* if there exists a protocol π such that

$$s_i(N,x) \leq \theta_i \text{ for all } x \gg 0 \text{, and all } i \in N \tag{1}$$

A feasible liability profile is *minimal* if for any different feasible liability profile θ', $\theta'_i > \theta_i$ for some $i \in N$.

Our first result characterizes the feasible liability profiles when the server is restricted to deterministic protocols.

Proposition 1

Given the set N of users, the vector θ is a minimal feasible liability profile for a deterministic protocol if and only if $\sum_N \frac{1}{\theta_i} = 1$. In this case it is implemented by the "earliest deadline first" protocol, serving job i before job j only if $\theta_i x_i \leq \theta_j x_j$.

Proof

If θ is a feasible liability profile, choose $x_i = \frac{1}{\theta_i}$ for all i and apply (1) to the agent i ranked last: this gives $\sum_N \frac{1}{\theta_i} \leq 1$. Conversely, assume $\sum_N \frac{1}{\theta_i} = 1$, and apply the earliest deadline first protocol, breaking ties arbitrarily. For the problem (N,x), this protocol will select an ordering $\{1,..,n\}$ of N such that $\theta_1 x_1 \leq \theta_2 x_2 \leq ... \leq \theta_n x_n$. For any k, we have then

$$\sum_1^k x_j \leq \sum_1^k \frac{\theta_k x_k}{\theta_j} \leq \theta_k x_k$$

thus our protocol implements θ. ■

Allowing probabilistic protocols reduces by nearly 100% the minimal liability profiles.

Theorem 1

Given the set N of users, the vector θ is a minimal feasible liability profile for a probabilistic protocol if and only if $\sum_N \frac{1}{2\theta_i - 1} = 1$.

Proof

Step 1 a preliminary result. Given a problem (N,x), write $F(N,x)$ for the set of feasible profiles of expected sojourn times, namely

$$y \in F(N,x) \Leftrightarrow \text{for some random ordering } \sigma,\ y_i = E_\sigma[\textstyle\sum_{P(i,\sigma)} x_j] \text{ for all } i$$

Define for all $x \in \mathbb{R}_+^N$ and all $S \subseteq N$, the function $v(S,x) = \sum_S x_i^2 + \sum_{S(2)} x_i \cdot x_j$, where $S(2)$ is the set (with cardinality $\frac{|S|\cdot(|S|-1)}{2}$) of non ordered pairs from S. Note that v is supermodular with respect to S. The following result is proven in [16] and [15] (Lemma 1)

$$y \in F(N,x) \Leftrightarrow \{\sum_N x_i \cdot y_i = v(N,x) \text{ and } \sum_S x_i \cdot y_i \geq v(S,x) \text{ for all } S \subseteq N\}$$

Step 2 only if statement Let θ be feasible at N. For any x there exists $y \in F(N,x)$ such that $y_i \leq \theta_i x_i$, therefore by step 1

$$\sum_N \theta_i \cdot x_i^2 \geq v(N,x) \Leftrightarrow \sum_N (2\theta_i - 1) \cdot x_i^2 \geq (\sum_N x_i)^2 \tag{2}$$

For $x_i = \frac{1}{2\theta_i - 1}$, this inequality reduces to $\sum_N \frac{1}{2\theta_i - 1} \leq 1$.
Step 3 if statement. Fix θ such that $\sum_N \frac{1}{2\theta_i - 1} \leq 1$, and a problem (N,x). Because the function v is supermodular, the core of the game $(N, v(\cdot,x))$ is "large" (see [17]). That is, there exists $y \in F(N,x)$ such that $y_i \leq z_i$ for all i if and only if $\sum_S z_i \cdot x_i \geq v(S,x)$ for all $S \subseteq N$. Applying this to $z_i = \theta_i \cdot x_i$, we find that the slowdown $(\theta_i \cdot x_i)$ is feasible at (N,x) if (2) holds for all subsets S of N, including N itself. Choose S , define $u_i = \sqrt{2\theta_i - 1} \cdot x_i$, $w_i = \frac{1}{\sqrt{2\theta_i - 1}}$, for all $i \in S$, and apply Schwartz's inequality to u and w:

$$(u \cdot w)^2 = (\sum_S x_i)^2 \leq ||u||^2 \cdot ||w||^2 = (\sum_S (2\theta_i - 1) \cdot x_i^2) \cdot (\sum_S \frac{1}{2\theta_i - 1})$$

concluding the proof. ■

An important special case of the two results above is that of an anonymous liability, $\theta_i = \overline{\theta}$ for all i. For deterministic protocols, the minimal feasible liability is $\overline{\theta} = n$, and is implemented by SJF. For probabilistic protocols, it is $\overline{\theta} = \frac{n-1}{2}$, and is implemented by the anonymous version of the parametric protocols to which we now turn.

Fix the set N of users and choose for each i and job size $x_i > 0$ a cumulative distribution function $F_{i,x_i}(z)$ on $[0, +\infty[$. That is, F_{i,x_i} is any non negative, non decreasing and right-continuous function on $[0, +\infty[$ such that $\lim_\infty F_{i,x_i}(z) = 1$. Given a problem (N,x), the corresponding *parametric protocol* draws for each user i a random variable Z_i according to F_{i,x_i}, and these draws are stochastically independent. It then serves the jobs in the order of the realizations $Z_i(\omega)$: job j is served before job i if $Z_j(\omega) < Z_i(\omega)$, ties being broken by a fair coin. This rich family of protocols, which includes in particular SJF and RO, was introduced in [15], and studied there from the point of view of the strategic maneuvers of splitting and merging jobs.

Proposition 2

Given N, a feasible liability profile θ as in Theorem 1 is implemented by the following two parametric protocols:

$$\text{weighted quadratic:} \quad F_{i,x_i}(z) = \min\{z^{(2\theta_i-1)^2 x_i^2}, 1\} \text{ for } 0 \leq z < \infty,$$

$$\text{weighted serial} \;:\; F_{i,x_i}(z) = \min\{\frac{z}{(2\theta_i - 1)x_i}, 1\} \text{ for } 0 \leq z < \infty.$$

Proof

Fix (N, x) and θ such that $\sum \frac{1}{2\theta_i - 1} = 1$, and consider the weighted quadratic protocol. Write p_{ij} for the probability of the event $Z_j < Z_i$, which is easily computed as

$$p_{ij} = \frac{(2\theta_i - 1)^2 x_i^2}{(2\theta_i - 1)^2 x_i^2 + (2\theta_j - 1)^2 x_j^2} \tag{3}$$

Because the distributions F_{i,x_i} are atomless, p_{ij} is the probability that job j precedes job i. Note that it does not depend on the realizations of Z_k for $k \neq i, j$. Compute next

$$\max_{x_j} p_{ij} \cdot x_j = \frac{2\theta_i - 1}{2\theta_j - 1} \cdot \frac{x_i}{2} \tag{4}$$

from which we get

$$y_i = x_i + \sum_{N \setminus i} p_{ij} \cdot x_j \leq (1 + \frac{2\theta_i - 1}{2} \cdot (\sum_{N \setminus i} \frac{1}{2\theta_j - 1})) \cdot x_i = \theta_i \cdot x_i$$

A similar computation shows that equation (4) holds for the weighted serial protocol as well. ■

We stress that the there are many more parametric protocols implementing a given liability profile θ. The key property is (4), namely the delay i expects to incur from j is at most the RHS term in (4). Besides the quadratic and serial c.d.f.smany other choices of F_{i,x_i} achieve this.

Equation (3) justifies our quadratic terminology: the probability that j precedes i is proportional to the square of the weighted job size. For an explanation of the serial terminology, see [15].

Remark 1

Proposition 1 generalizes easily to the case where N is countable, provided we assume that job sizes are integers $x_i = 1, 2, \ldots$ Given a profile $\theta = (\theta_i, i \in N)$ such that $\sum_N \frac{1}{\theta_i} \leq 1$ and a problem (N, x), we can always enumerate $N = \{i_1, i_2, \ldots\}$ in such a way that $\theta_{i_1} x_{i_1} \leq \theta_{i_2} x_{i_2} \leq \ldots$.. Indeed the convergence of the series $\frac{1}{\theta_i}$, and $x_i \geq 1$ imply that for all a, the set of agents such that $\theta_i x_i \leq a$ is finite. Then the earliest deadline first protocol implements θ as above.

As for Theorem 1, it is preserved word for word if N is countable, and so is its proof.

4 Scheduling with cash transfers

The scheduling model in this section is very different, technically and in spirit, than the probabilistic model above. Randomization is not feasible, instead the server can perform cash transfers, provided they balance to zero, and each user has a linear disutility over sojourn time and money $\delta_i \cdot y_i - t_i$, where δ_i is user i's waiting cost per unit of time and t_i her cash transfer. Efficiency (Pareto optimality) in this context amounts to select an ordering of service minimizing the sum of individual disutilities. Such orderings always serve job i before job j if $\frac{x_i}{\delta_i} < \frac{x_j}{\delta_j}$, and this property alone guarantees efficiency ([?]).

A scheduling *problem* is now a triple (N, x, δ), where $x_i, \delta_i > 0$ for all i, and an *efficient protocol* associates to each problem (N, x, δ) a pair (σ, t), where σ is an efficient ordering and $\sum_N t_i = 0$. The resulting net waiting cost w_i, and slowdown s_i of user i are

$$w_i = \delta_i \cdot \sum_{P(i,\sigma)} x_j - t_i, \text{ and } s_i(N, x, \delta) = \frac{w_i}{\delta_i \cdot x_i}$$

As in the previous model, $\theta = (\theta_i, i \in N)$ is a feasible liability profile if there exists a protocol such that $s_i(N, x, \delta) \leq \theta_i$ for all x and i.

Theorem 2

Given the set N of users, θ is a minimal feasible liability profile for a protocol with cash transfers if and only if $\sum_N \frac{1}{2\theta_i - 1} = 1$.

Proof

Given a problem (N, x, δ), the minimal total waiting cost (achieved by any efficient ordering) is easily computed as

$$W(x, \delta) = \min_{\sigma} \sum_N w_i = \sum_N \delta_i \cdot x_i + \sum_{N(2)} \min\{\delta_j \cdot x_i, \delta_i \cdot x_j\}$$

Therefore θ is a feasible liability profile at N if and only if

$$\sum_N \theta_i \cdot \delta_i \cdot x_i \geq W(x, \delta) \text{ for all } x, \delta \gg 0 \tag{5}$$

Suppose (5) holds and pick $x = \delta$. As $W(x, x) = v(N, x)$ (defined in the proof of Theorem 1), we get $\sum_N \theta_i \cdot x_i^2 \geq v(N, x)$. Like in Step 2 of that proof, this implies $\sum_N \frac{1}{2\theta_i - 1} \leq 1$.

Conversely, we pick θ such that $\sum_N \frac{1}{2\theta_i - 1} \leq 1$ and prove (5). Change the variables δ_i to $\varepsilon_i = \frac{\delta_i}{x_i}$, so that we need to prove, for all $x, \varepsilon \gg 0$:

$$\sum_N \theta_i \cdot \varepsilon_i \cdot x_i^2 \geq \sum_N \varepsilon_i \cdot x_i^2 + \sum_{N(2)} \min\{\varepsilon_i, \varepsilon_j\} \cdot x_i \cdot x_j \tag{6}$$

When ε stays inside the cone $\{0 \leq \varepsilon_1 \leq \varepsilon_2 \leq ... \leq \varepsilon_n\}$ of $\mathbb{R}^N$, inequality (6) is linear in ε, thus it is enough to prove it for the extreme directions of this cone, namely $\varepsilon^1, \varepsilon^2, .., \varepsilon^n$, where $\varepsilon_i^k = 0$ if $i = 1, .., k$, $\varepsilon_i^k = 1$ if $i = k+1, .., n$. Fix k and set $S = \{k+1, .., n\}$. For ε^k, inequality (6) reduces to

$$\sum_S \theta_i \cdot x_i^2 \geq v(S, x)$$

which follows from $\sum_N \frac{1}{2\theta_i - 1} \leq 1$, as in Step 3 of the proof of Theorem 1. The argument is then repeated for the cones of $\mathbb{R}^N$ corresponding to other orderings of the coordinates of ε. ■

If the interpretation of theorems 1 and 2 differ, their proofs are very similar, thanks to the description of the set of feasible profiles of expected sojourn times as the core of the cooperative game $(N, v(\cdot, x))$ (step 1 in the proof of Theorem 1).

We conclude this Section with a protocol implementing the liability profile just described. As before this protocol is not the only one to do the job.

Proposition 3
Given N, a feasible liability profile θ as in Theorem 2 is implemented by the following protocol. For all $x, \delta \gg 0$:

$$\begin{aligned} w_i &= \delta_i \cdot \sum_{P(i,\sigma)} x_j - t_i \\ &= (1 + (2\theta_i - 1)^2 \cdot \sum_{j \in N \diagdown i} \frac{\min\{\delta_j \cdot x_i, \delta_i \cdot x_j\}}{(2\theta_i - 1)^2 \delta_i \cdot x_i + (2\theta_j - 1)^2 \delta_j \cdot x_j}) \cdot \delta_i \cdot x_i \end{aligned}$$

Proof
Fix i, j, x_i, δ_i, and two positive numbers α_i, α_j, then compute

$$\max_{x_j, \delta_j} \frac{\min\{\delta_j x_i, \delta_i x_j\}}{\alpha_i \delta_i x_i + \alpha_j \delta_j x_j} = \max_{x_j} \frac{x_i x_j}{\alpha_i x_i^2 + \alpha_j x_j^2} = \frac{1}{2\sqrt{\alpha_i \alpha_j}}$$

where the first equality is because the ratio increases in δ_j (resp. δ_i) if $\delta_j \cdot x_i \leq \delta_i \cdot x_j$ (resp. $\delta_j \cdot x_i \geq \delta_i \cdot x_j$). Applying this to the net cost $w_i(x, \delta)$ gives

$$\max_{x_{-i}, \delta_{-i}} w_i(x, \delta) = (1 + \frac{(2\theta_i - 1)}{2} \sum_{N \diagdown i} \frac{1}{2\theta_j - 1}) \cdot \delta_i \cdot x_i = \theta_i \cdot \delta_i \cdot x_i$$

as claimed. ■

5 Queuing: off-line protocols

We return to the probabilistic server of Section 3. A *queuing problem* is a triple (N, x, τ), where the set N of users is at most countable, and user i's job of (positive) size x_i is released at time τ_i, $\tau_i \geq 0$. When N is infinite, we assume that the number of jobs released in any bounded interval $[0, a]$ is finite, so that the queue is finite at any point in time. In this Section and the next, the profile τ of release dates is entirely arbitrary, and our results are correspondingly fully general[10].

Unlike in the scheduling problem, efficiency is now compatible with preemption. The only constraint is this: if job j preempts job i, namely the

[10]In Section 7 the release of new jobs follows the familiar Poisson random process.

server starts processing job j while job i is either untouched or partially completed, then it must wait until job j is completed before returning to (or starting service on) job i. This is equivalent to selecting a certain priority ordering of N, and serving the highest priority among the jobs alive at any point in time (e.g., [6]).

As before the *slowdown* of job i in problem (N, x, τ) is $s_i(N, x, \tau) = \frac{y_i}{x_i}$, where y_i is the expected sojourn time of job i. Given N, a profile $\theta = (\theta_i, i \in N)$ is a *feasible liability profile* if there exists a protocol such that

$$s_i(N, x, \tau) \leq \theta_i \text{ for all } x \gg 0,\ \tau \geq 0 \text{ and all } i \in N \tag{7}$$

The key to this concept is the definition of a protocol, i.e., of the information that the server can use to prioritize the jobs. We shall look below at three very different informational assumptions, of which the first one is the subject of this brief Section. The server has full information at date 0 about the size and release dates of all jobs present and future. This is often called the *off-line* context , in which a protocol can be any mapping from a problem (N, x, τ) to a (deterministic or random) priority ordering of N.

The simple observation is that the feasible liability profiles θ are precisely the same as in the scheduling context, namely Proposition 1 and Theorem 1 are preserved word for word. Indeed fix any θ and assume there exists an off-line probabilistic (resp. deterministic) protocol guaranteeing property (7). The latter holds in particular for $\tau = 0$, thus $\sum_N \frac{1}{2\theta_i - 1} \leq 1$ follows by Theorem 1 (resp. $\sum_N \frac{1}{\theta_i} \leq 1$ by Proposition 1). Conversely, if $\sum_N \frac{1}{2\theta_i - 1} \leq 1$, we implement the liability θ by adapting the weighted quadratic (or serial) protocol of Proposition 2 as follows. The realizations of the variables Z_i determine the priority ordering of jobs and a job i is preempted by any job j that draws a higher priority. Clearly under this protocol for (N, x, τ) the sojourn time of any job is not larger than in $(N, x, 0)$ under the weighted quadratic (or serial) protocol of Proposition 2, hence the claim. In the case of deterministic protocols, we adapt similarly the earliest deadline first protocol of Proposition 1.

By the same argument, Theorem 2 extends to off-line protocols in queuing problems with cash transfers. In that context a queuing problem is a 4-uple (N, x, δ, τ), where δ is the profile of waiting costs. To any such queuing problem we associate the scheduling problem $(N, x, \delta, 0)$, and write w_i^0 for the net waiting cost in this latter problem under a protocol such as the one in Proposition 3. Keep the efficient ordering σ used in $(N, x, \delta, 0)$ as the priority ordering dictating preemption in (N, x, δ, τ). Note that σ is not necessarily efficient (may not minimize total waiting cost) in (N, x, δ, τ)[11]. Clearly the resulting net waiting cost w_i is not larger than w_i^0, implying the claim.

6 Queuing: on-line protocols

We turn to the more interesting and more realistic *on-line* protocols, where the server has no information whatsoever about future jobs, so the protocol can only rely on the characteristics of live (released and not completed) jobs. In particular the server does not know which or how many jobs will be released in the future, hence the liability of job i cannot depend on its "name" within the unknown set N.

Check first that the on-line protocols commonly discussed in the literature do not offer a bounded liability. Consider first SRJF: if K jobs slightly shorter than job i are released almost immediately after τ_i, its slowdown reaches K, and this number cannot be bounded when the future is entirely opaque. A similar argument applies to LCFS and to PS, in which a burst of large jobs released just after τ_i increases the slowdown without bound. On the other hand, FCFS guarantees to any job a finite sojourn time independently of the future, yet the slowdown grows arbitrarily large with the size of the jobs in the existing queue. A similar argument applies to the queuing variant of RO, constructing the priority ordering on-line by giving to a new job an equal chance to each possible priority level among live jobs.

Yet it is feasible to achieve a finite liability for all jobs, irrespective of the size of live jobs and of the number, size and release dates of future jobs. This cap only depends upon the *number* of live jobs at release time, which we write as $r(i)$ for job i. Note that job i itself is counted in $r(i)$, and that $r(i)$ depends not only upon the problem (N, x, τ) but also the on-line protocol. In particular, $r(i)$ is random if the protocol is.

The protocol is inspired by the Fair Sojourn Processing ([6]). At any point in time where m jobs are alive, the inefficient Processor Sharing protocol serves them all at the rate $\frac{1}{m}$. FSP runs the PS protocol *virtually*, and prioritizes the jobs according to their virtual completion date under PS (an

[11]Writing $\widetilde{x_i}$ for the *remaining* service time of job i, the protocol serving at any time a job i such that $\frac{\widetilde{x_i}}{\delta_i}$ is minimal among all live jobs, may still be inefficient, though less so than the protocol just described.

earlier completion date means a higher priority). The result is an efficient deterministic protocol in which no sojourn times is longer than under PS.

In the variant of FSP establishing our next result, the server chooses a sequence $\theta_r, r = 1, 2, ..$ of positive numbers such that $\sum_1^\infty \frac{1}{\theta_r} = 1$. At any point in time, the priority ordering of live jobs follows the ordering of their completion dates under the weighted version of PS where the processing rate of job i is proportional to $\theta_{r(i)}$. We illustrate the protocol by an example with 4 jobs:

$$(x_1, \tau_1) = (10, 0); (x_2, \tau_2) = (6, 1); (x_3, \tau_3) = (4, 10); (x_4, \tau_4) = (1, 13)$$

To fix ideas we set $\theta_r = 2^r$, for all r. Starting at date 1, jobs 1 and 2 are processed at rates 2/3 and 1/3 respectively, because $\frac{\theta_1}{\theta_2} = 2$, thus if no new job is released after them, job 1 is completed at $\tau = 14.5$, before job 2 at $\tau = 19$. So job 1 has priority over job 2 and is completed at date 10. At this point job 3 is released, and $r(3) = r(2) = 2$. Note that we do not update $r(2)$ when job 1 exits. To determine the priority between jobs 2 and 3, check that under weighted PS, job 2 would have only 3 units left at date 10, and would be processed at the same rate as job 3 henceforth. Thus job 2 would complete before job 3, hence the server processes only job 2 until $\tau = 13$. At that date three jobs are alive, and according to the virtual weighted PS, only 1.5 units of job 2 are left, because its rate of service was 1/3 between $\tau = 1$ and $\tau = 10$, then 1/2 between $\tau = 10$ and $\tau = 13$. Similarly only 2.5 units of job 3 are left at date 13. The rates of service starting at $\tau = 13$ are 2/5, for jobs $1, 2$ and 1/5 for job 4. Thus job 2 has priority over job 4 and the latter over job 3. The real completion times are $\tau = 16$ for job 2, $\tau = 17$ for job 4, and $\tau = 21$ for job 3.

Theorem 3

Fix a non-decreasing sequence $\theta_r, r = 1, 2, ..$ of positive numbers such that $\sum_1^\infty \frac{1}{\theta_r} = 1$. Then there exists an efficient on-line protocol implementing the liability $\theta_{r(i)}$ for job i:

$$s_i(N, x, \tau) \leq \theta_{r(i)} \text{ for all } x \gg 0,\ \tau \geq 0 \text{ and all } i \in N \tag{8}$$

Conversely such a protocol exists only if $\sum_1^\infty \frac{1}{\theta_r} \leq 1$.

Proof

Step 1. Suppose the liability $\theta_{r(i)}$ is implemented by some deterministic protocol (on-line or otherwise). Fix an integer n, a small positive number ε and consider the problem

$$N = \{1, 2, .., n\};\ x_i = \frac{1}{\theta_i}, \tau_i = (i-1) \cdot \varepsilon \text{ for all } i$$

Choose ε so that $n \cdot \varepsilon < \min_j x_j$. If job k is the last completed one, with sojourn time y_k, we have

$$\sum_N \frac{1}{\theta_i} - \frac{n(n-1)}{2} \cdot \varepsilon \leq y_k \leq \theta_k \cdot x_k = 1$$

from which the inequality $\sum_1^\infty \frac{1}{\theta_r} \leq 1$ follows in the limit.
Step 2. We fix the sequence θ_r such that $\sum_1^\infty \frac{1}{\theta_r} = 1$. To implement (8), we construct formally the weighted version of PS illustrated before the Theorem. Let M be the set of live jobs at a given time, and consider a user i in M. Recall that $r(i)$ is the number of jobs that were alive *when job i was released*, and bears no relation to $m = |M|$. However we claim

$$\sum_M \frac{1}{\theta_{r(i)}} \leq 1. \tag{9}$$

To see this, label the jobs in M in the order of their release dates, say $\tau_1 \leq \tau_2 \leq .. \leq \tau_m$. Note that jobs $1, .., i-1$ are alive when i is released, therefore $r(i) \geq i$ for $i = 1, .., m$. The claim follows by our assumptions on the sequence θ_r.
Let PS[θ] be the (inefficient) protocol dividing its time among the jobs in M in proportion to $\frac{1}{\theta_{r(i)}}$. Inequality (9) implies that the actual rate is no less than $\frac{1}{\theta_{r(i)}}$, therefore the sojourn time of i is at most $\theta_{r(i)} \cdot x_i$, and PS[θ] implements the liability (8) as announced. Then FSP[θ] runs PS[θ] virtually and prioritizes the jobs according to their virtual completion time. This definition makes sense only if the relative ordering of any two jobs never changes when future jobs are released. This is true because in PS[θ], the ratio of processing times for any two jobs i, j is $\frac{\theta_i}{\theta_j}$ as long as they are both alive, so the ordering of their virtual completion times never changes either. Finally FSP[θ] implements the liability (8) because it is a Pareto improvement over PS[θ]. $\blacksquare$

Step 1 in the proof of Theorem 3, shows its close relation with Proposition 1. The next question is whether or not an on-line probabilistic protocols can achieve a better liability function $r \longrightarrow \theta_r$ than a deterministic one. I do not know the answer to this question. From Theorem 1 such a function has to meet $\sum_1^\infty \frac{1}{2\theta_r - 1} \leq 1$ (to see this use the same construction as in Step 1 of the above proof).

Remark 2

In the context with cash transfers and linear waiting costs, there is no hope to implement on-line a feasible liability of the type $r \longrightarrow \theta_r$. Such a protocol cannot implement the efficient cost-minimizing service ordering. The least inefficient protocol serves at a given date τ a job for which the ratio $\frac{\widetilde{x}_i}{\delta_i}$ is minimal among all live jobs, where $\widetilde{x}_i$ is the remaining service time of job i. Suppose the liability $r \longrightarrow \theta_r$ is feasible for appropriate on-line transfers, and consider a problem where $(x_1, \delta_1, \tau_1) = (a, (1-\varepsilon)a, 0)$, $(x_2, \delta_2, \tau_2) = (1, 1, \varepsilon)$, $(x_i, \delta_i, \tau_i) = (1, 1, i-1)$ for $i = 3, .., n+1$. The server gives priority to job 2 at $\tau = \varepsilon$, to job 3 at $\tau = 1 + \varepsilon$, and to job i at $\tau = i - 1 + \varepsilon$. The total waiting cost

$$w_1 + \sum_2^{n+1} w_i = (1-\varepsilon)a(a+n) + n$$

should not exceed

$$\theta_1 \cdot \delta_1 \cdot x_1 + \sum_{i=2}^{n+1} \theta_2 \cdot \delta_i \cdot x_i = (1-\varepsilon) \cdot \theta_1 \cdot a^2 + n \cdot \theta_2$$

Letting ε go to zero we get

$$(\theta_1 - 1) \cdot a^2 - n \cdot a + n \cdot (\theta_2 - 1) \geq 0 \text{ for all } a > 0, \text{ all } n = 1, 2, ..$$

This is only possible if $\theta_1 = 1$ and $\theta_2 = \infty$ (and $\theta_i = \infty$ for $i \geq 3$ as well), which is the trivial liability of FCFS.

7 On-line protocols with Poisson arrivals

We now assume that the release of new jobs follow a Poisson process with parameter λ, known to the server and the users. The size of future jobs remains unknown to the server and users. The protocol is non anticipative, and only depends on λ and the characteristics of jobs currently alive. It does not discriminate among users on the basis of their name, namely the only relevant characteristics of a job are its size and release date.

We consider a job of size x requested by user i, and released at a time where the arrival process has reached its steady state. In this Section we do not index the job size by the name of the user, because we only discuss anonymous protocols. Contrary to the previous Sections, x is now a positive number, not a vector in $\mathbb{R}^N$.

Given a protocol and an arbitrary sequence $\widetilde{x} = (.., x_{-k}, .., x_{-1}, x_1, .., x_k, ..)$ of job sizes for the predecessors (x_{-k}) and successors (x_k) of i's job, we write $y(\lambda, x, \widetilde{x})$ for the expected sojourn time of job i, and $s(\lambda, x, \widetilde{x}) = \frac{y(\lambda,x,\widetilde{x})}{x}$ for its expected slowdown. Note that the expectation only bears on the arrival process, in its steady state.

As we allow for jobs of arbitrary size, the queue may well be unstable (infinitely long in the steady state), for instance if job sizes are drawn independently form a common distribution with mean no smaller than $\frac{1}{\lambda}$. Hence we cannot guarantee a finite cap on slowdown to jobs of all sizes: think of the case where they all have the same size $x, x \geq \frac{1}{\lambda}$. The liability must now depend upon own job size as well as the arrival rate.

In order to avoid topological difficulties arising when the sequence of job sizes contains subsequences converging to $\frac{1}{\lambda}$ from below, we will restrict attention to sequences $\widetilde{x}$ for which there exists a (small) positive number ε such that no job size x_k is in $[\frac{1}{\lambda} - \varepsilon, \frac{1}{\lambda}[$. We call *regular* a sequence with this property. The proof of Theorem 4 and Proposition 7 explains the role of this assumption.

We say that $\theta : \mathbb{R}^2_{++} \longrightarrow [1, \infty]$ is a *feasible liability function* if there exists a protocol such that

$$s(\lambda, x, \widetilde{x}) \leq \theta(\lambda, x) \text{ for all } \lambda, x > 0, \text{ and all regular sequences } \widetilde{x} \qquad (10)$$

We obtain first a simple lower bound on the feasible liability.

Proposition 4
If θ is a feasible liability function, we have

$$\theta(\lambda, x) \geq \frac{1}{1 - \lambda \cdot x}, \text{ if } x < \frac{1}{\lambda}, \ \theta(\lambda, x) = \infty \text{ if } x \geq \frac{1}{\lambda}$$

Proof.
It is enough to consider the case where all job are of identical size x. If $x \geq \frac{1}{\lambda}$, the queue is unstable, namely infinitely long in the steady state, therefore the sojourn time of some jobs is infinite, implying $\theta(\lambda, x) = \infty$. If $x < \frac{1}{\lambda}$,

the mean sojourn time in any efficient protocol is $\frac{x}{1-\lambda\cdot x}$ (e.g., [22]) and it can only be longer in an inefficient protocol, thus the first inequality follows. ■

The function $\theta^*(\lambda, x) = \frac{1}{1-\lambda\cdot x}$, on $[0, \frac{1}{\lambda}[$, $= \infty$ on $[\frac{1}{\lambda}, \infty[$ is the *fair slowdown*, namely the slowdown a job can expect when all other jobs are of the same size. If I am not responsible for the size of the other jobs, this is a fair benchmark for the worst slowdown I could incur. The same idea has been used successfully in a variety of fair division problems (see [11], [12],[13]). Compute the utility I would receive when all other users have the same characteristics as me, and use it as a *lower bound* for my utility when they don't. I conjecture that the idea applies to the M/G/1 queue as well, namely θ^* is a feasible liability function. In support of this conjecture, Theorem 4 offers a probabilistic protocol implementing the liability $(1.45)\theta^*$.

On the way to this result, we compute the liability functions of several familiar on-line protoocols, and compare them to the fair slowdown.

We know from Proposition 3 that we cannot achieve a finite liability for jobs larger than $\frac{1}{\lambda}$, therefore we simply ignore (give lower priority to) these jobs in order to help secure a low liability for jobs smaller than $\frac{1}{\lambda}$.

Consider the protocols FCFS and LCFS. If we do not ignore jobs larger than $\frac{1}{\lambda}$, the liability of any job is clearly infinite[12]. Now fix a number $a, a < \frac{1}{\lambda}$, call "long" any job larger than a, "short" any other job, and consider the protocols FCFS[a], LCFS[a], in which all long jobs are systematically pushed back behind short jobs. So in FCFS[a] a new short job must only wait until all short jobs released before him are completed, and in LCFS[a] a short job is preempted only by short jobs released after him.

Proposition 5
Fix the arrival rate λ, and $a, 0 < a < \frac{1}{\lambda}$. The liability of the LCFS[a] protocol is

$$\theta^{LC}(\lambda, x) = \frac{1}{1-\lambda\cdot a}, \text{ if } x \leq a,\ \theta^{LC}(\lambda, x) = \infty \text{ if } x > a$$

The liability of FCFS[a] is

$$\theta^{FC}(\lambda, x) = (1 + \lambda\cdot a\cdot(\frac{a}{x} - 1))\cdot\theta^{LC}(\lambda, x), \text{ if } x \leq a,\ \theta^{FC}(\lambda, x) = \infty \text{ if } x > a$$

[12]If all previous jobs are longer than $\frac{1}{\lambda}$, the steady state queue is infinitely long and contains infinitely many jobs, therefore my sojourn under FCFS is unbounded. To prove the claim for LCFS: the probability that the next job released after mine shows up before my own job is completed is $1 - e^{-\lambda x} > 0$; if the size of that job grows arbitrarily large, so does my expected sojourn.

In particular $\theta^{FC}(\lambda, x) \geq \theta^{LC}(\lambda, x)$, with equality only if $x \geq a$. Moreover $\lim_{x\to 0} \frac{\theta^{FC}(\lambda,x)}{\theta^{LC}(\lambda,x)} = +\infty$.

Proof

Under FCFS[a], the worst case for a job released at time t is that all jobs released earlier be of size a, in which case the expected length of the queue upon arrival of the new job is $\frac{\lambda\cdot a^2}{1-\lambda\cdot a}$ (e.g., [?]) hence

$$y = x + \frac{\lambda\cdot a^2}{1-\lambda\cdot a} \Longrightarrow s = 1 + \frac{\lambda\cdot a}{1-\lambda\cdot a}\frac{a}{x} = (1+\lambda\cdot a\cdot(\frac{a}{x}-1))\cdot\frac{1}{1-\lambda\cdot a}$$

as claimed.

Under LCFS[a], we write $W(q)$ for the worst expected time it takes a queue of size q to clear, where the maximization bears upon the size of future jobs, who will delay the completion of the job (or sequence of jobs) of size q. Clearly $W(q)$ obtains when all future jobs are of size a, and in this case $W(q) = \frac{q}{1-\lambda\cdot a}$. Indeed the expected sojourn time in a queue where *all* jobs are of size a is $\frac{a}{1-\lambda\cdot a}$, therefore the server is idle on average the fraction $(1-\lambda\cdot a)$.of the time, and the initial job q will be processedonly when the server is idle. ■

The truncated versions of LCFS, and FCFS yield the fair slowdown only to jobs of size exactly a. For job sizes between a and $\frac{1}{\lambda}$, the liability is infinite. To job sizes in $[0, a]$, LCFS[a] guarantees a multiple of the fair slowdown, but FCFS[a] does not:

$$\sup_{x\in[0,a]} \frac{\theta^{LC}(\lambda,x)}{\theta^*(\lambda,x)} = \frac{1}{1-\lambda\cdot a}; \lim_{x\to 0}\frac{\theta^{FC}(\lambda,x)}{\theta^*(\lambda,x)} = +\infty$$

Note the following tradeoff on a: if $\lambda\cdot a$ is small, θ^{LC} is close to θ^* for short jobs (e.g., for $\lambda\cdot a = 0.25$, $\theta^{LC} \leq (1.34)\cdot\theta^*$) but the set of such short jobs is small; if $\lambda\cdot a$ is close to 1, θ^{LC} becomes much larger than θ^* for very short jobs (e.g., for $\lambda\cdot a = 0.9$, $\theta^{LC} \leq 10\cdot\theta^*$).

We turn to the SRJF protocol, important because it minimizes total sojourn time.

Proposition 6

Fix the arrival rate λ. The liability of the SRJF protocol is bounded as follows

$$1+\frac{\lambda\cdot x}{(1-\lambda\cdot x)^2} \leq \theta^{SR}(\lambda,x) \leq \frac{1}{(1-\lambda\cdot x)^2}, \text{ if } x \leq \frac{1}{\lambda}, \theta^{SR}(\lambda,x) = \infty \text{ if } x \geq \frac{1}{\lambda}$$

The longer and related proofs of Proposition 6 and Theorem 4 are in the Appendix.

Note that the bounds for θ^{SR} are fairly tight, as they coincide for $x = 0$ or 1, and their ratio never exceeds $\frac{4}{3}$.

The protocol SRJF offers a different kind of tradeoff than LCFS$[a]$ in Proposition 4. For very short jobs,SRJF ensures the fair slowdown, but for jobs near the capacity of the server, its performance is infinitely worse:

$$\lim_{x\to 0} \frac{\theta^{SR}(\lambda,x)}{\theta^*(\lambda,x)} = 1; \ \lim_{x\to\frac{1}{\lambda}} \frac{\theta^{SR}(\lambda,x)}{\theta^*(\lambda,x)} = +\infty$$

Note that on any interval $[0,a], a < \frac{1}{\lambda}$, $\sup \frac{\theta^{SR}(\lambda,x)}{\theta^*(\lambda,x)} = \frac{1}{1-\lambda\cdot a}$, just like for θ^{LC}.

In order to obtain a close approximation of the fair slowdown over the entire interval $[0,\frac{1}{\lambda}[$, we introduce a family of simple protocols, that we call *probabilistic priority* protocols. Like the parametric protocols of Section 3, each such protocol is defined by a family of cumulative distribution functions $F_x(z)$ on $[0,+\infty[$, one for each job size $x, 0 < x < \frac{1}{\lambda}$.

In the protocol we denote by $P(F)$, any job $x, x \geq \frac{1}{\lambda}$, is pushed at the back of the queue upon its release. Next consider a job of size $x, 0 < x < \frac{1}{\lambda}$, and suppose upon its release the queue is $\{x_1, x_2, .., x_K\}$, where x_k is the remaining size of the job ranked k. The protocol draws a random variable Z according to F_x, and uses its realization to insert job x in the queue. The new queue is

$$\{x_1,..,x_k,x,x_{k+1},..,x_K\} \text{ if } \sum_1^k x_j \leq Z < \sum_1^{k+1} x_j, \text{ for } k = 0,1,..,K$$

with the convention $x_{K+1} = \infty$. In particular the new job is first in the queue if $Z < x_1$, and last if $\sum_1^K x_j \leq Z$. Note also that a "short" job ($x < \frac{1}{\lambda}$) does not care about future "long" jobs, but long jobs released earlier matter because they are part of the queue in which it is inserted.

Theorem 4
It is possible to choose the c.d.f.s $x \to F_x$ in such a way that the liability θ of the probabilistic priority protocol satisfies

$$\theta(\lambda,x) \leq (1.45)\cdot\theta^*(\lambda,x) \text{ for all } x$$

One such choice is

$$F_x(z) \quad = \quad \min\{\frac{1}{x} \cdot \frac{2z}{\lambda \cdot (2z + x) + 1.25}, 1\} \text{ for all } z \geq 0,$$

$$\implies \quad \theta(\lambda, x) = (\frac{1}{4} + \frac{u}{5}) \cdot (\frac{3.25 - u}{1 - u} - \frac{\log(1 - u)}{2u}) \text{ where } u = \lambda \cdot x$$

Figure 1 depicts the ratio $\frac{\theta}{\theta^*}$ as a function of $u = \lambda \cdot x$, for the above protocol.

2.In the proof of Theorem 4, we describe a rich family of probabilistic priority protocols for which the liability function can be estimated numerically: Proposition 7 below. It is an open question to find out the optimal choice of the c.d.f.s $x \to F_x$, namely that for which the upper bound on the ratio $\frac{\theta}{\theta^*}$ is the smallest. Perhaps some choice of F implements the fair slowdown $\theta = \theta^*$.

8 Concluding comments

1.Other functional forms of the liability.

2.In the proof of Theorem 4, we describe a rich family of probabilistic priority protocols for which the liability function can be estimated numerically: Proposition 7 below. It is an open question to find out the optimal choice of the c.d.f.s $x \to F_x$, namely that for which the upper bound on the ratio $\frac{\theta}{\theta^*}$ is the smallest. Perhaps some choice of F implements the fair slowdown $\theta = \theta^*$.

3.Role of the regularity assumption on sequences $\widetilde{x}$: does $P(F)$ in Proposition 7 implement θ on X ?

4.Role of the parametric protocols in on-line context.

References

[1] Bansal, N. and Harchol-Balter, M. (2001). "Analysis of SRPT Scheduling: Investigating Unfairness," in *Proc. ACM Sigmetrics '01.*

[2] Bansal, N. and Wierman, A. (2002). "Competitive Analysis of M/GI/1 Queueing Policies," mimeo, Carnegie-Mellon University.

[3] Bender, M., Chakrabarti, S., and Muthukrishnan, S. (1998). "Flow and Stretch Metrics for Scheduling Continuous Job Streams," in *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms.*

[4] Chun, Y. (2004). "Consistency and Monotonicity in Sequencing Problems, mimeo, Seoul National University.

[5] Demers A., S. Keshav and S. Shenker (1990) "Analysis and simulation of a fair queuing algorithm", *Internetworking: Research and Experience, 1, 3-26.*

[6] Demko and T. Hill (1988). "Equitable distribution of indivisible objects", *Mathematical Social sciences, 16,2, 145-58.*

[7] Dubins, L.F. (1977). "Group Decision Devices," *Amer. Math Monthly*, May, 350-356.

[8] Friedman, E.J. and Henderson, S.G. (2003). "Fairness and Efficiency in Web Server Protocols," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 229-237, ACM Press.

[9] Friedman, E.J., Henderson, S.G., and G. Hurley, (2004) Minimizing Mean Response Time Subject to Fairness, *mimeo*, Cornell University

[10] Harchol-Balter, M., Bansal, N., Schroeder, B., and Agrawal, M. (2001). "Size-based Scheduling to Improve Web Performance," mimeo, Carnegie-Mellon University.

[11] Maniquet, F. (2003). "A Characterization of the Shapley Value in Queueing Problems," *Journal of Economic Theory*, **109**, **1**, 90-103.

[12] Mitra, M. (2001). "Mechanism Design in Queueing Problems," *Economic Theory*, **17**, 277-305.

[13] Moulin, H. (1990). "Uniform Externalities: Two Axioms for Fair Allocation," *Journal of Public Economics*, 305–326.

[14] Moulin, H. (1991). "Welfare Bounds in the Fair Division Problem," *Journal of Economic Theory*, **54**, 2, 321–337.

[15] Moulin, H. (1992). “Welfare Bounds in the Cooperative Production Problem,” *Games and Economic Behavior*, **4**, 373–401.

[16] Moulin, H. (2004). “Split-proof probabilistic scheduling,” mimeo, Rice University.

[17] Queyranne, M. (1993). "Structure of a Simple Scheduling Polyhedron," *Mathematical Programming*, **58**, 263-285.

[18] Shapley, L. (1971). Core of Convex Games, *International Journal of game Theory,* **1**,11-26.

[19] Smith, W., 1956. "Various Optimizers for Single-Stage Production", *Naval Res. Logistics Quarterly* 3, 59-66

[20] Steinhaus, H. (1948). "The Problem of Fair Division," *Econometrica* **16**, 101-104.

[21] Suijs, J. (1996). "On Incentive Compatibility and Budget Balancedness in Public Decision Making," *Economic Design*, **2**, 193.209.

[22] Thomson, W., and Varian, H. (1985). "Theories of Justice Based on Symmetry," in *Social Goals and Social Organizations* (Hurwicz *et al.*, Eds.). Cambridge: Cambridge Univ.

[23] Wierman, A. and Harchol-Balter, M. (2003). "Bounds on a Fair Policy with Near Optimal Performance," submitted.

[24] Wierman, A. and Harchol-Balter, M. (2003). "Classifying Scheduling Policies with Respect to Unfairness in an M/GI/1," in *Proceedings of ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems.*

[25] Wolff, R. W. (1989). *Stochastic modeling and the theory of queues,* Prentice-Hall series in Industrial and Systems Engineering, New Jersey

9 Appendix: Proposition 7; proofs of Proposition 6, Theorem 4

9.1 Proof of Proposition 6

Step 1. We compute first the liability of the protocol *Shortest Job First* (SJF), namely the modification of SRJF where at any time the server processes (one of) the job that was shortest *at release time.* Fix a job of size $x, x < \frac{1}{\lambda}$, at release time – dubbed "job x"–, and consider a queue of total length q in which job x is last ($q < x$ is possible: it means that job x is partially completed and is alone in the queue) and all jobs before x in the queue were shorter than x at their release time. For such a queue, define $W(q|x)$ as the worst expected time it takes the queue to clear. This is well defined because subsequent jobs preempt our queue if and only if they are shorter than x, so we do not need to specify the initial length of jobs ahead of job x. As above, the "worst" bears upon the size of future jobs. Clearly it is achieved when they are all of size barely below x, therefore our function W satisfies the following functional equation, where for simplicity we omit x in $W(q|x)$:

$$W(q) = \exp(-\lambda q) \cdot q + \int_0^q \lambda \exp(-\lambda t)\{t + W(q - t + x)\}dt$$

$$\Leftrightarrow W(q) = \frac{1 - \exp(-\lambda q)}{\lambda} + \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t) W(t+x)dt \text{ for all } q \quad (11)$$

Routine computation shows that the function

$$W(q) = \frac{q}{1 - \lambda \cdot x} \quad (12)$$

is a solution of (ab). To see that it is the only one, observe that W grows no more than linearly, again because when all future jobs are of size x, the server is idle a positive fraction of the time. Therefore for any positive number μ, the integral

$$\|W\| = \int_0^\infty |W(q)| \exp(-\mu q)dq$$

is a well defined norm for W. In the space of continuous functions for which the above integral converges, consider the linear mapping Φ

$$\Phi(W)(q) = \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t) W(t+x) dt$$

(note that Φ depends upon the job size x, fixed throughout). Check that, for an appropriate choice of μ, Φ is contracting for the corresponding norm $\|\Phi\| = \sup \frac{\|\Phi(W)\|}{\|W\|}$:

$$\|\Phi(W)\| = \int_0^\infty \lambda \exp(-\lambda q)\{|\int_0^q \exp(\lambda t) W(t+x) dt|\} \exp(-\mu q) dq$$

$$\leq \int\int_{0 \leq t \leq q < \infty} \lambda \exp(-(\lambda+\mu)q) \exp(\lambda t) |W(t+x)| dq dt$$

$$= \frac{\lambda}{\lambda+\mu} \int_0^\infty |W(t+x)| \exp(-\mu t) dt = \frac{\lambda}{\lambda+\mu} \exp(\mu x) \int_x^\infty |W(t)| \exp(-\mu t) dt$$

$$\leq \frac{\lambda}{\lambda+\mu} \exp(\mu x) \cdot \|W\|$$

Now for μ small enough, $\lambda x < 1$ implies $\frac{\lambda}{\lambda+\mu} \exp(\mu x) < 1$, establishing that Φ is contracting. As the above functional space is complete for this norm, we conclude that the equation $W = U_0 + \Phi(W)$ has at most one solution for any constant U_0.

Having determined W ((ab)), we now compute the liability of job x under SJF. Upon its release, it will be pushed behind all jobs still alive and originally shorter than x. Thus the longest queue ahead of job x when it is released, obtains when *all* past jobs were barely shorter than x. In that case the expected length of this queue (including job x) is $q = \frac{x}{1-\lambda \cdot x}$. Combining the two results, the worst expected delay for job x is $\frac{x}{(1-\lambda \cdot x)^2}$, and we conclude that the liability of SJF is $\frac{1}{(1-\lambda \cdot x)^2}$.

Step 2. Observe first that the liability of SRJF is not worse than that of SJF. In order to maximize the steady-state sojourn time of a given job of size $x, x < \frac{1}{\lambda}$, it is enough to look at the case where all past and future jobs are no larger than x. This is clear for future jobs, and for past jobs the only qualification to this statement comes if there is a single other job alive when job x is released. Then the worst case is when that job is just below x at that time, hence longer when it was released. But this additional delay

only occurs when there is a single other job alive, and requires the adversary "Nature" to predict the exact release date of job x, so we will ignore it.
Now fix a sequence $\widetilde{x} = (.., x_{-k}, .., x_{-1}, x_1, .., x_k, ..)$ of sizes all below x for the other jobs, and arbitrary release dates. Compare then the sojourn time of job x under SRJF and under SJF: under the latter, all other jobs take precedence over x, but this is not necessarily the case under the former; as soon as job x is partially processed, it concedes priority to fewer jobs under SRJF. In view of Step 1, this proves the upper bound on $\theta^{SR}(\lambda, x)$.
Step 3. Finally we compute the slowdown of job x when all other jobs past and present are just below x. The function W defined as in Step 1 solves a slightly different functional equation, because as soon as the server starts processing job x it is not interrupted by any subsequent job:

$$\begin{aligned} W(q) &= \exp(-\lambda(q-x)) \cdot q + \int_0^{q-x} \lambda \exp(-\lambda t)\{t + W(q-t+x)\}dt \text{ if } q \geq x \\ &= q \;\text{ if } 0 \leq q \leq x \end{aligned}$$

Changing the unknown function to $\widetilde{W}(\widetilde{q}) = W(\widetilde{q} + x) - x$, we find that $\widetilde{W}$ satisfies the same equation (ab2), therefore

$$W(q) = x + \frac{q-x}{1-\lambda \cdot x} \text{ if } q \geq x$$

Now, as in Step 1, the queue ahead of (and including) job x when it is released has expected length $q = \frac{x}{1-\lambda \cdot x}$, so the expected slowdown is

$$s = \frac{W(\frac{x}{1-\lambda \cdot x})}{x} = 1 + \frac{\lambda \cdot x}{(1-\lambda \cdot x)^2}$$

This concludes the proof.

9.2 Proposition 7

We fix throughout λ, the arrival rate of jobs, and

In the statement of Proposition 7, we are given a real valued function W, differentiable on $[0, +\infty[$ with the following properties:

$$W(0) = 0; \dot{W}(0) = 1; \text{ and for all } x, 0 < x < \frac{1}{\lambda},$$

$$\frac{\dot{W}(q)-1}{W(q+x)-W(q)} \text{ is strictly increasing in } q \text{ and } \lim_{q\to\infty}\frac{\dot{W}(q)-1}{W(q+x)-W(q)}=\frac{1}{x} \tag{13}$$

Given such a function W, the equation in q

$$\frac{\dot{W}(q)-1}{W(q+x)-W(q)}=\lambda$$

has a unique solution $q(x)$ for all $x, 0<x<\frac{1}{\lambda}$, and we further assume

$$\lim_{x\to\frac{1}{\lambda}} q(x)=\infty \tag{14}$$

Proposition 7

For any function W satisfying (ab) and (ab), consider the probabilistic priority protocol $P(F)$ where

$$F_x(q)=\min\{\frac{1}{\lambda}\cdot\frac{\dot{W}(q)-1}{W(q+x)-W(q)},1\} \text{ for } x>0, q\geq 0 \tag{15}$$

Then $W(q)$ is the worst expected time to clear a queue of size q under $P(F)$, and this protocol implements the liability function

$$\theta(\lambda,x)=\frac{1}{x}\{W(x)+\int_0^{q(x)}(1-\frac{1}{\lambda}\frac{\dot{W}(q)-1}{W(q+x)-W(q)})\dot{W}(q+x)dq\} \tag{16}$$

Proof

Our assumptions on W ensure that equation (ab2) defines for all λ and all x a continuous c.d.f. F_x with support $[0,q(x)]$.

We introduce first some notations . Let X be the set of sequences $\widetilde{x}$ (namely $X=\mathbb{R}_*^{\mathbb{Z}\setminus 0}$, where $\mathbb{R}_*=]0,\infty[)$, X^r be the subset of regular sequences (property (10)), and for all $a\in]0,\frac{1}{\lambda}[$, $X(a)$ be the suset of X^r such that no job size x_k is in $[a,\frac{1}{\lambda}[$. The sets $X(a)$ are nested and their union is X^r. Let $P(a,F)$ be the "$a-$truncation" of $P(F)$ where all jobs longer than a are systematically pushed at the end of the queue. Thus in $P(a,F)$ we treat a job $x, x>a$, exactly as we treat jobs $x, x\geq\frac{1}{\lambda}$ in $P(F)$.

Step 1. We write $V(q)$ (resp. $V^a(q)$) for the worst expected time to clear a queue of length q under $P(F)$ (resp. $P(a, F)$), where the maximization bears on the size of future jobs. In Steps 2 and 3 below we show that $V^a(q)$ is the following function

$$\begin{aligned} V^a(q) &= W(q) \text{ if } 0 \le q \le q(a) \\ &= W(q(a)) + \frac{q - q(a)}{1 - \lambda \cdot a} \text{ if } q(a) \le q \end{aligned} \tag{17}$$

Here we check that (ab) implies Proposition 7. Write $V(q; \widetilde{x})$ for the expected time to clear a queue of size q given a sequence $\widetilde{x}$ of (future) jobs sizes, and define similarly $V^a(q; \widetilde{x})$. Clearly $V^a(q; \widetilde{x}) = V(q; \widetilde{x})$ whenever $\widetilde{x} \in X(a)$, therefore

$$V(q) = \sup_{X^r} V(q; \widetilde{x}) = \sup_{a \in]0, \frac{1}{\lambda}[} \sup_{X(a)} V(q; \widetilde{x}) = \sup_{a \in]0, \frac{1}{\lambda}[} V^a(q) = W(q)$$

where the right hand equality follows from (ab) and assumption (ab4).

We show next that $P(F)$ implements the liability function (ab2). Fix any $x \in [0, \frac{1}{\lambda}[$, and choose a^* such that $q(a^*) \ge x + q(x)$, which is possible by (ab4). In particular $x \le a^*$. We claim now

$$\sup_{X(a)} y(\lambda, x, \widetilde{x} | P(a, F)) \le \theta(\lambda, x) \cdot x \text{ forall } a, a^* \le a < \frac{1}{\lambda}, \tag{18}$$

(recall that $y(\lambda, x, \widetilde{x})$ is the expected sojourn of job x given $\widetilde{x}$). Assume for a moment the claim holds. By construction $x \le a^* \le a$, therefore $y(\lambda, x, \widetilde{x})$ is the same under $P(a, F)$ and under $P(F)$ and (ab) holds as well when we replace $P(a, F)$ by $P(F)$. Moreover for any $\widetilde{x} \in X^r$ we can choose a such that $a^* \le a < \frac{1}{\lambda}$ and $\widetilde{x} \in X(a)$, establishing that $P(F)$ implements θ.

We now prove property (ab). Fix a as announced there, in particular $x \le a$ and $x + q(x) \le q(a)$. Upon its release, job x draws Z with distribution F_x and is placed at the back of a queue of length at most Z, resulting in a worst expected sojourn no larger than $V^a(q + Z)$. In view of (ab2) this implies

$$\sup_{X(a)} y(\lambda, x, \widetilde{x} | P(a, F)) \le \int_0^{q(x)} V^a(q + x) dF_x(q) = \int_0^{q(x)} W(q + x) dF_x(q)$$

Finally we use (ab4) and an integration by parts to compute

$$\begin{aligned}\int_0^{q(x)} W(q+x)dF_x(q) &= W(q+x) - \int_0^{q(x)} F_x(q)\dot{W}(q+x)dq \\ &= W(x) + \int_0^{q(x)} (1 - \frac{1}{\lambda}\frac{\dot{W}(q)-1}{W(q+x)-W(q)})\dot{W}(q+x)dq\end{aligned}$$

*Step 2.*We call W^a the function on the right hand side of (ab2), and in this step we show $V^a \geq W^a$ by proving that, for all q, the expected time $U^a(q)$ to clear a queue of size q when *all* future jobs are of size a is $U^a(q) = W^a(q)$. If a new job shows up when the queue size is q, the clearing time increases to $U^a(q+a)$ with probability $F_a(q)$ and is otherwise unaffected, therefore U^a satisfies the following equation

$$U^a(q) = \exp(-\lambda q) \cdot q + \int_0^q \lambda \exp(-\lambda t)\{t + H(U^a)(q-t)\}dt$$

$$\Leftrightarrow U^a(q) = \frac{1-\exp(-\lambda q)}{\lambda} + \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t)H(U^a)(t)dt \text{ for all } q \geq 0 \tag{19}$$

where H is the following (linear) operator for arbitrary U:

$$H(U)(q) = F_a(q) \cdot U(q+a) + (1 - F_a(q)) \cdot U(q) \text{ for all } q \geq 0$$

We solve this equation in two steps.

Case 1: $q \geq q(a)$

In this case $H(U)(q) = U(q+a)$, so that equation (ab) is identical to (pf6top), except for the restriction on the domain of q. We claim that its unique solution is

$$U^a(q) = U^a(q(a)) + \frac{q - q(a)}{1 - \lambda \cdot a} \text{ for all } q \geq q(a) \tag{20}$$

To prove the claim, we note that equation (ab2) implies

$$\int_0^{q(a)} \exp(\lambda t)U^a(t+a)dt = \frac{\exp(\lambda q(a))}{\lambda}\{U^a(q(a)) - \frac{1-\exp(-\lambda q(a))}{\lambda}\}$$

Then for $q \geq q(a)$, we replace the integral in the RHS of (ab2) by

$$\int_0^q \exp(\lambda t)U^a(t+a)dt = \int_0^{q(a)} \exp(\lambda t)U^a(t+a)dt + \int_{q(a)}^q \exp(\lambda t)U^a(t+a)dt$$

Using the two above equations in (ab2) a straightforward computation. gives:

$$U^a(q) - U^a(q(a)) = \frac{1-\exp\{-\lambda(q-q(a))\}}{\lambda}$$

$$+\lambda \exp\{-\lambda(q-q(a))\} \cdot \int_{q(a)}^q \exp\{\lambda(t-q(a)\}(U^a(t+a) - U^a(q(a))dt$$

which is precisely the same equation as (tppf6) up to the change of unknown function $U^*(q) = U^a(q+q(a)) - U^a(q(a))$. The same argument as in Step 1 of the proof of Proposition 6 shows that (ab) is the unique solution of (ab2) on $[q(a), \infty[$.

Case 2: $0 \leq q \leq q(a)$

In equation (ab2) the integral involves values taken by U^a at $q+a$, yet by Case1 we can regard the unknown in equation (ab2) as a continuous function defined on $[0, q(a)]$, with the convention that it extends continuously beyond this interval as the straight line with slope $\frac{1}{1-\lambda \cdot a}$. Now we check that the operator Ψ, defined for continuous functions on $[0, q(a)]$:

$$\Psi(U)(q) = \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t)H(U)(t)dt$$

is contracting for the supremum norm $\|U\|_\infty = \sup_{[0,q(a)]} |U(q)|$. Observe that for all U^1, U^2

$$\begin{aligned} |H(U^1)(t) - H(U^2)(t)| &\leq \sup\{|(U^1-U^2)(t+a)|, |(U^1-U^2)(t)|\} \\ &\leq \sup_{[0,q(a)]} |(U^1-U^2)(q)| = \left\|U^1-U^2\right\|_\infty \end{aligned}$$

where the second inequality follows from the canonical extension of both functions beyond $q(a)$. Then

$$\Psi(U^1-U^2)(q) \leq \{\lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t)dt\} \cdot \left\|U^1-U^2\right\|_\infty$$

$$= (1 - \exp(-\lambda q)) \cdot \left\|U^1 - U^2\right\|_\infty \leq (1 - \exp(-\lambda q(a))) \cdot \left\|U^1 - U^2\right\|_\infty$$

concluding the proof that Ψ is contracting, and that equation (ab2) has a unique solution on $[0.q(a)]$. It remains to check that W is this solution, which follows from

$$H(W)(q) = W(q)+F_a(q)\cdot(W(q+a)-W(q)) = W(q)+\frac{\dot{W}(q)-1}{\lambda} \text{ for all } q \in [0, q(a)]$$

and a routine computation in (ab2).
Step 3. We show $V^a \leq W^a$. We compute the worst expected time $T^a(q)$ to clear a queue of size q when the adversary "Nature" chooses the size of each future job as a function of the remaining queue size. This more than it is allowed to do under our definition of $y(\lambda, x, \widetilde{x})$ so that $V^a \leq T^a$ We will show $T^a = W^a$.

The functional equation satisfied by $T^a is$

$$T^a(q) = \exp(-\lambda q) \cdot q + \int_0^q \lambda \exp(-\lambda t)\{t + K(T^a)(q-t)\}dt$$

$$\Leftrightarrow T^a(q) = \frac{1-\exp(-\lambda q)}{\lambda} + \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t) K(T^a)(t)dt \text{ for all } q \geq 0 \tag{21}$$

where K is the following (non linear) operator for arbitrary T:

$$K(T)(q) = \sup_{x\in[0.a]} F_x(q) \cdot T(q+x) + (1 - F_x(q)) \cdot T(q) \text{ for all } q \geq 0$$

The proof parallels that of Step 2. We consider first (ab) for $q \geq q(a)$. Then $K(T)(q) = T(q+a)$ provided T is non decreasing, which is clearly true for T^a, so we are back to Case 1 in the previous Step. We conclude that T^a satisfies (ab2). Next we consider (ab) on $[0, q(a)]$, using the same convention as in Case 2 above to view the unknown as a function on $[0, q(a)]$ with a canonical extension beyond this interval given by (ab2). We define an operator Θ

$$\Theta(T)(q) = \lambda \exp(-\lambda q) \cdot \int_0^q \exp(\lambda t) K(T)(t)dt$$

and show as in case 2 that it is contracting for the supremum norm $\|T\|_\infty$. We have

$$|K(T^1)(t) - K(T^2)(t)| \leq \sup_{[0,q(a)+a]} |(T^1 - T^2)(t)| = \sup_{[0,q(a)]} |(T^1 - T^2)(t)|$$

where the last equality follows the extension property. The proof that Θ is contracting, i.e., $\|\Theta\| \leq 1 - \exp(-\lambda q(a))$, follows as above. Finally we check that the function W satisfies (ab) on $[0, q(a)]$. Fix q in this interval; the definition of F ((?)) implies

$$\begin{aligned} F_x(q) \cdot (W(q+x) - W(q)) &= \frac{\dot{W}(q) - 1}{\lambda} \text{ if } q(x) \geq q \\ &\leq \frac{\dot{W}(q) - 1}{\lambda} \text{ if } q(x) \leq q \end{aligned}$$

$$\Longrightarrow K(W)(q) = W(q) + \frac{\dot{W}(q) - 1}{\lambda} \text{ for all } q \in [0, q(a)]$$

and the desired conclusion exactly as in Case 2.

9.3 Proof of Theorem 4