Queueing Systems 44, 209–252, 2003

Queueing Dynamics and Maximal Throughput Scheduling in Switched Processing Systems *

MOR ARMONY marmony@stern.nyu.edu
Department of Information, Operations and Management Sciences, Stern School of Business, New York University, USA

NICHOLAS BAMBOS bambos@stanford.edu
Department of Management Science and Engineering, and Department of Electrical Engineering, Stanford University, USA

Received 18 December 2001; Revised 15 January 2003

Abstract. We study a processing system comprised of parallel queues, whose individual service rates are specified by a global service mode (configuration). The issue is how to switch the system between various possible service modes, so as to maximize its throughput and maintain stability under the most workload-intensive input traffic traces (arrival processes). Stability preserves the job inflow–outflow balance at each queue on the traffic traces. Two key families of service policies are shown to maximize throughput, under the mild condition that traffic traces have long-term average workload rates. In the first family of *cone policies*, the service mode is chosen based on the system backlog state belonging to a corresponding cone. Two distinct policy classes of that nature are investigated, MaxProduct and FastEmpty. In the second family of *batch policies* (BatchAdapt), jobs are collectively scheduled over adaptively chosen horizons, according to an asymptotically optimal, robust schedule. The issues of *nonpreemptive* job processing and non-negligible *switching times* between service modes are addressed. The analysis is extended to cover *feed-forward networks* of such processing systems/nodes. The approach taken unifies and generalizes prior studies, by developing a general *trace-based modeling* framework (sample-path approach) for addressing the queueing stability problem. It treats the queueing structure as a *deterministic* dynamical system and analyzes directly its evolution trajectories. It does not require any probabilistic superstructure, which is typically used in previous approaches. Probability can be superposed later to address finer performance questions (e.g., delay). The throughput maximization problem is seen to be primarily of structural nature. The developed methodology appears to have broader applicability to other queueing systems.

Keywords: switched processing systems, dynamic scheduling, throughput maximization, cone policies, adaptive batching policies, trace-based modeling

1. Introduction: model, applications, approach

Processing systems arising in surprisingly diverse application areas (computing, manufacturing, packet-switching, wireless networking, call centers, etc.) share the following simple, yet important, core feature. Each system can be *switched* between various ser-

* Early versions of some results in this paper have appeared in [1,2]. This research work was supported in part by the National Science Foundation.

vice modes at different times, while jobs queued up in FIFO buffers receive service at rates that depend exclusively on the particular mode the system is in at each point in time. We call those *Switched Processing Systems* (SPS) and focus on controlling their service modes over time to maximize throughput.

Consider a processing system comprised of $Q \in \mathbb{Z}_+$ first-in-first-out (FIFO) parallel queues (of infinite buffer capacity each) which are indexed by $q \in \mathbf{Q} = \{1, 2, \ldots, Q\}$. Jobs arrive to queue q according to an arbitrary *traffic trace* (a deterministic sequence that encompasses both the arrival and the service processes)

$$\mathcal{I}^q = \left\{\left(t_j^q, \sigma_j^q\right),\ j \in \mathbb{Z}_+\right\}, \tag{1.1}$$

where t_j^q is the arrival time of the jth job into queue q (let $0 \leqslant t_1^q \leqslant t_2^q \leqslant \cdots \leqslant t_j^q \leqslant t_{j+1}^q \leqslant \cdots$) and σ_j^q is its service requirement. The latter implies that, if the jth job were to be served at a *constant* rate r, its service time would be σ_j^q / r.[1] At any point in time, the system can be in one of $M \in \mathbb{Z}_+$ possible service modes or configurations, which are indexed by $m \in \mathbf{M} = \{1, 2, \ldots, M\}$. When it is in service mode m, the head job of the queue q receives service at rate R_m^q. Therefore, mode m is associated with the vector $\vec{R}_m = (R_m^1, R_m^2, \ldots, R_m^q, \ldots, R_m^Q) \in \mathbb{R}_+^Q$ of service rates that the queues receive when the system is in that mode.

Given the wide scope of applications (discussed later) of this model, we impose only the mildest of assumptions on the nature of each traffic trace $\mathcal{I}^q$, as follows. Let

$$\Sigma^q(s,t) = \sum_{j \in \mathbb{Z}_+} \sigma_j^q \mathbb{1}_{\{t_j^q \in (s,t]\}} \tag{1.2}$$

be the total workload (service requirement) of the jobs arriving to queue q in the time interval $(s,t]$ and form the corresponding vector $\vec{\Sigma}(s,t) = (\Sigma^1(s,t), \Sigma^2(s,t), \ldots, \Sigma^q(s,t), \ldots, \Sigma^Q(s,t))$. The expression $\mathbb{1}_{\{*\}}$ is the standard indicator function. The only *assumption* imposed on the traffic traces is that

$$\lim_{t \to \infty} \frac{\vec{\Sigma}(0,t)}{t} = \vec{\rho} > \vec{0} \tag{1.3}$$

exists, that is, we can define a long-term *traffic load* or intensity ρ^q on the input trace of each queue. The vector $\vec{\rho} = \{\rho^1, \rho^2, \ldots, \rho^q, \ldots, \rho^Q\}$ is the *load vector* of the system. It is natural to assume that each queue has positive traffic load ($\rho^q > 0$ for every $q \in \mathbf{Q}$) *and* receives positive service rate under at least one of the service vectors ($R_m^q > 0$ for at least one $m \in \mathbf{M}$, for each fixed $q \in \mathbf{Q}$). If this is not the case for some queue, this can be removed from the system, since its behavior can be easily predicted and will not substantially interact with the other queues. The degenerate service vector $\vec{0} = \{0, 0, \ldots, 0, \ldots, 0\} \in \mathbb{R}_+^Q$ is *not* included in the service modes defined above.

A *service policy* π is a rule that determines which service vector $\vec{R}_\pi(t) \in \{\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_m, \ldots, \vec{R}_M\}$ should be used at time t, or which service mode to switch the system

[1] A natural trace does not have any accumulation points, and brings only a finite amount of work into the system during any finite interval.

into, given its evolution history up to that time. Let Π be the set of all possible service policies. Those preserve the FIFO processing discipline in each individual queue, though not necessarily across different queues. Define $W_\pi^q(t)$ to be the workload in queue q at time t, when the system operates under policy $\pi \in \Pi$. This is easily seen to evolve according to the equation:

$$W_\pi^q(t) = W_\pi^q(s) + \Sigma^q(s,t) - \sum_{m=1}^{M} R_m^q \int_s^t \mathbb{1}_{\{\vec{R}_\pi(\tau)=\vec{R}_m\}} \mathbb{1}_{\{W_\pi^q(\tau)>0\}} \, \mathrm{d}\tau, \tag{1.4}$$

for all $s < t$. The workload state of the system is $\overrightarrow{W}_\pi(t) = \{W_\pi^1(t), W_\pi^2(t), \ldots, W_\pi^q(t), \ldots, W_\pi^Q(t)\}$ and the initial workload at time 0 is $\overrightarrow{W}_0$ (the same under all policies). The workload $\overrightarrow{W}_\pi(t)$ depends on the history of the traffic traces $\{\mathcal{I}^q, \ q \in \mathbf{Q}\}$ up to time t and the followed service policy $\pi \in \Pi$. In what follows, the policy π is omitted (as well as the traffic traces) in the workload notation, when it is clear from the context which particular one is used.

We focus on the practical problem of finding efficient processing policies in Π which maximize the system throughput, maintaining stability (quantified below) under traffic traces of the highest possible load. We employ the concept of rate stability used in [4,5] which is as general and practical as typically needed to address our target applications. Specifically, *rate stability* implies that at each queue the job departure rate is equal to the arrival one, capturing the basic engineering intuition about *throughput* of a processing system. To define the concept formally, suppose the system starts empty and let $d_j^q(\pi)$ be the departure time (under policy π) of the jth job arriving (at t_j^q) into queue q on trace $\mathcal{I}^q$. Since the queue follows the FIFO discipline, $\{d_j^q(\pi)/j\}$ is the mean inter-departure time of the first j jobs to depart (and arrive) and $\{t_j^q/j\}$ is their corresponding inter-arrival time. Therefore, $\{j/d_j^q(\pi)\}$ is their mean departure rate and $\{j/t_j^q\}$ their arrival one. We define the queue $q \in \mathbf{Q}$ to be rate stable under $\pi \in \Pi$ on $\mathcal{I}^q$, when

$$\lambda_{\text{out}}^q(\pi) = \frac{1}{\tau_{\text{out}}^q(\pi)} = \lim_{k\to\infty}\left\{\frac{j_k}{d_{j_k}^q(\pi)}\right\} = \lim_{k\to\infty}\left\{\frac{j_k}{t_{j_k}^q}\right\} = \frac{1}{\tau_{\text{in}}^q} = \lambda_{\text{in}}^q \tag{1.5}$$

for every increasing unbounded sequence of job indices $\{j_k\}_{k=1}^\infty$ for which the arrival rate λ_{in}^q exists. The long-term average inter-departure time is $\tau_{\text{out}}^q(\pi)$ and the corresponding inter-arrival time is τ_{in}^q. Overall, rate stability forces the departure rate λ_{out}^q to exist and be equal to the arrival one λ_{in}^q on any job sequence for which the latter exists. This captures the intuition that – in a queueing system that is not overloaded – there should be balance between the job inflow and outflow rates; otherwise, the flow deficit would accumulate in the buffer and the backlog would blow up linearly with time. In principle, the arrival rate λ_{in}^q may even depend on the chosen job index sequence.

As shown below, the condition $\lim_{t\to\infty}\{W_\pi^q(t)/t\} = 0$ implies rate stability of queue $q \in \mathbf{Q}$ under policy π. We say that the processing system is *rate stable* under policy π when *all* its queues $q \in \mathbf{Q}$ are rate stable, and *unstable* when there is at least

one that is not. Our ultimate goal is to find good service policies that satisfy the above condition for the highest possible traffic loads.

If we consider the previously described processing system as an individual node, the question arises how to analyze networks of interconnected such nodes. Fortunately, the theory developed below for the single node extends directly to *feed-forward switched processing networks*, where a job completing service at some node joins an input queue of a downstream one, according to a route specified by the flow that the job belongs to.

The described switched processing model provides a unifying framework for studying a wide variety of resource allocation mechanisms in diverse technology areas. We survey some key *applications* below. *First*, consider a wireless packet network of Q communication links sharing the same channel. Each transmitter is equipped with a FIFO buffer, where arriving packets are queued up, waiting to be transmitted to their corresponding receiver. Time is slotted. At each time slot, transmitter $q \in \mathbf{Q}$ can transmit one packet from its queue at a power level $p^q \in \mathbf{P}^q$ (where $\mathbf{P}^q$ is a discrete set of power levels). The communication links interfere with each other. When the vector of transmitter power levels is $\vec{p} = (p^1, p^2, \ldots, p^q, \ldots, p^Q)$ the probability that the packet transmitted by transmitter q is successfully received at its receiver is $R^q(\vec{p})$. Typically,

$$R^q(\vec{p}) = \frac{G^{qq} p^q}{\sum_{q' \neq q} G^{qq'} p^{q'} + \eta^q},$$

where $G^{qq'}$ is the power gain (interaction strength) between the transmitter of the q' link and the receiver of the q one, and η^q is the thermal noise at the receiver of the q link. Successful (or not) packet transmission events are independent in each time slot, hence, $R^q(\vec{p})$ is basically the service rate of queue q when the system power vector is $\vec{p}$. Thus, the global service vector $\vec{R}(\vec{p}) = (R^1(\vec{p}), R^2(\vec{p}), \ldots, R^q(\vec{p}), \ldots, R^Q(\vec{p}))$ depends on the global power vector $\vec{p}$, which takes a finite number of values in $\mathbf{P}^1 \times \mathbf{P}^2 \times \cdots \times \mathbf{P}^q \times \cdots \times \mathbf{P}^Q$ and generates a discrete set of service vectors.

A *second* interesting application, is that of packet or cell switching in high-speed communication networks. Consider the simplest example of a switch with two input $i = \{1, 2\}$ and two output $o = \{1, 2\}$ ports. Incoming packets are queued up in separate (logical) queues $i \to o$ according to the input port i they arrive at and the output port o they are destined to. Thus, there are four queues $q = \{1, 2, 3, 4\}$, with $q = 1$ for $1 \to 1$, $q = 2$ for $1 \to 2$, $q = 3$ for $2 \to 1$, and $q = 4$ for $2 \to 2$. Suppose time is slotted and in each time slot one cell (packet of size one) may be forwarded to an output port, while two cells from two distinct queues cannot be forwarded concurrently (in the same slot) to the same output port. Thus, the only allowable service vectors are $\vec{R}_1 = (1, 0, 0, 1)$, $\vec{R}_2 = (0, 1, 1, 0)$, as well as $\vec{R}_3 = (1, 0, 0, 0)$, $\vec{R}_4 = (0, 1, 0, 0)$, $\vec{R}_5 = (0, 0, 1, 0)$, $\vec{R}_6 = (0, 0, 0, 1)$. There are several *other applications* of the model, which we briefly mention below. In manufacturing systems, the service vectors may be generated by combinations of specialized workers and/or limited number tools, which are necessary to process jobs in each queue. In call/service centers, the service vectors may reflect combinations of skills of the personnel for various tasks/jobs. In database transaction systems, the service vectors may correspond to combinations of read/write

locks on data items and memory places. In parallel computation, the service vectors may correspond to combinations of processing and memory resources required by the computation tasks in each queue.

Problems of dynamic control and allocation of service resources for throughput maximization of processing systems have attracted considerable attention during the past several years in various contexts. Dynamic scheduling policies which react to instantaneous queue backlogs have been considered in several works, including [6,12,14,15, 19–21,24–29] within a Markovian, renewal, or fluid modeling framework. Another family of policies, which schedule jobs collectively over time horizons, have been considered in [4,5,16–18,22,23]. Several are referred to later in the context of our study.

The goals and structure of this paper are the following. At the *methodological level*, it aims to explore and develop a general *trace-based modeling* (TBM)[2] framework for studying the stability (capacity, throughput) problem of queueing systems. Stability is perhaps the most fundamental issue of their behavior, followed by (and closely related to) the 'finer' performance one of delay. The feasibility of the TBM framework is demonstrated for switched processing systems, but TBM is potentially applicable to other more complicated structures. It is particularly suited for directly extending the stability of an isolated node to that of a network of interacting ones (as discussed later).

Given the design pressures towards large scale queueing/processing structures in computing/networking today, which do not seem to obey traditional statistical assumptions (Poisson, Markovian, renewal, independence), the issue arises whether we can address fundamental structural questions – like that of stability – within a TBM framework in a manner largely agnostic to the fine structure of the system. Testing and performance engineering of computing/networking equipment is increasingly done experimentally, by driving simulated or real traffic traces through it and observing its response. One of the roles that theory could play here is to provide 'structural' results of ubiquitous applicability and sharp characterizations of system behavior under very 'light' assumptions, acting as reference points in the design process. The stability problem studied here within the general TBM framework is of that nature. One could eventually revert to simulation and testbeds to explore finer performance engineering questions like delay in response, knowing the performance 'envelope' established by stability.

At the *operational level*, the paper aims to demonstrate the throughput maximizing property of two major families of service/scheduling policies, using the general TBM framework:

- *Cone policies* (like MaxProduct and FastEmpty – see below);
- *Batch policies* (like BatchAdapt – see below).

They are shown to induce system stability when $\vec{\rho} \in \mathbf{S}$, where $\mathbf{S}$ is the convex hull of all service vectors (and their projections on the axes), without needing any explicit knowledge of the load vector $\vec{\rho}$.

[2] The term 'trace-based modeling' can be used interchangeably with the 'sample-path approach'. We use the former to underline the lack of need for a probability superstructure.

Cone policies partition the workload space into geometric cones, each one being associated with a set of service vectors. When the workload $\overrightarrow{W}(t)$ is in a certain cone, then the system is switched into any service vectors/mode associated with that cone. A *MaxProduct* policy structures its cones by choosing the service vectors that have maximal inner product with the workload vector. It is akin to policies studied in [12,24,27], but is substantially generalized through a geometric view, and its stability is established in the broad TBM framework in section 3.1. A *FastEmpty* policy structures its cones so that, if new job arrivals are blocked when the workload is in a certain cone, the service vectors associated with this cone can empty the system in the fastest possible time. FastEmpty could potentially be more agile [14] and mitigate delay better than MaxProduct, due to the finer optimality structure of its cones. It is described in section 3.3 and its stability proof is given in appendix A.2.

Batch policies are fundamentally different from the previous ones and are motivated primarily by those in [4], which were developed for a different system. Their specification and stability analysis are given in section 4. A *BatchAdapt* policy aggregates jobs into batches and schedules them to be executed nonpreemptively, trying to emulate an asymptotically optimal preemptive schedule. Contrary to MaxProduct and FastEmpty, a BatchAdapt policy can tolerate non-negligible *switching times* between service states without compromising throughput. It belongs to the family of dynamic scheduling-over-horizons policies [4,17,22], but the horizons are adaptively chosen. In section 5, the extension of the results to acyclic networks of switched processing nodes is discussed.

Preceding all the above, in section 2, the analytics of the TBM framework are developed in the form of general results to be leveraged later. The paper ends with some concluding remarks in section 6.

Remark 1.1 (Time-sublinearity of trace asymptotics). A recurrent theme throughout stability analysis within the TBM framework is that of 'sublinear' growth of key quantities defined on the traffic traces, as the time grows large. This culminates in the sublinear evolution of the workload vector $\lim_{t\to\infty}(\overrightarrow{W}(t)/t) = 0$ when the system is rate-stable, but is prevalent in several steps along the way. Most proofs in this paper are established by *contradiction*, doubting sublinearity on some time/event sequence and deriving a contradiction via repeated *refinements* (or thinnings) on limiting operations of that original event sequence, until a structural property of the system is violated.

Remark 1.2 (Probabilistic superstructure and rate stability). It is interesting to see how the stability problem would evolve in the presence of an imposed probabilistic superstructure on the input traces. Consider the simplest possible case of a sample space $\Omega = \{\mathcal{I}_1, \mathcal{I}_2\}$ of two trace sets $\mathcal{I}_1 = \{\mathcal{I}_1^q, q \in \mathbf{Q}\}$ and $\mathcal{I}_2 = \{\mathcal{I}_2^q, q \in \mathbf{Q}\}$ occurring with probabilities p_1 and p_2 correspondingly. The load on $\mathcal{I}_1$ is $\vec{\rho}_1$ and on $\mathcal{I}_2$ it is $\vec{\rho}_2$. Furthermore, $\vec{\rho}_1 \neq \vec{\rho}_2$ and, hence, a global 'strong law of large numbers' type framework or even an ergodic one cannot be naturally established. If a fixed service policy $\pi \in \Pi$ is used, $\overrightarrow{W}_1(t)$ is the workload at time t on $\mathcal{I}_1$ and $\overrightarrow{W}_2(t)$ on $\mathcal{I}_2$. Then, the expected

workload at t is $E[\overrightarrow{W}(t)] = p_1\overrightarrow{W}_1(t) + p_2\overrightarrow{W}_2(t)$. If both $\vec{\rho}_1 \in \mathbf{S}$ and $\vec{\rho}_2 \in \mathbf{S}$ the system is rate stable. If both $\vec{\rho}_1 \notin \mathbf{S}$ and $\vec{\rho}_2 \notin \mathbf{S}$ the system is unstable. But if $\vec{\rho}_1 \in \mathbf{S}$ and $\vec{\rho}_2 \notin \mathbf{S}$ (or $\vec{\rho}_1 \notin \mathbf{S}$ and $\vec{\rho}_2 \in \mathbf{S}$) the stability status of the system is not globally specifiable within the probabilistic framework. However, note that it is still specifiable over individual traces.

2. General preliminaries and the case of instability

In this section, we establish some general system properties, identify conditions leading to queue instability, and finally 'profile' the stability domain of the system. In the following lemmas, it is assumed that the system is driven by arbitrarily fixed traffic traces (1.1) satisfying (1.3) and that an arbitrarily fixed service policy $\pi \in \Pi$ is used, generating the workload vector $\overrightarrow{W}(t)$.

Lemma 2.1 (Trace asymptotics in expanding time windows). Consider any two increasing unbounded time sequences $\{s_n\}_{n=1}^{\infty}$ and $\{t_n\}_{n=1}^{\infty}$ such that $s_n \leqslant t_n$ for all n.

1. When $\lim_{n\to\infty}((t_n - s_n)/t_n) = \phi \in (0, 1]$ or (equivalently) $\lim_{n\to\infty}(s_n/t_n) = 1 - \phi$, the following are true:

   (a) $\displaystyle\lim_{n\to\infty} \frac{\vec{\Sigma}(s_n, t_n)}{t_n - s_n} = \vec{\rho},$

   (b) $\displaystyle\lim_{n\to\infty} \frac{\vec{\Sigma}(s_n, t_n)}{t_n} = \phi\vec{\rho}$ and, if $\phi < 1$, then $\displaystyle\lim_{n\to\infty} \frac{\vec{\Sigma}(s_n, t_n)}{s_n} = \frac{\phi}{1-\phi}\vec{\rho}.$

2. When $\lim_{n\to\infty}((t_n - s_n)/t_n) = 0$ or (equivalently) $\lim_{n\to\infty}(s_n/t_n) = 1$, the following are true:

   (a) $\displaystyle\lim_{n\to\infty} \frac{\vec{\Sigma}(s_n, t_n)}{t_n} = \lim_{n\to\infty} \frac{\vec{\Sigma}(s_n, t_n)}{s_n} = \vec{0},$

   (b) $\displaystyle\lim_{n\to\infty} \frac{\overrightarrow{W}(t_n) - \overrightarrow{W}(s_n)}{t_n} = \lim_{n\to\infty} \frac{\overrightarrow{W}(t_n) - \overrightarrow{W}(s_n)}{s_n} = \vec{0}.$

3. Finally, $\lim_{n\to\infty}((\overrightarrow{W}(t_n) - \overrightarrow{W}(t_n^-))/t_n) = \vec{0}$ (and similarly $\lim_{n\to\infty}((\overrightarrow{W}(s_n) - \overrightarrow{W}(s_n^-))/s_n) = \vec{0}$).

Lemma 2.1 characterizes the asymptotic behavior of traces in 'sliding windows' $(s_n, t_n]$ moving out to infinity and growing linearly ($t_n - s_n \approx \phi t_n$) in part 1, and sublinearly in part 2. Its proof is given in appendix A.1.

Lemma 2.2 (Asymptotics of job sizes and inter-arrival times). Let the arrival time of the first job to arrive after time t be $\tau(t) = \inf\{\tau > t\colon t_j^q = \tau \text{ for some } q \in \mathbf{Q},\ j \in \mathbb{Z}_+\}$ on the traffic traces of the various queues. Moreover, let $q(t)$ be the queue where this job arrives (choose any such queue arbitrarily if jobs arrive to more than one queue at the same time), and $j(t)$ its index. We then have:

1. $\lim_{t\to\infty}(\sigma_{j(t)}^{q(t)}/\tau(t)) = 0$. That is, job sizes can grow sublinearly (at most) in time on any traces (1.1) satisfying (1.3).[3]
2. $\lim_{t\to\infty}((\tau(t) - t)/\tau(t)) = 0$. That is, job inter-arrival times can also grow sublinearly (at most) in time.

Lemma 2.2 characterizes the job size (service requirement) and inter-arrival time asymptotics on the traffic traces of the queues. Its proof is given in appendix A.1.

Lemma 2.3 (Rate stability and workload/delay asymptotics). Let d_j^q be the departure time of the jth job to arrive into queue $q \in \mathbf{Q}$ on some arbitrarily fixed trace (1.1) satisfying (1.3), when an arbitrarily fixed service policy $\pi \in \Pi$ is used, generating workload $W^q(t)$ at time t in queue q. We then have:

(1) If $\lim_{t\to\infty}(W^q(t)/t) = 0$, then $\lambda_{\text{out}}^q = \lim_{k\to\infty}(j_k/d_{j_k}^q) = \lambda_{\text{in}}^q$ for any increasing unbounded job sequence $\{j_k\}_{k=1}^{\infty}$ such that $\lim_{k\to\infty}(j_k/t_{j_k}^q) = \lambda_{\text{in}}^q$ – that is, the job departure rate λ_{out}^q exists and is equal to the arrival one λ_{in}^q when the latter exists, so the queue is rate stable.

(2) If $\limsup_{t\to\infty}(W^q(t)/t) = \phi > 0$, then there exists an increasing unbounded job index sequence $\{j_l\}_{l=1}^{\infty}$ such that $0 \leqslant \lim_{l\to\infty}(j_l/d_{j_l}^q) < \lim_{l\to\infty}(j_l/t_{j_l}^q)$ – hence, the time-average job departure rate is less than the arrival one on that particular job sequence, and potentially on several others, so the queue is not rate stable.

(3) If $\lim_{t\to\infty}(W^q(t)/t) = 0$, then $\lim_{j\to\infty}((d_j^q - t_j^q)/t_j^q) = \lim_{j\to\infty}((d_j^q - t_j^q)/d_j^q) = 0$, that is, the sojourn time $d_j^q - t_j^q$ of the jth job through the system grows sublinearly in time.

Lemma 2.3 establishes that if $\lim_{t\to\infty}(W^q(t)/t) = 0$ *for all* $q \in \mathbf{Q}$ then the system is rate stable, as discussed in definition (1.5). The proof of this key lemma is given in appendix A.1.

The importance of all three previous lemmas becomes evident later, when they are leveraged in the proofs of key facts that follow. Recall that they are very general, in the sense that they are valid for any traffic trace and any service policy.

Let us now consider the issue of which load vectors $\vec{\rho}$ could drive the system unstable, in the sense that the workload of one or more queues could grow linearly unbounded at large times. To glean some relevant intuition, consider the case where some service policy spends a proportion of time x_m (in a long-term average sense) utilizing service vector $\vec{R}_m$, therefore, $\sum_{m=1}^{M} x_m \leqslant 1$. During the 'time deficit' $x_0 = (1 - \sum_{m=1}^{M} x_m)$, the system is in the degenerate service state $\vec{0} = (0, 0, \ldots, 0, \ldots, 0) \in \mathbb{R}_+^Q$ by convention, hence, the rate of service provided to each queue is simply 0. The long-term average service rate of queue $q \in \mathbf{Q}$ is then $\sum_{m=1}^{M} x_m R_m^q$. If the queue load is less than its average

[3] If more than one job arrives to this queue at the same time, one can similarly show that the sum of their work requirement goes to 0 when divided by $\tau(t)$.

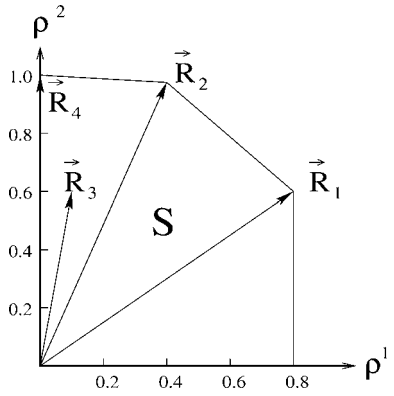

Figure 1. An example of the load set **S** of (2.1) in the case of two queues ($Q = 2$) and four service vectors: $\vec{R}_1 = (0.8,\ 0.6)$, $\vec{R}_2 = (0.4,\ 0.95)$, $\vec{R}_3 = (0.1,\ 0.6)$ and $\vec{R}_4 = (0,\ 1)$. **S** is the convex hull of $\vec{R}_1$, $\vec{R}_2$, $\vec{R}_3$, $\vec{R}_4$ (and their projections on the axes). Note that it would not change if $\vec{R}_3$ was omitted, since the latter is 'dominated' by a convex combination of the other three vectors.

service rate – that is, $\rho^q \leqslant \sum_{m=1}^{M} x_m R_m^q$ – we intuitively expect the queue to be 'stable' in some 'meaningful' way. To formalize this intuition, define the load space:

$$\mathbf{S} = \left\{ \vec{\rho} \in \mathbb{R}_+^Q\colon\ \vec{\rho} \leqslant \sum_{m=1}^{M} x_m \vec{R}_m,\ \text{for some } x_m \geqslant 0,\ m \in \mathbf{M} \text{ with } \sum_{m=1}^{M} x_m \leqslant 1 \right\} \quad (2.1)$$

where vector inequalities are considered componentwise. Therefore, **S** is the convex hull of all the service vectors $\{\vec{R}_1, \ldots, \vec{R}_m, \ldots, \vec{R}_M\}$ and their projections on the axes and the origin (for example, see figure 1). Using 'extreme value' parameters β_m (instead of x_m) that add up to 1, we can equivalently write

$$\mathbf{S} = \left\{ \vec{\rho} \in \mathbb{R}_+^Q\colon\ \vec{\rho} \leqslant \sum_{m=1}^{M} \beta_m \vec{R}_m,\ \text{for some } \beta_m \geqslant 0,\ m \in \mathbf{M} \text{ with } \sum_{m=1}^{M} \beta_m = 1 \right\}, \quad (2.2)$$

so that a load vector $\vec{\rho} \in \mathbf{S}$ is dominated by some convex combination of the service vectors. This proves to be a conceptually useful format to use later. **S** turns out to be the stability region of the system. The following proposition shows that system is certainly unstable under *any* service policy, when its load vector is not in **S**.

Proposition 2.1 (Instability). For any arbitrarily fixed traffic traces (1.1) such that $\lim_{t\to\infty}(\vec{\Sigma}(0,t)/t) = \vec{\rho} > \vec{0}$, and any service policy $\pi \in \Pi$ generating the workload $\vec{W}(t)$, we have:

$$\vec{\rho} \notin \mathbf{S} \quad \Rightarrow \quad \limsup_{t\to\infty} \frac{W^q(t)}{t} > 0 \quad \text{for some } q \in \mathbf{Q}. \quad (2.3)$$

Thus, by lemma 2.3(2), the system is not rate stable when $\vec{\rho} \notin \mathbf{S}$, irrespectively of what service policy is used.

Proof. Arguing by contradiction, assume that $\lim_{t\to\infty}(W^q(t)/t) = 0$ for *all* $q \in \mathbf{Q}$, for some arbitrarily fixed $\vec{\rho} \notin \mathbf{S}$. From (1.4) we get $W^q(t) \geqslant W^q(0) + \Sigma^q(0,t) - \sum_{m=1}^{M} R_m^q \int_0^t \mathbb{1}_{\{\vec{R}(\tau)=\vec{R}_m\}}\,\mathrm{d}\tau$ for each $q \in \mathbf{Q}$, by including the idle periods of the queue in the term that represents the received service. Therefore,

$$\overrightarrow{W}(t) \geqslant \overrightarrow{W}(0) + \vec{\Sigma}(0,t) - \sum_{m=1}^{M} \vec{R}_m \int_0^t \mathbb{1}_{\{\vec{R}(\tau)=\vec{R}_m\}}\,\mathrm{d}\tau. \tag{2.4}$$

Dividing both sides of the inequality by t, letting $t \to \infty$, using (1.3) together with the assumption that $\lim_{t\to\infty}(W^q(t)/t) = 0$, and rearranging the terms, we get

$$\sum_{m=1}^{M} \vec{R}_m \lim_{n\to\infty}\left\{\frac{1}{t_n}\int_0^{t_n} \mathbb{1}_{\{\vec{R}(\tau)=\vec{R}_m\}}\,\mathrm{d}\tau\right\} \geqslant \vec{\rho} \tag{2.5}$$

for some increasing unbounded time sequence $\{t_n\}_{n=1}^{\infty}$. Such a time sequence can be directly constructed by successively thinning convergent subsequences of the limit terms for consecutive $m \in \mathbf{M}$. In view of the above, setting

$$\lim_{n\to\infty}\left\{\frac{1}{t_n}\int_0^{t_n} \mathbb{1}_{\{\vec{R}(\tau)=\vec{R}_m\}}\,\mathrm{d}\tau\right\} = x_m \geqslant 0,$$

we get

$$\sum_{m=1}^{M} \vec{R}_m x_m \geqslant \vec{\rho} \quad \text{with } \sum_{m=1}^{M} x_m \leqslant 1,$$

which contradicts the fact that $\vec{\rho} \notin \mathbf{S}$. □

Proposition 2.1 shows that the workload of at least one queue blows up linearly in time on at least one time sequence when $\vec{\rho} \notin \mathbf{S}$. The following lemma demonstrates how this impacts the emptying times of the queue.

Lemma 2.4 (Emptying times of unstable queues). Suppose that for some $q \in \mathbf{Q}$ we have $\lim_{k\to\infty}(W^q(t_k)/t_k) = \eta^q$ on some increasing unbounded time sequence $\{t_k\}_{k=1}^{\infty}$ for some arbitrarily fixed traces (1.1) satisfying (1.3) and an arbitrarily fixed service policy $\pi \in \Pi$ generating the workload $W^q(t)$. Let

$$s_k^q = \sup\bigl\{t < t_k\colon\ W^q(t) = 0\bigr\} \tag{2.6}$$

be the *last* time before t_k that the queue was *empty*. By convention, we set $s_k^q = 0$, if this queue has never been empty in $(0, t_k]$. Then,

$$\lim_{k\to\infty} \frac{W^q(t_k)}{t_k} = \eta^q > 0 \quad \Rightarrow \quad \liminf_{k\to\infty} \frac{t_k - s_k^q}{t_k} = \varepsilon^q > 0, \tag{2.7}$$

for some $\varepsilon^q \in (0, 1]$. Note that the queue is never empty in the intervals $(s_k^q, t_k]$.

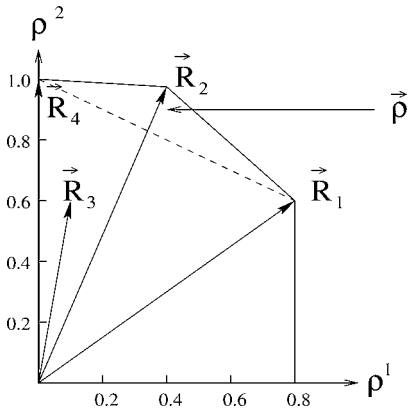

Figure 2. Maximizing the instantaneous throughput might not stabilize the queueing system. This is an example of the situation described in remark 2.1 for the system of figure 1. In this case, in order to maximize the instantaneous throughput one should use $\vec{R}_1$ when $W^1 > 0$ and $\vec{R}_4$ otherwise. Suppose now that $\vec{\rho} = 0.9\vec{R}_2$, so $\vec{\rho} \in \mathbf{S}$. However, since only $\vec{R}_1$ and $\vec{R}_4$ are used by the service policy and $\vec{\rho}$ is not within the convex hull of the latter two vectors, the system is not rate stable.

Lemma 2.4 shows that, if the workload diverges linearly ($W^q(t_k) \approx \eta^q t_k$) in a queue, then the time since the queue was last empty must also diverge linearly ($t_k - s_k^q \approx \varepsilon^q t_k$). Its proof is given in appendix A.1.

Since proposition 2.1 reveals that there is *no* service policy that would maintain system stability when $\vec{\rho} \notin \mathbf{S}$, the question arises whether there are any families of service policies that would keep the system rate stable when $\vec{\rho} \in \mathbf{S}$. Indeed, identifying such *efficient* policies is the core issue addressed in the following sections. However, before turning to that, let us first develop some relevant intuition in the following remarks.

Remark 2.1 (Maximizing the instantaneous throughput might not stabilize the queueing system). One might think that a service policy maintaining system stability when $\vec{\rho} \in \mathbf{S}$ could be one that uses a service vector maximizing the instantaneous throughput $\max_{m \in \mathbf{M}} \{\sum_{q=1}^{Q} R_m^q \mathbb{1}_{\{W^q > 0\}}\}$ when the workload vector is $\vec{W}$ at time t, that is, maximizes the *total instantaneous* service rate of non-empty queues. It turns out that this is not true, as the counter-example of figure 2 demonstrates. The problem is that this policy is 'myopic' and does not take into account operational entanglements between queues occurring over time.

Remark 2.2 (Emptying the queues optimally – the static case). Some useful intuition can be obtained by considering the following scenario. Suppose there are no job arrivals, but some initial workload w^q is placed in each queue $q \in \mathbf{Q}$. The issue is to select a service policy to empty *all* queues jointly in the minimum possible time (allowing service preemption). This is easy to do, operating the system for time x_m using service vector $\vec{R}_m$ and minimizing the total time $\sum_{m=1}^{M} x_m$ to empty the system by selecting the

x_m parameters to be an optimal solution of the linear program:

$$\begin{aligned} &\text{minimize} \sum_{m=1}^{M} x_m \\ &\text{subject to:} \quad \mathcal{R}\vec{x} \geqslant \vec{w}, \quad \vec{x} \geqslant 0, \end{aligned} \tag{LP$_{\vec{w}}$}$$

where vector inequalities are considered componentwise, $\mathcal{R}$ is the $Q \times M$ matrix whose columns are the service vectors $\{\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_m, \ldots, \vec{R}_M\}$, and $\vec{x} = (x_1, x_2, \ldots, x_m, \ldots, x_M)$. If $\vec{x}^*(\vec{w}) = (x_1^*, x_2^*, \ldots, x_m^*, \ldots, x_M^*)$ is an optimal solution of (LP$_{\vec{w}}$), the total minimal time to empty all queues is

$$\delta(\vec{w}) = \sum_{m=1}^{M} x_m^*, \tag{2.8}$$

and is attainable by a policy using service vector $\vec{R}_1$ for time $x_1^*, \ldots,$ $\vec{R}_m$ for time $x_m^*, \ldots,$ and $\vec{R}_M$ for time x_M^*.

Remark 2.3 (Time division multiplexing policy – TDM). Suppose that $\vec{\rho} \in \mathbf{S}$, hence, from (2.1) there are some $x_m \geqslant 0$, $m \in \{1, 2, \ldots, M\}$ with $\sum_{m=1}^{M} x_m \leqslant 1$, such that $\vec{\rho} \leqslant \sum_{m=1}^{M} x_m \vec{R}_m$. Consider now finding specific x-values that *minimize* $\sum_{m=1}^{M} x_m$ as an objective function, *subject to* $\vec{\rho} \leqslant \sum_{m=1}^{M} x_m \vec{R}_m$ with $x_m \geqslant 0$ for every $m \in \mathbf{M}$. This is a standard linear program (LP$_{\vec{\rho}}$) and can be solved to obtain a minimizing vector $\vec{x}^*(\vec{\rho}) = (x_1^*, x_1^*, \ldots, x_m^*, \ldots, x_M^*)$ whose elements will depend on $\vec{\rho}$ through the inequality it is subjected to. Let then $\delta(\vec{\rho}) = \sum_{m=1}^{M} x_m^*$ and note that

$$\vec{\rho} \in \mathbf{S} \quad \Leftrightarrow \quad \delta(\vec{\rho}) \leqslant 1. \tag{2.9}$$

Taking a time-division multiplexing (TDM) approach, split now time into consecutive frames of size 1 each, and in each time frame spend time x_1^* using service vector $\vec{R}_1$, x_2^* using $\vec{R}_2, \ldots, x_m^*$ using $\vec{R}_m, \ldots,$ and time x_M^* using $\vec{R}_m$. Since $\delta(\vec{\rho}) \leqslant 1$, we can fit all that within each frame of size 1, and during the remaining time $x_o^* = 1 - \delta(\vec{\rho}) \geqslant 0$ we can set the system to the degenerate service state $\vec{0}$. The fact that $\vec{\rho} \leqslant \sum_{m=1}^{M} x_m^* \vec{R}_m$ is a condition in the calculation of $\vec{x}^*(\vec{\rho}) = (x_1^*, x_1^*, \ldots, x_m^*, \ldots, x_M^*)$ implies that the time-averaged workload rate into each queue is less than the time-averaged service rate, so we expect each queue to be stable.

The previously mentioned TDM service policy provides an interesting intuitive picture of system stability, but has some important practical disadvantages, especially concerning the target applications of this work. *First*, explicit knowledge of the load vector $\vec{\rho}$ is required in order to compute the time-frame partitioning vector $\vec{x}^*(\vec{\rho}) = (x_1^*, x_1^*, \ldots, x_m^*, \ldots, x_M^*)$ and implement the TDM policy. However, $\vec{\rho}$ will typically not be given apriori. *Second*, the TDM policy is not adaptive, in the sense that it is not 'backlog responsive.' As a result, an unusually intense burst of jobs arriving in some queue will not push the system to devote any short-term extra service capacity to this

queue to alleviate congestion, even when all other queues are almost empty. In most practical systems we address, the backlog in each queue is observable, either as cumulative workload or as number of jobs in the queue. This information ought to be used by any efficient processing policy. *Third*, if there is a 'dead time' required to switch from one service combination to another, the stability region of the above TDM policy would shrink correspondingly to accommodate the service reconfiguration times. *Finally*, issues of scalability may arise, since in certain applications we may be dealing with hundreds to thousands of queues. The same practical problems would have other service policies that would require apriori knowledge of the load vector, would ignore backlog information, or would have non-negligible switching times. Such a policy, for example, would be the one choosing the service vector $\vec{R}_m$ independently at random in consecutive time frames and with probability equal to x_m^*. There are several other such policies.

Contrary to the previous backlog-agnostic policies, our focus in this paper is on adaptive, backlog-responsive ones that require no apriori knowledge of the load vector $\vec{\rho}$. Indeed, as long as $\vec{\rho}$ is in $\mathbf{S}$, the MaxProduct, FastEmpty, and BatchAdapt policies will keep the system stable without knowing the specific $\vec{\rho}$ value. Even if $\vec{\rho}$ changes (drifts in $\mathbf{S}$) over a time scale that is much longer than the one for load estimation/convergence in (1.3), these policies will automatically adapt and keep the system stable, being robust to long term changes of the load. Furthermore, by utilizing backlog information they are agile in adapting to short term bursts of incoming workload, tending to provide higher service capacity to stressed queues. Finally, the BatchAdapt policy will maximize the system throughput, even in the presence of substantial switching times between service vectors.

3. Throughput maximizing dynamic cone policies

In this section, we study a family of adaptive (backlog-responsive) service policies, which are based on partitioning the workload space into a set of cones, and selecting the service vector to use at each point in time depending on which cone the workload belongs to then. A *cone* $\mathbf{C} \subseteq \mathbb{R}_+^Q$ is a subset of the workload space, such that $\vec{w} \in \mathbf{C}$ implies $\phi\vec{w} \in \mathbf{C}$ for every $\phi \in \mathbb{R}_+$. A *cone covering* $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m, \ldots, \mathbf{C}_M\}$ of the workload space $\mathbb{R}_+^Q$ is a set of M cones such that $\bigcup_{m=1}^M \mathbf{C}_m = \mathbb{R}_+^Q$. A *cone policy*, based on the workload space cone covering $\{\mathbf{C}_m,\ m \in \mathbf{M}\}$, can use the service vector $\vec{R}_m$ whenever $\vec{W}(t) \in \mathbf{C}_m$. In general, some of the M cones may be empty (degenerate). Moreover, the intersection of several different cones may be non-empty, for example, several cones may have common boundary segments, or even overlap substantially. For workload vectors in such cone intersections, any one of the corresponding service vectors may be chosen arbitrarily. Figure 3 shows an example of a cone covering. Note that the cone corresponding to $\vec{R}_3$ is empty. We focus our attention on two types of cone policies. The first is characterized by an inner product maximization and is described and analyzed in section 3.1. The second is based on optimal workload depletion and is presented and studied in section 3.3.

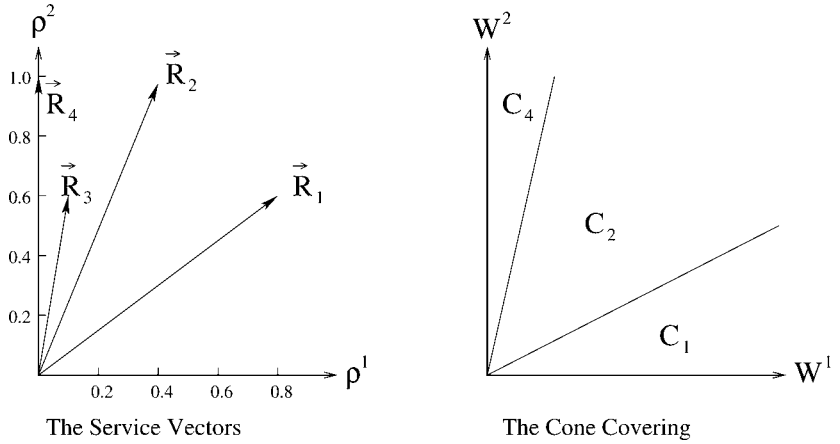

Figure 3. The MaxProduct cone covering/partitioning is shown on the right graph for the system described in figure 1. Its service vectors are shown on the left graph.

3.1. Cone policies based on maximal inner products (MaxProduct)

A family of throughput maximizing cone policies (not requiring knowledge of $\vec{\rho}$ to operate) can be constructed as follows. Given a weight vector $\vec{\alpha} \in \mathbb{R}_+^Q$ with positive components ($\alpha^q > 0$ for every $q \in \mathbf{Q}$) we define the (weighted) inner product

$$\langle \vec{u}, \vec{v} \rangle_{\vec{\alpha}} = \sum_{q=1}^{Q} \alpha^q u^q v^q = \langle \vec{u}, \vec{v} \rangle \tag{3.1}$$

of every two vectors $\vec{u}, \vec{v} \in \mathbb{R}^Q$. The inner product depends on $\vec{\alpha}$ intimately, but in what follows $\vec{\alpha}$ is arbitrarily fixed, so we drop the subscript $\vec{\alpha}$ in $\langle *, * \rangle_{\vec{\alpha}}$ for notational simplicity. Based on (3.1) we can define the workload space cone

$$\mathbf{C}_m = \left\{ \vec{w} \in \mathbb{R}_+^Q \colon \langle \vec{w}, \vec{R}_m \rangle = \max_{m' \in \mathbf{M}} \langle \vec{w}, \vec{R}_{m'} \rangle \right\}, \tag{3.2}$$

which is simply the set of workload vectors whose inner product with the service vector $\vec{R}_m$ is maximal. Under a slight 'stretch' of the standard vector projection concept, one might think of $\mathbf{C}_m$ as the set of workload vectors with *maximal projection* on $\vec{R}_m$ (skewed by the $\vec{\alpha}$-weights of the inner product and the service vector norms). This set is clearly a polyhedral cone. There are M cones $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m, \ldots, \mathbf{C}_M$ (one for each service vector), forming a cone covering of the workload space. Note that if a service vector $\vec{R}_{m'}$ is strictly dominated by some other $\vec{R}_m$ (that is, $R_{m'}^q < R_m^q$ for every queue $q \in \mathbf{Q}$) or a convex combination of other service vectors, the cone $\mathbf{C}_{m'}$ is simply $\{\vec{0}\}$, hence, degenerate. Moreover, note that several cones may share common boundary segments, partially 'touching' each other. For a workload vector on the common boundary

of several cones, there would be several service vectors having maximal inner product with it (all those generating the adjacent cones). In view of that, let us define

$$\mu(\vec{w}) = \left\{ m \in \mathbf{M}\colon \left\langle \vec{w}, \vec{R}_m \right\rangle = \max_{m' \in \mathbf{M}} \left\langle \vec{w}, \vec{R}_{m'} \right\rangle \right\} \tag{3.3}$$

to be the index set of service vectors of maximal inner product with $\vec{w}$. Note that in the *typical case* of $\vec{w}$ being in the non-empty interior of a cone $\mathbf{C}_m$ we have $\mu(\vec{w}) = \{m\}$ – the index of that single cone. However, in the special – yet important – case of $\vec{w}$ sitting on a common boundary of two or more adjacent cones, $\mu(\vec{w})$ is comprised of the indices of *all* those cones. Motivated by the latter case, let us finally define the workload-centric cone

$$\mathbf{C}(\vec{w}) = \bigcup_{m \in \mu(\vec{w})} \mathbf{C}_m \tag{3.4}$$

to be the union of all cones generated by service vectors that have maximal inner product with $\vec{w}$. Therefore, if $\vec{w}$ is in the non-empty interior of the cone $\mathbf{C}_m$, then $\mathbf{C}(\vec{w}) = \mathbf{C}_m$. However, when $\vec{w}$ is on the boundary of two or more adjacent cones, then $\mathbf{C}(\vec{w})$ is the union of all those cones 'touched' by $\vec{w}$, since the inner product of $\vec{w}$ with the service vectors generating all those cones is maximal. Note the equivalence

$$\mathbf{C}(\vec{w}) \subseteq \mathbf{C}(\vec{v}) \quad \Leftrightarrow \quad \mu(\vec{w}) \subseteq \mu(\vec{v}), \tag{3.5}$$

for every two vectors $\vec{w}, \vec{v} \in \mathbb{R}_+^Q$. Therefore, $\mathbf{C}(\vec{w}) \subseteq \mathbf{C}(\vec{v})$ implies that only service vectors in $\mu(\vec{v})$ might be used when the workload is $\vec{w}$. Moreover, it is easy to see that

$$\mathbf{C}(\vec{w}) \subseteq \mathbf{C}(\vec{v}) \quad \Rightarrow \quad \vec{w} \in \mathbf{C}(\vec{v}), \tag{3.6}$$

but the converse is *not* generally true; it holds only when $\vec{w}$ is in the *interior* of the cone $\mathbf{C}(\vec{v})$. Indeed, if $\vec{w}$ is on the boundary of $\mathbf{C}(\vec{v})$, then $\mathbf{C}(\vec{w})$ may not be a subset of $C(\vec{v})$. All these are *key cone properties* that will be leveraged extensively later.

Given the above setup, the *MaxProduct* policy $\pi_{\vec{\alpha}}$ chooses the service vector $\vec{R}_m$ at time t, if the workload $\overrightarrow{W}(t)$ belongs to the interior of the cone $\mathbf{C}_m$ (when non-empty). In the special case, where $\overrightarrow{W}(t)$ is on the boundary of two or more cones, the policy arbitrarily chooses any service vector with index in $\mu(\overrightarrow{W}(t))$.

A rich family of MaxProduct policies $\pi_{\vec{\alpha}}$ can be generated by varying the weight vector $\vec{\alpha}$. Individual queue weights α^q may reflect priority (or importance) and could be chosen to address various application-specific design and performance engineering considerations. Note that the higher the α^q the more service capacity $\pi_{\vec{\alpha}}$ will devote to queue q, tending to select vectors $\vec{R}_m$ that have significant rate R_m^q even for low workload $W^q(t)$. Due to receiving more service, queue q enjoys lower backlog and delay, when α^q gets higher while the other queue weights remain the same. This is an important property of the $\pi_{\vec{\alpha}}$ service policies for applications where differentiated quality of service (QoS) support is required. As shown below, when $\vec{\rho} \in \mathbf{S}$, any MaxProduct policy $\pi_{\vec{\alpha}}$ will maintain rate stability of the queueing system. Then, increasing the α-weight

of a queue results in lowering its backlog and delay, at the expense of performance in the other queues.

Next, let us develop some intuition about the nature of the $\pi_{\vec{\alpha}}$ cone policy. By its construction, it is easy to see that for any $\vec{\rho} \in \mathbf{S}$ and any workload vector $\vec{w}$ in the non-empty interior of a cone $\mathbf{C}_m$ we have $\langle \vec{w}, \vec{\rho} \rangle \leqslant \langle \vec{w}, \vec{R}_m \rangle$. Indeed, from (2.2) we see that $\vec{\rho} \in \mathbf{S}$ implies that $\vec{\rho}$ is dominated (componentwise) by some convex combination of the service vectors; that is, $\vec{\rho} \leqslant \sum_{m'=1}^{M} \beta_{m'} \vec{R}_{m'}$ for some $\beta_{m'} \geqslant 0$, $m' \in \mathbf{M}$, such that $\sum_{m'=1}^{M} \beta_{m'} = 1$. Taking the inner product with $\vec{w}$ and recalling that its maximal inner product is that with $\vec{R}_m$ (since $\vec{w} \in \mathbf{C}_m$) we get the inequality. Arguing analogously, we can see that in the general case, where $\vec{w}$ may be on cone boundaries, we have

$$\vec{\rho} \in \mathbf{S} \quad \Rightarrow \quad \left\langle \vec{w}, \vec{R}_m \right\rangle - \langle \vec{w}, \vec{\rho} \rangle = \gamma(\vec{w}) \geqslant 0, \quad \text{for any } m \in \mu(\vec{w}). \tag{3.7}$$

This is an important property, characteristic of the cones generated by MaxProduct policies, and is leveraged later.

Our *objective* is to show that when $\vec{\rho} \in \mathbf{S}$ we have $\lim_{t\to\infty}(\overrightarrow{W}(t)/t) = \vec{0}$, so the system is rate stable (due to lemma 2.3). Since $\langle *, * \rangle$ is an inner product, it is enough to show that $\lim_{t\to\infty} \langle \overrightarrow{W}(t)/t, \overrightarrow{W}(t)/t \rangle = 0$. Before showing this fact, we present a sequence of methodological steps which may be applicable to proving rate stability of other queueing systems. We also discuss some key intuition, which is based on the geometry of the MaxProduct cones.

- Arguing by contradiction, assume that $\limsup_{t\to\infty} \langle \overrightarrow{W}(t)/t, \overrightarrow{W}(t)/t \rangle > 0$ and let $\{t_a\}_{a=1}^{\infty}$ be an increasing unbounded time sequence on which the supremum limit is attained. By successive 'thinnings' of $\{t_a\}_{a=1}^{\infty}$ over the individual components of the workload vector, we can obtain an increasing unbounded subsequence $\{t_b\}_{b=1}^{\infty}$ of $\{t_a\}_{a=1}^{\infty}$ such that

$$\lim_{b\to\infty} \frac{\overrightarrow{W}(t_b)}{t_b} = \vec{\eta} \neq \vec{0}, \quad \text{and} \quad \langle \vec{\eta}, \vec{\eta} \rangle = \limsup_{t\to\infty} \left\langle \frac{\overrightarrow{W}(t)}{t}, \frac{\overrightarrow{W}(t)}{t} \right\rangle > 0, \tag{3.8}$$

  where $\vec{\eta} = (\eta^1, \eta^2, \ldots, \eta^q, \ldots, \eta^Q)$ is a nonzero vector with non-negative components (some – but not all – components may be zero and none is negative).
- To establish a contradiction, we need to construct some other increasing unbounded time sequence $\{s_d\}_{d=1}^{\infty}$ with $\lim_{d\to\infty}(\overrightarrow{W}(s_d)/s_d) = \vec{\psi}$ such that

$$\lim_{d\to\infty} \left\langle \frac{\overrightarrow{W}(s_d)}{s_d}, \frac{\overrightarrow{W}(s_d)}{s_d} \right\rangle = \left\langle \vec{\psi}, \vec{\psi} \right\rangle > \langle \vec{\eta}, \vec{\eta} \rangle, \tag{3.9}$$

  contradicting the fact that $\langle \vec{\eta}, \vec{\eta} \rangle$ is the limsup in (3.8). We pursue this strategy below.

The *method* for establishing the aforementioned contradiction is based on finding an increasing unbounded subsequence $\{t_c\}_{c=1}^{\infty}$ of $\{t_b\}_{b=1}^{\infty}$, as well as an associated increasing unbounded time sequence $\{s_c\}_{c=1}^{\infty}$ with the following key *properties* 1–3 below:

1. $\lim_{c\to\infty}((t_c - s_c)/t_c) = \varepsilon \in (0, 1)$ and $s_c < t_c$ for each c. That is, the length $t_c - s_c \approx \varepsilon t_c$ of the interval $(s_c, t_c]$ *grows linearly* in time t_c.

2. $\mathbf{C}(\overrightarrow{W}(z)) \subseteq \mathbf{C}(\vec{\eta})$ for all $z \in (s_c, t_c]$ and each c. Hence, from (3.6) we have $\overrightarrow{W}(z) \in \mathbf{C}(\vec{\eta})$, therefore, the *workload drifts within the cone* $\mathbf{C}(\vec{\eta})$ throughout the time interval $(s_c, t_c]$ and from (3.5) only vectors $\vec{R}_m$ with $m \in \mu(\vec{\eta})$ might be used by $\pi_{\vec{\alpha}}$ in this interval.

3. $W^q(z) > 0$ for all $z \in (s_c, t_c]$ and each c, when $\eta^q > 0$. That is, a queue with $\eta^q > 0$ *never empties* in the interval $(s_c, t_c]$.

The key *intuition* behind the existence of sequences with such properties is the following. Since the workload grows as $\overrightarrow{W}(t_c) \approx \vec{\eta} t_c$ at large times, its distance from the boundary of cone $\mathbf{C}(\vec{\eta})$ should also grow proportionally to t_c. Hence, if s_c is the last time before t_c that the workload enters this cone, returning from some excursion on or beyond its boundary, the time $t_c - s_c$ required to cover the distance to reach the 'ray' $\vec{\eta}$ should grow at least linearly with t_c. This is formalized in claim 3.1 later.

Let us now consider some important 'geometric' *implications* of the existence of a $\{s_c\}_{c=1}^{\infty}$ with such properties 1–3, which can lead to establishing the sought after contradiction:

- When $\eta^q > 0$, queue q never empties in $(s_c, t_c]$ (property 3), while only service vectors with $m \in \mu(\vec{\eta})$ may be used (property 2). Hence,

$$W^q(t_c) - W^q(s_c) = \Sigma^q(s_c, t_c) - \sum_{m\in\mu(\vec{\eta})} R_m^q T_m(s_c, t_c) \tag{3.10}$$

  where $T_m(s_c, t_c)$ is the amount of time in $(s_c, t_c]$ that the system spends using the service vector $\vec{R}_m$.

- Multiplying both sides of (3.10) by $\alpha^q \eta^q$ and summing up over all components $q \in \mathbf{Q}$, we get

$$\left\langle \overrightarrow{W}(t_c) - \overrightarrow{W}(s_c), \vec{\eta} \right\rangle = \left\langle \vec{\Sigma}(s_c, t_c), \vec{\eta} \right\rangle - \sum_{m\in\mu(\vec{\eta})} \left\langle \vec{R}_m, \vec{\eta} \right\rangle T_m(s_c, t_c). \tag{3.11}$$

  Components with $\eta^q = 0$ will suppress the behavior of the corresponding workload terms in the inner product sum, so only components with $\eta^q > 0$ play a role. Since $\langle \vec{R}_m, \vec{\eta} \rangle$ is the same for all $m \in \mu(\vec{\eta})$ – recall (3.3) – we have

$$\left\langle \overrightarrow{W}(t_c) - \overrightarrow{W}(s_c), \vec{\eta} \right\rangle = \left\langle \vec{\Sigma}(s_c, t_c), \vec{\eta} \right\rangle - \left\langle \vec{R}_m, \vec{\eta} \right\rangle (t_c - s_c) \tag{3.12}$$

  for any $m \in \mu(\vec{\eta})$. Note that $\sum_{m\in\mu(\vec{\eta})} T_m(s_c, t_c) = t_c - s_c$ because the cone policy is non-idling.

- Dividing (3.12) by $(t_c - s_c)$, letting $c \to \infty$, and using lemma 2.1.1(a), we get

$$\lim_{c\to\infty} \left\langle \frac{\overrightarrow{W}(t_c) - \overrightarrow{W}(s_c)}{t_c - s_c}, \vec{\eta} \right\rangle = \langle \vec{\rho}, \vec{\eta} \rangle - \left\langle \vec{R}_m, \vec{\eta} \right\rangle = -\gamma(\vec{\eta}) \leqslant 0, \tag{3.13}$$

where the inequality is due to (3.7). From (3.8) we have $\lim_{c\to\infty}(\overrightarrow{W}(t_c)/t_c) = \vec{\eta}$ (recall that $\{t_c\}_{c=1}^{\infty}$ is a thinning of $\{t_b\}_{b=1}^{\infty}$), so using property 1 we immediately get

$$\lim_{c\to\infty}\left\langle \frac{\overrightarrow{W}(s_c)}{s_c}, \vec{\eta}\right\rangle = \frac{\langle\vec{\eta},\vec{\eta}\rangle + \varepsilon\gamma(\vec{\eta})}{1-\varepsilon} > \langle\vec{\eta},\vec{\eta}\rangle, \tag{3.14}$$

after standard calculations. The inequality is due to the fact that $\varepsilon \in (0, 1)$.

- By successive 'thinnings' of $\{s_c\}_{c=1}^{\infty}$ over the individual components of the workload vector, we can finally obtain an increasing unbounded subsequence $\{s_d\}_{d=1}^{\infty}$ of $\{s_c\}_{c=1}^{\infty}$ such that

$$\lim_{d\to\infty}\frac{\overrightarrow{W}(s_d)}{s_d} = \vec{\psi}, \tag{3.15}$$

for some vector $\vec{\psi} \in \mathbb{R}_+^Q$. From (3.14) we then have $\langle\vec{\psi},\vec{\eta}\rangle > \langle\vec{\eta},\vec{\eta}\rangle$, which implies[4] that $\langle\vec{\psi},\vec{\psi}\rangle > \langle\vec{\eta},\vec{\eta}\,\rangle$. Therefore,

$$\lim_{d\to\infty}\left\langle \frac{\overrightarrow{W}(s_d)}{s_d}, \frac{\overrightarrow{W}(s_d)}{s_d}\right\rangle > \langle\vec{\eta},\vec{\eta}\,\rangle = \limsup_{t\to\infty}\left\langle \frac{\overrightarrow{W}(t)}{t}, \frac{\overrightarrow{W}(t)}{t}\right\rangle > 0, \tag{3.16}$$

which is clearly a *contradiction* to the choice of $\vec{\eta}$ in (3.8). Specifically, under these circumstances $\langle\vec{\eta},\vec{\eta}\rangle$ cannot attain the 'limit supremum' which was initially assumed.

The core result of this section is now given in proposition 3.1 below.

Proposition 3.1 (Rate stability under MaxProduct). For any arbitrarily fixed, strictly positive vector $\vec{\alpha} \in \mathbb{R}_+^Q$, let $\overrightarrow{W}(t)$ be the workload vector of the queueing system operating under the MaxProduct policy $\pi_{\vec{\alpha}}$ on any arbitrarily fixed traffic traces (1.1) satisfying (1.3). Then,

$$\vec{\rho} \in \mathbf{S} \quad\Rightarrow\quad \lim_{t\to\infty}\frac{\overrightarrow{W}(t)}{t} = \vec{0}, \tag{3.17}$$

which, by lemma 2.3(1), implies that the system is rate stable.

Proof. In light of the preceding discussion of the methodological approach to building the core contradiction, the proof of the proposition boils down to showing how to construct the sequence $\{s_c\}_{c=1}^{\infty}$ satisfying the previously mentioned properties 1–3.

The construction of $\{s_c\}_{c=1}^{\infty}$ will follow the important intermediate step stated in the following claim. As a matter of fact, this is the 'core' of the rate stability result, and reflects the key intuition discussed immediately after presenting properties 1–3 above.

Claim 3.1. Suppose that $\lim_{k\to\infty}(\overrightarrow{W}(t_k)/t_k) = \vec{\eta} \neq \vec{0}$ for some increasing unbounded time sequence $\{t_k\}_{k=1}^{\infty}$ and some nonzero vector $\vec{\eta} = (\eta^1, \eta^2, \ldots, \eta^q, \ldots, \eta^Q) \in \mathbb{R}_+^Q$

[4] Note that $\langle\vec{\eta},\vec{\eta}\rangle < \langle\vec{\psi},\vec{\eta}\rangle$ implies $\langle\vec{\eta},\vec{\eta}\rangle < \langle\vec{\psi},\vec{\psi}\rangle$, for any $\vec{\psi},\vec{\eta} \in \mathbb{R}^Q$. Indeed, this is a simple consequence of the fact that $\langle\vec{\psi}-\vec{\eta},\vec{\psi}-\vec{\eta}\rangle > 0$, since $\langle .,.\rangle$ is an inner product and $\vec{\psi} \neq \vec{\eta}$.

(some, but not all, components may be zero). Note that from the continuity of the inner product, we have $\mathbf{C}(\overrightarrow{W}(t_k)) \subseteq \mathbf{C}(\vec{\eta})$ for all large enough k. Define

$$s_k = \sup\{t < t_k\colon \mathbf{C}(\overrightarrow{W}(t)) \not\subseteq \mathbf{C}(\vec{\eta})\} \tag{3.18}$$

be the *last* time before t_k that $\mathbf{C}(\overrightarrow{W}(t))$ was *not* within $\mathbf{C}(\vec{\eta})$, that is, the last time that $\overrightarrow{W}(t)$ entered $\mathbf{C}(\vec{\eta})$ before t_k. By convention, set $s_k = 0$ if $\mathbf{C}(\overrightarrow{W}(t))$ has always been within $\mathbf{C}(\vec{\eta})$ up to t_k. Then,

$$\liminf_{k\to\infty} \frac{t_k - s_k}{t_k} = \varepsilon_1 > 0, \tag{3.19}$$

for some $\varepsilon_1 \in (0, 1]$. Therefore, $\overrightarrow{W}(t) \in \mathbf{C}(\vec{\eta})$ throughout the interval $(s_k, t_k]$, and the time $t_k - s_k \approx \varepsilon_1 t_k$ needed for the workload to reach its final state $\overrightarrow{W}(t_k) \approx \vec{\eta} t_k$ (after its most recent entrance into cone $\mathbf{C}(\vec{\eta})$) grows linearly in t_k.

Proof. Arguing by contradiction, suppose there is an increasing unbounded subsequence $\{t_n\}_{n=1}^{\infty}$ of $\{t_k\}_{k=1}^{\infty}$ with $\lim_{n\to\infty}((t_n - s_n)/t_n) = 0$. From lemma 2.1.2(b), we get $\lim_{n\to\infty}((\overrightarrow{W}(t_n) - \overrightarrow{W}(s_n))/s_n) = 0$, which together with $\lim_{n\to\infty}(\overrightarrow{W}(t_n)/t_n) = \vec{\eta}$, implies that $\lim_{n\to\infty}(\overrightarrow{W}(s_n)/s_n) = \vec{\eta}$. Using lemma 2.1.3, we then get $\lim_{n\to\infty}(\overrightarrow{W}(s_n^-)/s_n) = \vec{\eta}$, covering the possible case where a job arrival occurs at s_n and so $\overrightarrow{W}(s_n^-) \neq \overrightarrow{W}(s_n)$. This forces the cone of $\overrightarrow{W}(s_n^-)$ to be a subset of the cone $\mathbf{C}(\vec{\eta})$ for all large n, contradicting the fact that $\mathbf{C}(\overrightarrow{W}(s_n^-)) \not\subseteq \mathbf{C}(\vec{\eta})$ by the definition (3.18). □

Finally, the construction of the sequence $\{s_c\}_{c=1}^{\infty}$ satisfying properties 1–3 in order to apply the developed methodology is done as follows. Recall that $\{t_b\}_{b=1}^{\infty}$ in (3.8) is an increasing, unbounded sequence with $\lim_{b\to\infty}(\overrightarrow{W}(t_b)/t_b) = \vec{\eta}$ for a non-negative vector $\vec{\eta} \neq \vec{0}$. Using claim 3.1 and lemma 2.4 it can be easily seen that we can find an increasing unbounded subsequence $\{t_c\}_{c=1}^{\infty}$ of $\{t_b\}_{b=1}^{\infty}$ such that the following hold:

1. $\hat{s}_c = \sup\{t < t_c\colon \mathbf{C}(\overrightarrow{W}(t)) \not\subseteq \mathbf{C}(\vec{\eta})\}$ and $\lim_{c\to\infty}((t_c - \hat{s}_c)/t_c) = \varepsilon_1 \in (0, 1]$ (claim 3.1),
2. $s_c^q = \sup\{t < t_c\colon W^q(t) = 0\}$ and $\lim_{c\to\infty}((t_c - s_c^q)/t_c) = \varepsilon_2^q \in (0, 1]$, for $q \in \mathbf{Q}$ such that $\eta^q > 0$ (lemma 2.4).

Choose then $s_c = \max\{\hat{s}_c, \{s_c^q\colon \eta^q > 0,\ q \in \mathbf{Q}\},\ (1 - \varepsilon_3)t_c\}$ for some $\varepsilon_3 \in (0, 1)$, which clearly implies that:

(a) $\lim_{c\to\infty}((t_c - s_c)/t_c) = \varepsilon = \min\{\varepsilon_1, \{\varepsilon_2^q\colon \eta^q > 0\},\ \varepsilon_3\} \in (0, 1)$.

(b) $\mathbf{C}(\overrightarrow{W}(z)) \subseteq \mathbf{C}(\vec{\eta})$ for all $z \in (s_c, t_c]$ for all large c.

(c) For $q \in \mathbf{Q}$ with $\eta^q > 0$, $W^q(z) > 0$ for all $z \in (s_c, t_c]$, for all large c.

In other words, $\{s_c\}_{c=1}^{\infty}$ and $\{t_c\}_{c=1}^{\infty}$ satisfy properties 1–3, so the developed method can be applied to establish the required contradiction (3.16). This completes the proof of the proposition. □

Remark 3.1. What is the most general family of cone policies that could maintain system stability when $\vec{\rho} \in \mathbf{S}$? In particular, one might want to consider the cone policy generated by an inner product of the form $\vec{u}'A\vec{v}$, where A is a positive definite matrix. This is a natural extension of the inner product defined before, which corresponds to diagonal matrices A. Unfortunately, this general cone policy might not be throughput maximizing. Indeed, consider the case of two queues and two service vectors with:

$$\vec{R}_1 = (1, 0), \qquad \vec{R}_2 = (0, 1) \quad \text{and} \quad A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}. \tag{3.20}$$

In this case, for every workload vector $\vec{w} \neq \vec{0}$, we have $\vec{w}'A\vec{R}_2 > \vec{w}'A\vec{R}_1$. Therefore, $\vec{R}_1$ is never used and this policy does not maximize system throughput. However, one might conjecture that if A is such that it simply 'resizes' the cones generated by the inner product (3.1) without changing the 'neighbor relationships' of the cones, then the corresponding cone policy would maximize the throughput. We are currently exploring such issues.

Remark 3.2 (Adding probabilistic superstructure). The approach taken for the analysis of the system here is 'from first principles' and requires no probabilistic superstructure. The analysis framework is that of deterministic dynamical systems, driven by deterministic traffic traces/signals, and focuses on the asymptotics of its evolution trajectories. Assuming some additional 'light' probabilistic superstructure (a probabilistic framework allowing for having 'laws of large numbers' and taking stochastic limits), a corresponding analysis of cone policies is given in [1], utilizing fluid modeling methods [11] and applying them using the Lyapunov function $f(t) = \frac{1}{2}\langle \overrightarrow{W}(t), \overrightarrow{W}(t) \rangle$. Using a 'heavier' probabilistic superstructure (independence, renewal, exponentiallity, etc.) and the fluid approximation methodology [10], stronger forms of stability are further established (Harris recurrence), using the Lyapunov function $g(t) = \sqrt{\langle \overrightarrow{W}(t), \overrightarrow{W}(t) \rangle}$. For more details, please see [1]. The dynamical systems approach emphasized here for stability analysis of switched processing systems aims to explore their behavior in the most assumption-agnostic way and hopefully develop techniques for applying the TBM approach to other queueing structures.

3.2. Queue-length driven MaxProduct policies

The MaxProduct policy described so far is based on the maximal inner product of the *workload* vector with the service vectors. In many practical situations, however, the workload is not known prior to processing the jobs in the queue, but the queue lengths are directly observable. Hence, a policy based on the queue lengths (rather than the workloads) is needed in these situations. We develop below a modified version of the MaxProduct policy, which is driven by the queue lengths, and establish its rate stability.

In this section, we additionally assume that on each traffic trace (1.1) we can define a long term average value $\bar{\sigma}^q$ of the service requirement per job (on this trace), that is,

$$\lim_{k\to\infty} \frac{\sum_{j=1}^{k} \sigma_j^q}{k} = \bar{\sigma}^q > 0 \tag{3.21}$$

for each $q \in \mathbf{Q}$. Let $L^q(t) \in \mathbb{Z}_+$ be the number of jobs in queue q at time t, including the one currently under service, and $\vec{L}(t) = \{L^1(t),\ L^2(t), \ldots, L^q(t), \ldots, L^Q(t)\} \in \mathbb{Z}_+^Q$.

Recalling that each queue is FIFO, the key observation is that $W^q(t) \approx \bar{\sigma}^q L^q(t)$, when $W^q(t)$ and $L^q(t)$ are very large. As a matter of fact, under any service policy that follows the FIFO discipline in each queue, we have

$$\lim_{n\to\infty} \frac{\vec{W}(t_n)}{t_n} = \vec{\eta} \quad \Leftrightarrow \quad \lim_{n\to\infty} \frac{\vec{L}(t_n)}{t_n} = \vec{\eta}_{\bar{\sigma}} = \left(\frac{\eta^1}{\bar{\sigma}^1}, \frac{\eta^2}{\bar{\sigma}^2}, \ldots, \frac{\eta^q}{\bar{\sigma}^q}, \ldots, \frac{\eta^Q}{\bar{\sigma}^Q}\right), \tag{3.22}$$

for every increasing unbounded sequence $\{t_n\}_{n=1}^{\infty}$ of time points, on which the limit on either side exists. Indeed, let $J^q(t)$ be the index of the head job in queue q (the job under service) and observe that $J^q(t)+L^q(t)-1$ is the index of the tail job in that queue. Then, at every time t, we have $\sum_{j=J^q(t)+1}^{J^q(t)+L^q(t)-1} \sigma_j^q \leqslant W^q(t) \leqslant \sum_{j=J^q(t)}^{J^q(t)+L^q(t)-1} \sigma_j^q$, therefore

$$\frac{L^q(t)}{t} \frac{\sum_{j=J^q(t)+1}^{J^q(t)+L^q(t)-1} \sigma_j^q}{L^q(t)} \leqslant \frac{W^q(t)}{t} \leqslant \frac{L^q(t)}{t} \frac{\sum_{j=J^q(t)}^{J^q(t)+L^q(t)-1} \sigma_j^q}{L^q(t)}. \tag{3.23}$$

Using (3.21) we can easily see from (3.23) that $\lim_{n\to\infty}(\vec{L}(t_n)/t_n) = \vec{\eta}_{\bar{\sigma}}$ implies $\lim_{n\to\infty}(\vec{W}(t_n)/t_n) = \vec{\eta}$. Conversely, note that if $\lim_{n\to\infty}(\vec{W}(t_n)/t_n) = \vec{\eta}$, then $L^q(t) \to \infty$ when $\eta^q > 0$ (since jobs' size is sublinear; see lemma 2.2/1). Using (3.21) we get from the left side of (3.23) $\limsup_{n\to\infty}(L^q(t_n)/t_n)\bar{\sigma}^q \leqslant \eta^q$ and from its right side $\eta^q \leqslant \liminf_{n\to\infty}(L^q(t_n)/t_n)\bar{\sigma}^q$ which implies that $\lim_{n\to\infty}(L^q(t_n)/t_n) = \eta^q/\bar{\sigma}^q$ for each $q \in \mathbf{Q}$. This establishes the equivalence (3.22).

In view of the above, we can amend the workload-driven cone policy of the previous section to make it work with the queue lengths $\vec{L}$, as follows. First, for any $\vec{u} = \{u^1, u^2, \ldots, u^q, \ldots, u^Q\}$ and $\vec{v} = \{v^1, v^2, \ldots, v^q, \ldots, v^Q\}$ in $\mathbb{R}^Q$ define the modified α-weighted, $\bar{\sigma}$-skewed inner product

$$\langle \vec{u}, \vec{v} \rangle^{\bar{\sigma}} = \sum_{q=1}^{Q} \bar{\sigma}^q \alpha^q u^q v^q, \tag{3.24}$$

with fixed $\bar{\sigma}^q > 0$ and $\alpha^q > 0$ for each $q \in \mathbf{Q}$. Based on this, for each $m \in \mathbf{M}$ define now the (modified) cone

$$\mathbf{C}_m = \left\{\vec{l} \in \mathbb{Z}_+^Q \colon \left\langle \vec{l}, \vec{R}_m \right\rangle^{\bar{\sigma}} = \max_{m' \in \mathbf{M}} \left\langle \vec{l}, \vec{R}_{m'} \right\rangle^{\bar{\sigma}} \right\} \tag{3.25}$$

and note that $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m, \ldots, \mathbf{C}_M\}$ is a cone covering of the queue-length space $\mathbb{Z}_+^Q$.

The *queue-length driven MaxProduct* policy $\pi_{\vec{\alpha}/\vec{\sigma}}$ can be specified as follows: the service vector $\vec{R}_m$ may be used at time t only if $\vec{L}(t) \in \mathbf{C}_m$, that is, $\langle \vec{L}(t), \vec{R}_m \rangle^{\vec{\sigma}}$ is maximal. There may be more than one service vector that can be used when $\vec{L}(t) = \vec{l}$, for example, when $\vec{l}$ is on the boundary of two cones; specifically, any $\vec{R}_m$ with m in the index set

$$\mu(\vec{l}) = \left\{ m \in \mathbf{M} \colon \left\langle \vec{l}, \vec{R}_m \right\rangle^{\vec{\sigma}} = \max_{m' \in \mathbf{M}} \left\langle \vec{l}, \vec{R}_{m'} \right\rangle^{\vec{\sigma}} \right\} \tag{3.26}$$

can be used by the service policy when the backlog is $\vec{l}$. Similarly, the union of all cones associated with these service vectors is $\mathbf{C}(\vec{l}) = \bigcup_{m \in \mu(\vec{l})} \mathbf{C}_m$.

When $\vec{\rho} \in \mathbf{S}$, the queue-length driven MaxProduct policy $\pi_{\vec{\alpha}/\vec{\sigma}}$ keeps the system *rate-stable*. The proof of this fact parallels closely that for the workload driven MaxProduct studied in the previous section. This is due to the following two key facts:

1. A property equivalent to (3.7) holds also for the $\vec{\sigma}$-skewed inner product, namely, for any $\vec{l} \in \mathbb{R}_+^Q$ we have

$$\vec{\rho} \in \mathbf{S} \quad \Rightarrow \quad \left\langle \vec{l}, \vec{R}_m \right\rangle^{\vec{\sigma}} - \left\langle \vec{l}, \vec{\rho} \right\rangle^{\vec{\sigma}} \geqslant 0, \quad \text{for any } m \in \mu(\vec{l}). \tag{3.27}$$

   This is because, when $\vec{\rho}$ is in the stability region, it is dominated by a convex combination of the service vectors. It can be argued as (3.7), using the $\vec{\sigma}$-skewed inner product.

2. A claim equivalent to claim 3.1 still holds, with the modification that s_k is now defined as

$$s_k = \sup\left\{ t < t_k \colon \mathbf{C}(\vec{L}(t)) \nsubseteq \mathbf{C}(\vec{\eta}_{\vec{\sigma}}) \right\}. \tag{3.28}$$

   The proof of the modified claim follows by using (3.22).

Remark 3.3 (Implementation of the queue-length driven MaxProduct policy). Note that despite the use of $\vec{\sigma}$ in the definition of the queue-length driven MaxProduct policy, it is not necessary to know the actual value of this vector in order to implement this policy. Specifically, observe that any strictly positive vector can take the role of the vector: $(\vec{\sigma}^1 \alpha^1, \ldots, \vec{\sigma}^q \alpha^q, \ldots, \vec{\sigma}^Q \alpha^Q)$ in the definition of $\langle \vec{u}, \vec{v} \rangle^{\vec{\sigma}}$ in (3.24).

This completes the discussion of the modified MaxProduct that is driven by queue lengths. We next turn our attention to a fundamentally different class of cone service policies.

3.3. Cone policies based on optimal workload depletion (FastEmpty)

If a workload-driven MaxProduct policy is used in a 'static' context to drain out an initial workload $\vec{w}$, given that new arrivals are not admitted, it might not empty the system in the fastest possible time. Indeed, for certain weight $\vec{\alpha}$ and initial workload $\vec{w}$, the time to completely drain all queues could be larger than the minimum achievable by solving the corresponding linear program ($\mathrm{LP}_{\vec{w}}$) (defined in section 2 and below). For example, this

is the case for two queues and two service vectors $\vec{R}_1 = (0.5, 0.5)$, $\vec{R}_2 = (0.6, 0.45)$. When the initial workload is $\vec{w} = (1, 1)$, the minimal time to empty the system is 2 time units. However, the MaxProduct policy with $\vec{\alpha} = (1, 1)$ will empty the system in 2.1333 time units and is suboptimal[5] in that respect.

In this section, we describe the FastEmpty cone policies, which do not only maintain rate stability when $\vec{\rho} \in \mathbf{S}$, but are also *statically optimal* in the sense that they drain and empty the system in minimal time after new job admissions are blocked. The *key idea* behind the FastEmpty policies is that only service vectors that drain the workload optimally may be used at any time. Specifically, recall the linear program ($\mathrm{LP}_{\vec{w}}$) defined in section 2:

$$\begin{aligned} \text{minimize} \quad & \sum_{m=1}^{M} x_m \\ \text{subject to:} \quad & \mathcal{R}\vec{x} \geqslant \vec{w}, \quad \vec{x} \geqslant 0, \end{aligned} \tag{$\mathrm{LP}_{\vec{w}}$}$$

where $\mathcal{R}$ is the matrix with columns the vectors $\{\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_m, \ldots, \vec{R}_M\}$. For every $\vec{x}^*(\vec{w}) = (x_1^*, x_2^*, \ldots, x_m^*, \ldots, x_M^*)$ that is an optimal solution of ($\mathrm{LP}_{\vec{w}}$), the total time to empty all queues is

$$\delta(\vec{w}) = \sum_{m=1}^{M} x_m^*, \tag{3.29}$$

using each service vector $\vec{R}_m$ for time x_m^*.

The *FastEmpty* cone policy $\pi_\Delta \in \Pi$ is defined as follows. The parameter Δ is an arbitrarily *fixed* number in $(0, 1/M]$. The policy π_Δ can use at time t any $\vec{R}_m$ for which $x_m^*(\overrightarrow{W}(t)) \geqslant \Delta\delta(\overrightarrow{W}(t))$, where $\vec{x}^*(\overrightarrow{W}(t))$ is an optimal solution of ($\mathrm{LP}_{\vec{w}}$). If there is more than one such $\vec{R}_m$, any one of them may be chosen arbitrarily.

The role of Δ in the definition of the policy is simply to bound $x_m^*(\overrightarrow{W}(t))/\delta(\overrightarrow{W}(t))$ away from 0. The condition $\Delta \leqslant 1/M$ guarantees that the policy π_Δ is well defined. Indeed, recall that $\delta(\overrightarrow{W}(t)) = \sum_{m=1}^{M} x_m^*(\overrightarrow{W}(t))$, so there are M positive numbers (at maximum, since some summands may be zero) summing up to $\delta(\overrightarrow{W}(t))$. Therefore, at least one of the summands must be greater than $\delta(\overrightarrow{W}(t))/M$. This is why we choose $\Delta \leqslant 1/M$ – so that we can guarantee that there exists at least one positive summand greater than $\Delta\delta(\overrightarrow{W}(t))$ in the previous sum and hence the policy is well defined.

To express π_Δ directly in terms of cones, define the sets $\mathbf{C}_m$ for $m \in \mathbf{M}$, as follows (observe that they are actually cones):

$$\mathbf{C}_m = \left\{\vec{w} \in \mathbb{R}_+^Q\colon\ x_m^*(\vec{w}) \geqslant \Delta\delta(\vec{w}) \text{ for some optimal solution } \vec{x}^*(\vec{w}) \text{ of } \mathrm{LP}_{\vec{w}}\right\}. \tag{3.30}$$

[5] However, if $\vec{\alpha}$ is appropriately chosen so that $\vec{R}_m \in \mathbf{C}_m$ for every efficient service vector $\vec{R}_m$, then the resulting cone policy would drain the workload optimally and empty the system in minimal time. Unfortunately, such an $\vec{\alpha}$ does not always exist, for example, for three queues and three service vectors $\vec{R}_1 = (1, 1, 1)$, $\vec{R}_2 = (0.5, 2, 0.1)$, $\vec{R}_3 = (0.8, 1.9, 2)$.

Note that $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m, \ldots, \mathbf{C}_M\}$ is a cone covering of the workload space, hence, $\bigcup_{m=1}^{M} \mathbf{C}_m = \mathbb{R}_+^Q$. The policy π_Δ may use $\vec{R}_m$ when the workload is in cone $\mathbf{C}_m$. Actually, in contrast to the MaxProduct cone covering defined in section 3.1, the cones here may have non-negligible intersections, beyond the cone boundaries. This implies that typically more than one service vector may be used by π_Δ at any point in time. Motivated by this, define

$$\mu(\vec{w}) = \{m \in \mathbf{M}\colon\ \vec{w} \in \mathbf{C}_m\} \tag{3.31}$$

to be the index set of all the service vectors that may be used at time t, when $\overrightarrow{W}(t) = \vec{w}$. Equivalently,

$$\mu(\vec{w}) = \left\{m \in \mathbf{M}\colon\ x_m^*(\vec{w}) \geqslant \Delta\delta(\vec{w}) \text{ for some optimal solution } \vec{x}^*(\vec{w}) \text{ of } (\mathrm{LP}_{\vec{w}})\right\}. \tag{3.32}$$

Given this notation, the FastEmpty policy π_Δ is defined as arbitrarily choosing any vector $\vec{R}_m$ with $m \in \mu(\vec{w})$, when the workload is $\vec{w} = \overrightarrow{W}(t)$ at time t.

When $\vec{\rho} \in \mathbf{S}$, the FastEmpty policy π_Δ keeps the system *rate-stable*. The proof of this fact is given in appendix A.2. It is very similar in nature to the corresponding proof for the MaxProduct policy $\pi_{\vec{\alpha}}$ in section 3.1. Their main executional difference is that, in the FastEmpty proof, the minimal time $\delta(\overrightarrow{W}(t))$ to empty the system plays the role of the inner product $\langle \overrightarrow{W}(t), \overrightarrow{W}(t)\rangle$ in the MaxProduct proof. In particular, in order to show that under this policy $\lim_{t\to\infty}(\overrightarrow{W}(t)/t) = 0$, it is sufficient to show instead that $\lim_{t\to\infty} \delta(\overrightarrow{W}(t)/t) = 0$. The complete details are given in appendix A.2.

Remark 3.4 (The queue-length driven FastEmpty policy). Similarly to the queue-length driven MaxProduct policy, one could define a queue-length driven FastEmpty policy. This policy would be based on optimal solutions of the linear program $(\mathrm{LP}_{\vec{w}})$, with $\vec{w} = (\bar{\sigma}^1 L^1, \ldots, \bar{\sigma}^q L^q, \ldots, \bar{\sigma}^Q L^Q)$, where $\vec{L}$ is the vector of queue-lengths. This policy can be shown to also guarantee system rate stability. The details are omitted.

4. Adaptive batching policies and switching times (BatchAdapt)

In the cone policies studied in the previous section, it has been implicitly assumed that the time needed to switch from one service vector (mode, configuration) to another is negligible (theoretically, zero). However, in several important application areas of the model, switching between service modes may take non-negligible time (for example, in context switching in computer operating systems). Indeed, it may take significant time to reset and reconfigure system resources to a new service vector. During this switching time, the system may provide limited or even no service to the jobs. Therefore, policies involving frequent reconfigurations of the service mode may experience lower throughput, due to service idling during the reconfiguration periods.

In this section, we introduce a new family of service policies, called BatchAdapt, which achieve maximal throughput, even in the presence of substantial switching delays.

This is done by adaptively batching up jobs and processing them using each service configuration at most once per batch. The adaptive batching policies are implemented online and maintain the system rate-stable under the maximum possible load, despite the service capacity lost during switching intervals. Moreover, BatchAdapt policies allow for nonpreemptive (uninterrupted) job processing, which is required in several applications. BatchAdapt policies generalize those discussed in [4].

4.1. The batching scheme π_B and its rate-stability condition

We start by defining an adaptive batching scheme π_B that aggregates arriving jobs and processes them in batches. Jobs within each batch are processed according to a static schedule π^* (defined in the next section), which becomes asymptotically optimal as the number of jobs in the batch grows larger. The batches $\mathbf{B}_0,\ \mathbf{B}_1,\ \mathbf{B}_2, \ldots, \mathbf{B}_n, \ldots$ of π_B are inductively generated, as follows. Define S_n to be the time when $\mathbf{B}_n$ has fully formed and execution of its jobs commences, and T_n the time when its jobs have been fully processed.

1. *Batch* $\mathbf{B}_0$: The initial batch $\mathbf{B}_0$ is comprised of the jobs present in the system at time 0. It is empty if there are no such jobs. The jobs in $\mathbf{B}_0$ are scheduled according to π^* and processed in the time interval $(S_0, T_0]$, where $S_0 = 0$ and T_0 is the time when all jobs in $\mathbf{B}_0$ have departed (upon execution completion).

2. *Batch* $\mathbf{B}_1$: The jobs arriving (if any) during the interval $(S_0, T_0]$ comprise batch $\mathbf{B}_1$, which is fully formed at time $S_1 = T_0$ in this case. If no job arrives in $(S_0, T_0]$, then the first one to arrive after T_0, say at $t' > T_0$, comprises the batch $\mathbf{B}_1$ (single job one, or multiple jobs, in case more jobs arrive at the same time) and $S_1 = t'$. The jobs in $\mathbf{B}_1$ are scheduled and processed according to π^* and the last one to finish execution departs at T_1.

3. *Batch* $\mathbf{B}_{n+1}$: In general, given that the batch $\mathbf{B}_n$ was fully formed at time S_n and fully processed according to π^* by time T_n, the next job batch $\mathbf{B}_{n+1}$ is inductively structured as follows:

   (a) The jobs arriving while $\mathbf{B}_n$ is processed in $(S_n, T_n]$ comprise batch $\mathbf{B}_{n+1}$, which is fully formed at $S_{n+1} = T_n$.

   (b) If no job arrives in $(S_n, T_n]$, then the first one to arrive after T_n, say at $t'' > T_n$, comprises the single job batch $\mathbf{B}_{n+1}$ and $S_{n+1} = t''$.

   The batch $\mathbf{B}_{n+1}$ is scheduled according to π^* and completes processing at time T_{n+1}.

Note that $S_{n+1} = T_n$, if at least one job arrives while $\mathbf{B}_n$ is being processed. Observe also that jobs arriving while $\mathbf{B}_n$ is being processed in $(S_n, T_n]$ are blocked from receiving service until T_n and placed in the next batch $\mathbf{B}_{n+1}$, which starts processing at $S_{n+1} = T_n$. Figure 4 illustrates the workload dynamics and batch formation under the BatchAdapt policy.

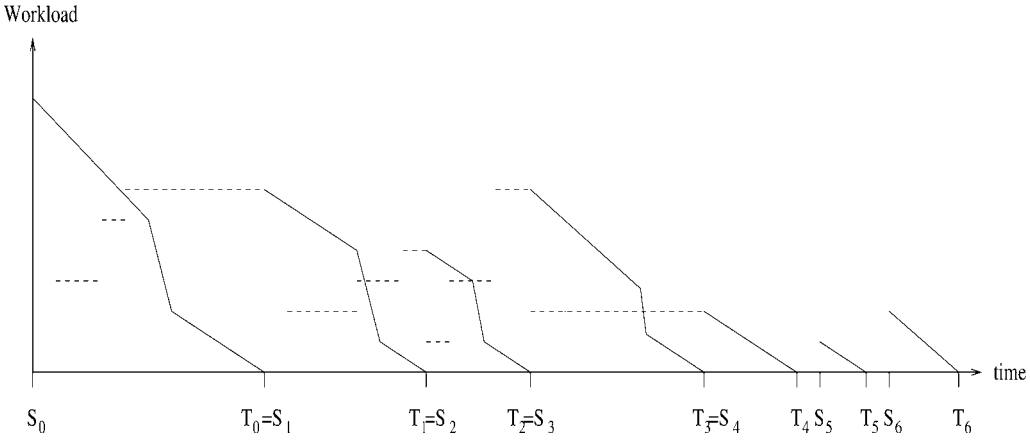

Figure 4. The BatchAdapt policy is 'schematically' illustrated in this figure. The solid line represents the decreasing workload of the batch that is currently being processed, while the dashed line represents the increasing workload of the next batch which is currently being formed and will be processed subsequently.

We now focus on the scheduling scheme π^* used to schedule the jobs in $\mathbf{B}_n$ for $n \in \{1, 2, \ldots\}$. First, note that the structure of $\mathbf{B}_n$ – and its formation time S_n and service completion time T_n – all depend implicitly on π^*. Actually, we should have written $\mathbf{B}_n(\pi^*)$, $S_n(\pi^*)$, $T_n(\pi^*)$, but we chose to drop the π^* argument for notational simplicity. Therefore, a complete description of the service policy is given by the pair π_B/π^* specifying both the above *batching scheme* π_B and the job *scheduling scheme* π^* within each batch.

The following proposition shows that in order for π_B/π^* to maintain rate stability of the system, it suffices that the processing times $T_n - S_n$ of $\mathbf{B}_n$ grow at most 'sublinearly' in T_n, as $T_n \to \infty$. Note that $T_n \to \infty$, as $n \to \infty$.

Proposition 4.1 (Sublinear batch growth implies rate stability). Let $\overrightarrow{W}(t)$ be the workload vector at time t, given that the system operates under the policy π_B/π^* and π^* is some fixed job scheduling scheme within each batch. Then,

$$\lim_{n\to\infty} \frac{T_n - S_n}{T_n} = 0 \quad \Rightarrow \quad \lim_{t\to\infty} \frac{\overrightarrow{W}(t)}{t} = 0. \tag{4.1}$$

Therefore, if the batch processing times grow 'sublinearly' the system is rate stable (by lemma 2.3).

Proof. Suppose that $\lim_{n\to\infty}((T_n - S_n)/T_n) = 0$ and note that from lemma 2.1.2(a) we have

$$\lim_{n\to\infty} \frac{\overrightarrow{W}(T_n)}{T_n} = \lim_{n\to\infty} \frac{\vec{\Sigma}(S_n, T_n)}{T_n} = 0, \tag{4.2}$$

since $\overrightarrow{W}(T_n) = \vec{\Sigma}(S_n, T_n)$ due to the structure of π_B. For any time point $t > 0$, define $n(t)$ so that $T_{n(t)-1} < t \leqslant T_{n(t)}$; it is well defined since $T_n \to \infty$, as $n \to \infty$, and, by convention, $T_{-1} = 0$. Note that every job present in the system at time t should also be in it, either at time $S_{n(t)}$ when the batch $\mathbf{B}_{n(t)}$ was formed and its processing started, or at time $T_{n(t)} = S_{n(t)+1}$ when the next batch $\mathbf{B}_{n(t)+1}$ starts its processing. Therefore, $\overrightarrow{W}(t) \leqslant \overrightarrow{W}(T_{n(t)}) + \overrightarrow{W}(S_{n(t)})$, where the vector inequality is valid componentwise, so

$$\begin{aligned} 0 &\leqslant \lim_{t\to\infty} \frac{\overrightarrow{W}(t)}{t} \leqslant \lim_{t\to\infty} \left[\frac{\overrightarrow{W}(T_{n(t)})}{T_{n(t)}} \frac{T_{n(t)}}{t} + \frac{\overrightarrow{W}(S_{n(t)})}{S_{n(t)}} \frac{S_{n(t)}}{t} \right] \\ &\leqslant 0 \times \lim_{t\to\infty} \frac{T_{n(t)}}{S_{n(t)}} \frac{S_{n(t)}}{T_{n(t)-1}} + \lim_{t\to\infty} \frac{\overrightarrow{W}(S_{n(t)})}{S_{n(t)}} \frac{S_{n(t)}}{T_{n(t)-1}} = 0. \end{aligned} \tag{4.3}$$

The second inequality follows from (4.2) and from the definition of $n(t)$. The final equality follows from the assumption that $\lim_{n\to\infty}((T_n - S_n)/T_n) = 0$ and by observing that $S_{n(t)} = T_{n(t)-1}$ if at least one job arrives in $(S_{n(t)-1}, T_{n(t)-1}]$; if not, the workload $\overrightarrow{W}(S_{n(t)})$ is comprised of a single job and we can use lemma 2.2. □

4.2. The robust asympotically optimal schedule $\pi^*_G(\vec{\rho})$

We now construct a specific scheduling scheme π^*_G which takes the role of π^* in 4.1 and leads to sublinear batch growth and, hence, rate stability of the system. It processes the jobs of each batch in *groups*. Recalling that jobs arriving in $(S_{n-1}, S_n]$ form the batch $\mathbf{B}_n$, consider the jobs arriving in some general time interval $(s, s']$. The schedule π^*_G structures job groups as follows:

- Given the load vector $\vec{\rho}$, let $\vec{x}^*(\vec{\rho})$ be an *optimal solution* of $(\mathrm{LP}_{\vec{\rho}})$. Therefore,

$$\sum_{a=1}^{M} x^*_a(\vec{\rho}) R^q_a \geqslant \rho^q, \tag{4.4}$$

  for every $q \in \mathbf{Q}$ and $\delta(\vec{\rho}) = \sum_{a=1}^{M} x^*_a(\vec{\rho})$ is minimal.
- Define the group *allocation parameters* $\Phi^q_m(\vec{\rho})$ for $m \in \mathbf{M}$ and $q \in \mathbf{Q}$ by

$$\Phi^q_m(\vec{\rho}) = \frac{\sum_{a=1}^{m} x^*_a(\vec{\rho}) R^q_a}{\rho^q}, \tag{4.5}$$

  recalling that it is assumed that $\rho^q > 0$ for all queues. By convention, we define $\Phi^q_0(\vec{\rho}) = 0$. Note that $\Phi^q_0(\vec{\rho}) \leqslant \Phi^q_1(\vec{\rho}) \leqslant \Phi^q_2(\vec{\rho}) \leqslant \cdots \leqslant \Phi^q_{m-1}(\vec{\rho}) \leqslant \Phi^q_m(\vec{\rho}) \leqslant \cdots \leqslant \Phi^q_M(\vec{\rho})$, while $\Phi^q_M(\vec{\rho}) \geqslant 1$ because of (4.4).
- For each $q \in \mathbf{Q}$ and $m \in \mathbf{M}$, define now the following intervals (disjoint for different m, given fixed q)

$$\mathbf{D}^q_m(s, s'; \vec{\rho}) = \left(s + (s' - s)\Phi^q_{m-1}(\vec{\rho}), s + (s' - s)\Phi^q_m(\vec{\rho})\right] \cap (s, s'] \tag{4.6}$$

and observe that (since $\Phi_M^q(\vec{\rho}) \geqslant 1$) the intervals $\mathbf{D}_m^q(s, s'; \vec{\rho})$, $m \in \mathbf{M}$ form a disjoint partition of $(s, s']$.

- The jobs arriving in $(s, s']$ are now structured into *job groups*

$$\mathbf{G}_m^q(s, s'; \vec{\rho}) = \left\{(q, j) \in \mathbf{Q} \times \mathbb{Z}_+ \colon\ t_j^q \in \mathbf{D}_m^q(s, s'; \vec{\rho})\right\}, \tag{4.7}$$

according to their arrival times t_j^q – where (q, j) denotes the jth job arriving to queue q. That is, $\mathbf{G}_m^q(s, s'; \vec{\rho})$ is comprised of the jobs arriving to queue q in the time interval $\mathbf{D}_m^q(s, s'; \vec{\rho})$. For any fixed $q \in \mathbf{Q}$, the groups $\mathbf{G}_m^q(s, s'; \vec{\rho})$, $m \in \mathbf{M}$ form a disjoint partition of the set of jobs arriving in $(s, s']$ at queue q.

The jobs in the previously defined groups are processed by π_G^* in a *nonpreemptive* manner, that is, the processing of a job cannot be interrupted while in progress, nor can the rate at which it is processed be changed. This is required in several application areas. The analysis, however, obviously carries over to the more relaxed case of preemptive/resume job processing. The schedule π_G^* cycles between the service combinations $\vec{R}_m$, $m \in \mathbf{M}$ with $x_m^*(\vec{\rho}) > 0$ (in any given order) and processes the jobs as follows:

1. For $m \in \mathbf{M}$ such that $x_m^*(\vec{\rho}) > 0$, the schedule π_G^* switches the system into service configuration $\vec{R}_m$ and initiates parallel processing of the groups $\mathbf{G}_m^q(s, s'; \vec{\rho})$, $q \in \mathbf{Q}$ – one in each queue. The jobs in group $\mathbf{G}_m^q(s, s'; \vec{\rho})$ in queue q are processed in a first-come-first-served (or other priority discipline) nonpreemptive manner at service rate R_m^q. Note that if $R_m^q = 0$ the group $\mathbf{G}_m^q(s, s'; \vec{\rho})$ is empty.

2. When $R_m^q > 0$, the time to process the jobs in group $\mathbf{G}_m^q(s, s'; \vec{\rho})$ is simply the sum of the service requirements of the jobs in this group divided by the service rate R_m^q. Moreover, since for a fixed m, the groups $\mathbf{G}_m^q(s, s'; \vec{\rho})$, $q \in \mathbf{Q}$ are processed in parallel, the service configuration $\vec{R}_m$ is used for as long as there are jobs present in any of these groups. Hence, if $\tau_m(s, s'; \vec{\rho})$ denotes the time that the vector $\vec{R}_m$ is used to process the jobs in groups $\mathbf{G}_m^q(s, s'; \vec{\rho})$, $q \in \mathbf{Q}$, while $\tau_m^q(s, s'; \vec{\rho})$ is the processing time of all jobs in $\mathbf{G}_m^q(s, s'; \vec{\rho})$, we have

$$\tau_m(s, s'; \vec{\rho}) = \max_{q \in \mathbf{Q}} \left\{\tau_m^q(s, s'; \vec{\rho})\right\} = \max_{q \in \mathbf{Q},\, R_m^q > 0} \left\{\frac{1}{R_m^q} \times \sum_{(q,j) \in \mathbf{G}_m^q(s, s'; \vec{\rho})} \sigma_j^q \right\}. \tag{4.8}$$

3. When *all* these groups $\mathbf{G}_m^q(s, s'; \vec{\rho})$, $q \in \mathbf{Q}$ have completed processing the system switches to another service configuration $\vec{R}_{m'}$ with $x_{m'}^*(\vec{\rho}) > 0$ and starts processing the groups $\mathbf{G}_{m'}^q(s, s'; \vec{\rho})$, $q \in \mathbf{Q}$ in parallel.

Switching the system from $\vec{R}_m$ to $\vec{R}_{m'}$ may require some transition time. Note, however, that in order to process all the jobs arriving in $(s, s']$, at most M switchings are required. Hence, if *switching times* are bounded, the total switching time when processing the jobs in any particular batch is bounded. Actually, if a basic solution $\vec{x}^*(\vec{\rho})$ of $(\mathrm{LP}_{\vec{\rho}})$ is selected, then the number of nonzero elements $\{x_m^*(\vec{\rho}) > 0;\ m \in \mathbf{M}\}$ is at most Q, and such is also the number of switchings between service configurations.

The above completes the specification of the schedule π_G^*. Note that since its allocation parameters and job groups depend on $\vec{\rho}$, we should actually write $\pi_G^*(\vec{\rho})$. The following proposition reveals the two *key properties* of π_G^*, which are that (1) it is asymptotically optimal as the interval $(s, s']$ increases in size and larger job populations are scheduled, and (2) that it is robust with respect to $\vec{\rho}$.

Proposition 4.2 (Robust asymptotic optimality of π_G^*). Consider any sequence of load vectors $\{\vec{\rho}_k\}_{k=0}^{\infty}$, such that:

$$\lim_{k\to\infty} \vec{\rho}_k = \vec{\rho} \in \mathbf{S}, \tag{4.9}$$

with $\rho_k^q > 0$ for all q and k. Consider also any two increasing unbounded time sequences $\{s_k\}_{k=0}^{\infty}$ and $\{s'_k\}_{k=0}^{\infty}$, such that $s_k < s'_k$ and

$$\lim_{k\to\infty} \frac{s'_k - s_k}{s'_k} = \theta > 0. \tag{4.10}$$

Suppose that jobs arriving in $(s_k, s'_k]$ are batched up together and processed according to the schedule $\pi_G^*(\vec{\rho}_k)$. Then, the total time $\tau(s_k, s'_k; \vec{\rho}_k)$ to process these jobs under $\pi_G^*(\vec{\rho}_k)$ satisfies:

$$\lim_{k\to\infty} \frac{\tau(s_k, s'_k; \vec{\rho}_k)}{s'_k - s_k} = \delta(\vec{\rho}), \tag{4.11}$$

where

$$\tau(s_k, s'_k; \vec{\rho}_k) = \sum_{m=1}^{M} \tau_m(s_k, s'_k; \vec{\rho}_k), \tag{4.12}$$

and $\tau_m(s_k, s'_k; \vec{\rho}_k)$, $m \in \mathbf{M}$, is the time that $\pi_G^*(\vec{\rho}_k)$ spends processing jobs under $\vec{R}_m$ – as defined in (4.8) – and $\delta(\vec{\rho})$ is the minimal emptying time that $(\mathrm{LP}_{\vec{\rho}})$ yields.

Proof. First note that from (4.10) and lemma 2.1.1(a) we have $\lim_{k\to\infty}(\vec{\Sigma}(s_k, s'_k)/(s'_k - s_k)) = \vec{\rho}$. Consider the vector $\vec{\tau}(s_k, s'_k; \vec{\rho}_k) = \{\tau_m(s_k, s'_k; \vec{\rho}_k),\ m \in \mathbf{M}\}$, whose elements are the group processing times, as defined in (4.8). To show (4.11) it suffices to show that the following two conditions hold:

1. $\{\vec{\tau}(s_k, s'_k; \vec{\rho}_k)/(s'_k - s_k)\}_{k=1}^{\infty}$ is a bounded sequence of vectors in $\mathbb{R}_+^M$, and

2. For every increasing unbounded subsequences $\{s_l\}_{l=1}^{\infty}$ and $\{s'_l\}_{l=1}^{\infty}$ of $\{s_k\}_{k=1}^{\infty}$ and $\{s'_k\}_{k=1}^{\infty}$, respectively, such that $\lim_{l\to\infty}(\vec{\tau}(s_l, s'_l; \vec{\rho}_l)/(s'_l - s_l))$ exists, we have

$$\sum_{m=1}^{M} \lim_{l\to\infty} \frac{\tau_m(s_l, s'_l; \vec{\rho}_l)}{s'_l - s_l} = \delta(\vec{\rho}).$$

To show condition 1, we first define $R_* = \min_{q\in\mathbf{Q}} \min_{m\in\mathbf{M}}\{R_m^q\colon\ R_m^q > 0\}$. Notice that since π_G^* is non-idling (except for when switchings occur) we have for all k,

$$\sum_{m=1}^{M} \frac{\tau_m(s_k, s_k'; \vec{\rho}_k)}{s_k' - s_k} \leqslant \frac{\sum_{q=1}^{Q} \Sigma^q(s_k, s_k')}{R_*(s_k' - s_k)} \to \frac{1}{R_*} \sum_{q=1}^{Q} \rho^q, \tag{4.13}$$

where the limit on the right-hand side is taken for $k\to\infty$ and is due to lemma 2.1.1(a). The inequality follows from the worst case scenario in which each queue is processed separately using the slowest service rate possible.

To show condition 2, suppose that $\{s_l\}_{l=1}^{\infty}$ and $\{s_l'\}_{l=1}^{\infty}$ are increasing unbounded subsequences of $\{s_k\}_{k=1}^{\infty}$ and $\{s_k'\}_{k=1}^{\infty}$, respectively, such that $\lim_{l\to\infty}(\vec{\tau}(s_l, s_l'; \vec{\rho}_l)/(s_l' - s_l)) = \vec{x}$, for some vector $\vec{x} \in \mathbb{R}_+^M$. We establish condition 2 by showing that $\vec{x}$ is an optimal solution of $(\mathrm{LP}_{\vec{\rho}})$. For $q \in \mathbf{Q}$ and $m \in \mathbf{M}$, let $\tau_m^q(s_l, s_l'; \vec{\rho}_l)$ be the time that $\vec{R}_m$ is used to process jobs in queue q, as defined in (4.8); hence, for all l, $\tau_m(s_l, s_l'; \vec{\rho}_l) = \max_{q\in\mathbf{Q}}\{\tau_m^q(s_l, s_l'; \vec{\rho}_l)\}$. Let $\{s_i\}_{i=1}^{\infty}$ and $\{s_i'\}_{i=1}^{\infty}$ be increasing unbounded subsequences of $\{s_l\}_{l=1}^{\infty}$ and $\{s_l'\}_{l=1}^{\infty}$, respectively, such that $\lim_{i\to\infty}(\tau_m^q(s_i, s_i'; \vec{\rho}_i)/(s_i' - s_i))$ exists for all $q \in \mathbf{Q}$ and all $m \in \mathbf{M}$, and is equal to some $y_m^q \geqslant 0$. In addition, $\{s_i\}_{i=1}^{\infty}$ and $\{s_i'\}_{i=1}^{\infty}$ are chosen so that for each $m \in \mathbf{M}$ we have, either

(a) $x_m^*(\vec{\rho}_i) > 0$ for all large i, or

(b) $x_m^*(\vec{\rho}_i) = 0$ for all large i,

where for all i, $\vec{x}^*(\vec{\rho}_i)$ is the optimal solution of $(\mathrm{LP}_{\vec{\rho}_i})$ that is used to construct the groups associated with the policy π_G^*. Fix $m \in \mathbf{M}$ such that $x_m^*(\vec{\rho}_i) > 0$ for all i. For all $q \in \mathbf{Q}$ such that $R_m^q > 0$, we have

$$\begin{aligned}
y_m^q &= \lim_{i\to\infty} \frac{\tau_m^q(s_i, s_i'; \vec{\rho}_i)}{s_i' - s_i} \\
&= \lim_{i\to\infty} \frac{\Sigma^q\big(s_i + (s_i' - s_i)\big(\sum_{a=1}^{m-1} x_a^*(\vec{\rho}_i) R_a^q/\vec{\rho}_i^q\big),\, s_i + (s_i' - s_i)\big(\sum_{a=1}^{m} x_a^*(\vec{\rho}_i) R_a^q/\vec{\rho}_i^q\big)\big)}{(s_i' - s_i) R_m^q} \\
&= \lim_{i\to\infty} \frac{\Sigma^q\big(s_i + (s_i' - s_i)\big(\sum_{a=1}^{m-1} x_a^*(\vec{\rho}_i) R_a^q/\vec{\rho}_i^q\big),\, s_i + (s_i' - s_i)\big(\sum_{a=1}^{m} x_a^*(\vec{\rho}_i) R_a^q/\vec{\rho}_i^q\big)\big)}{(s_i' - s_i)((x_m^*(\vec{\rho}_i) R_m^q)/\vec{\rho}_i^q)} \\
&\quad \times \frac{x_m^*(\vec{\rho}_i)}{\vec{\rho}_i^q} \\
&= \rho^q \frac{\lim_{i\to\infty} x_m^*(\vec{\rho}_i)}{\rho^q} = \lim_{i\to\infty} x_m^*(\vec{\rho}_i),
\end{aligned} \tag{4.14}$$

where the first limit is obtained from lemma 2.1.1(a) using (4.10). On the other hand, if $q \in \mathbf{Q}$ is such that $R_m^q = 0$, or $m \in \mathbf{M}$ is such that $x_m^*(\vec{\rho}_i) = 0$ for all i, then the set $\mathbf{G}_m^q(s_i, s_i'; \vec{\rho}_i)$ is empty for all i, and so $y_m^q = \tau_m^q(s_i, s_i'; \vec{\rho}_i) = 0$. Putting it all together, for all $m \in \mathbf{M}$, we have

$$x_m = \lim_{i\to\infty} \frac{\tau_m(s_i, s_i'; \vec{\rho}_i)}{s_i' - s_i} = \max_{q\in\mathbf{Q}} \lim_{i\to\infty} \frac{\tau_m^q(s_i, s_i'; \vec{\rho}_i)}{s_i' - s_i} = \max_{q\in\mathbf{Q}} y_m^q = \lim_{i\to\infty} x_m^*(\vec{\rho}_i). \tag{4.15}$$

In particular, in vector form we have,

$$\vec{x} = \lim_{l\to\infty} \frac{\vec{\tau}(s_l, s'_l; \vec{\rho}_l)}{s'_l - s_l} = \lim_{i\to\infty} \vec{x}^*(\vec{\rho}_i),$$

where $\vec{x}^*(\vec{\rho}_i)$ is an optimal solution of $(\mathrm{LP}_{\vec{\rho}_i})$ for all i. From the continuity of the optimal solution of a linear program as a function of its right-hand side vector (see [7]) we conclude that $\vec{x}$ is an optimal solution of $(\mathrm{LP}_{\vec{\rho}})$. In particular, $\sum_{m=1}^{M} \lim_{l\to\infty}(\tau_m(s_l, s'_l; \vec{\rho}_l)/(s'_l - s_l)) = \sum_{m=1}^{M} x_m = \delta(\vec{\rho})$. This completes the proof of the proposition. □

Note that the switching times between service vectors are not included in (4.11). However, even if we include them, the result (4.11) will obviously not change. The reason is that there are at most M such switchings in the processing of any batch, so the total switching time is fixed (bounded) and 'washes out' in the limit.

4.3. Rate stability of π_B/π_G^*

Consider now the system operating under $\pi_B/\pi_G^*(\vec{\rho}_n)$, where $\vec{\rho}_n$ is an estimate of the load vector $\vec{\rho} \in \mathbf{S}$. For example, given that S_n and T_n are the service start and end times of batch $\mathbf{B}_n$ specified in section 4.1, a practical estimate would be

$$\vec{\rho}_n = \frac{\vec{\Sigma}(0, S_n)}{S_n}, \tag{4.16}$$

which would converge to $\lim_{n\to\infty} \vec{\rho}_n = \vec{\rho} \in \mathbf{S}$ because of (1.3). Therefore, $\vec{\rho}_n$ could be the estimated time-averaged load over the interval $(0, S_n]$ and would be used to schedule the batch $\mathbf{B}_n$ – comprised of the jobs that arrived in $(S_{n-1}, S_n]$ according to the schedule $\pi_G^*(\vec{\rho}_n)$.

The following proposition shows that $\pi_B/\pi_G^*(\vec{\rho}_n)$ is rate stable, for any $\lim_{n\to\infty} \vec{\rho}_n = \vec{\rho} \in \mathbf{S}$. The intuition is the following. At large times (when $\vec{\rho}_n \approx \vec{\rho}$), if a batch is small, π_G^* might not schedule it well to execute in the shortest possible time; however, as the batches grow larger due to congestion, the asymptotic optimality property of π_G^* kicks in and schedules the batches extremely efficiently, bringing congestion under control and keeping the system rate stable.

Proposition 4.3 (Rate stability of π_B/π_G^*). Suppose that the jobs in batch $\mathbf{B}_n$ of π_B are scheduled according to $\pi_G^*(\vec{\rho}_n)$, using some load estimation vector $\vec{\rho}_n$ such that $\lim_{n\to\infty} \vec{\rho}_n = \vec{\rho}$, with $\rho_n^q > 0$ for all n and q. Then,

$$\vec{\rho} \in \mathbf{S} \quad \Rightarrow \quad \lim_{n\to\infty} \frac{T_n - S_n}{T_n} = 0, \tag{4.17}$$

hence, by proposition 4.1, the system is rate stable under π_B/π_G^*.

Proof. Arguing by contradiction, suppose that $\limsup_{n\to\infty}((T_n - S_n)/T_n) = \varepsilon > 0$. Let $\{S_i\}_{i=1}^{\infty}$ and $\{T_i\}_{i=1}^{\infty}$ be two increasing unbounded subsequences of $\{S_n\}_{n=1}^{\infty}$ and $\{T_n\}_{n=1}^{\infty}$,

respectively, such that $\lim_{i\to\infty}((T_i - S_i)/T_i) = \varepsilon$. Let $\phi = \limsup_{i\to\infty}((T_{i-1} - S_{i-1})/T_{i-1})$, and let $\{S_k\}_{k=1}^{\infty}$ and $\{T_k\}_{k=1}^{\infty}$ be two increasing unbounded subsequences of $\{S_i\}_{i=1}^{\infty}$ and $\{T_i\}_{i=1}^{\infty}$, respectively, such that $\lim_{k\to\infty}((T_{k-1} - S_{k-1})/T_{k-1}) = \phi$. Clearly, $0 \leqslant \phi \leqslant \varepsilon$. We consider two cases:

1. $\phi > 0$. In this case, for all k large enough, we have $S_k = T_{k-1}$ because there are always some jobs arriving during the time interval $(S_{k-1}, T_{k-1}]$ (see lemma 2.1.1(b)). In particular, according to $\pi_B/\pi_G^*(\vec{\rho}_k)$, we have $\overrightarrow{W}(T_{k-1}) = \overrightarrow{W}(S_k) = \vec{\Sigma}(S_{k-1}, T_{k-1})$ and $T_k - S_k = \tau(T_{k-1}, S_{k-1}; \vec{\rho}_k) = \tau(S_k, S_{k-1}; \vec{\rho}_k)$. Consequently, using proposition 4.2, we get

$$\begin{aligned}\varepsilon &= \lim_{k\to\infty} \frac{T_k - S_k}{T_k} = \lim_{k\to\infty} \frac{T_k - S_k}{T_{k-1} - S_{k-1}} \frac{T_{k-1} - S_{k-1}}{T_{k-1}} \frac{T_{k-1}}{T_k} \\ &= \delta(\vec{\rho})\phi(1 - \varepsilon) < \phi \leqslant \varepsilon, \end{aligned} \tag{4.18}$$

   since $\delta(\vec{\rho}) \leqslant 1$ for $\vec{\rho} \in \mathbf{S}$. This is clearly a contradiction.

2. $\phi = 0$. First, recall that if some jobs arrive in $(S_{k-1}, T_{k-1}]$, then $S_k = T_{k-1}$ and $\overrightarrow{W}(T_{k-1}) = \overrightarrow{W}(S_k) = \vec{\Sigma}(S_{k-1}, T_{k-1})$; if no job arrives in $(S_{k-1}, T_{k-1}]$, then S_n is the arrival time of the next job to arrive and $\overrightarrow{W}(S_k)$ is just the service requirement of that job. Therefore, in any case $\lim_{k\to\infty}(\overrightarrow{W}(S_k)/S_k) = \vec{0}$, using lemma 2.1.2(a) (in the first case) and lemma 2.2.1 (in the second one). For all $q \in \mathbf{Q}$, let $R_* = \min_{q\in\mathbf{Q}} \min_{m\in\mathbf{M}}\{R_m^q\colon\ R_m^q > 0\}$. Finally, we reach a contradiction as follows:

$$\varepsilon = \lim_{k\to\infty} \frac{T_k - S_k}{T_k} \leqslant \frac{1}{R_*} \lim_{k\to\infty} \frac{\sum_{q=1}^{Q} W^q(S_k)}{S_k} \frac{S_k}{T_k} = 0, \tag{4.19}$$

since the workload limit is 0 and $S_k < T_k$. This completes the proof of the proposition. □

In order to form the groups $\mathbf{G}_m^q(S_n, T_n, \vec{\rho}_n)$ of the jobs in batch $\mathbf{B}_n$, one needs to solve the linear program $(\mathrm{LP}_{\vec{\rho}_n})$ and compute the quantities $\Phi_m^q(\vec{\rho}_n)$, the end points of the intervals $\mathbf{D}_m^q(S_n, T_n; \vec{\rho}_n)$, and so the minimal and maximal job indices in each $\mathbf{G}_m^q(S_n, T_n; \vec{\rho}_n)$. It is easy to see that the complexity of these calculations does not depend on the number of jobs in the batch $\mathbf{B}_n$. In particular, the time to compute these quantities is bounded. Hence, rate stability is not affected. In the special case where the load vector $\vec{\rho}$ is known in advance, $\vec{x}^*(\vec{\rho})$ and $\Phi_m^q(\vec{\rho})$ can be computed once off-line. In the general case as $\vec{\rho}_n \to \infty$, we shall not need to recalculate the allocation parameters after the load estimate has stabilized enough.

Note that when the schedule π_G^* uses the load estimator $\vec{\rho}_n$ in (4.16) at the execution start of batch $\mathbf{B}_n$ – in order to determine the allocation parameters $\Phi_m^q(\vec{\rho}_n)$ – this estimator depends on the service requirements of jobs that have already been served in the *past*. Therefore, the groups $\mathbf{G}_m^q(S_n, T_n; \vec{\rho}_n)$ may be determined by knowing only the arrival times of the jobs present in the system, while no information about their service requirements is needed.

Remark 4.1 (Robust asymptotic optimality (RAO) and rate stability). The grouping schedule π_G^* described in the previous section and used in the current one is a particularly useful example of a potentially rich class of schedules that share the property of π_G^* described in proposition 4.2. Let us formalize this property in the following definition of *Robust Asymptotic Optimality* (RAO): A schedule π_{RAO}^* is robust and asymptotically optimal if

$$\lim_{k\to\infty} \frac{\tau(s_k, s'_k; \vec{\rho}_k)}{s'_k - s_k} = \delta(\vec{\rho}), \tag{4.20}$$

when $\lim_{k\to\infty}((s'_k - s_k)/s'_k) = \theta > 0$ and $\lim_{k\to\infty} \vec{\rho}_k = \vec{\rho} \in \mathbf{S}$. The time span $\tau(s_k, s'_k; \vec{\rho}_k)$ is now the time it takes to process the jobs arriving in the interval $(s_k, s'_k]$ (when scheduled under the scheme π_{RAO}^*) using $\vec{\rho}_n$ in computing the schedule, and including the switching times between service combinations. Any scheme π_B/π_{RAO}^* would be rate stable, since the proof of the previous proposition 4.3 would hold. For example, such RAO schedules could be randomized schemes for allocating the jobs to service vectors, trying to balance the load appropriately.

5. Extension to feed-forward networks of switched processing systems/nodes

The previous analysis of a single processing node extends directly to acyclic networks of such nodes. As a matter of fact, the chosen 'trace-based' stability analysis framework has been developed with an eye towards extension to networks in a straight-forward manner. By a network being acyclic, we mean that its nodes can be partitioned into consecutive levels or stages (where the levels are numbered in increasing order), so that each job always visits lower level nodes *before* higher level ones. Let us model the above networking situation as follows.

1. There are L node levels, indexed by $l \in \mathbf{L} = \{1, 2, \ldots\}$.
2. The set of nodes at level l is $\mathbf{N}_l$. The set of all nodes is $\mathbf{N} = \bigcup_{l\in\mathbf{L}} \mathbf{N}_l$.
3. For each node $n \in \mathbf{N}$, we denote by $\mathcal{L}(n) \in \mathbf{L}$ the level that the node is at. Therefore, $n \in \mathbf{N}_l \Leftrightarrow \mathcal{L}(n) = l$.
4. Node $n \in \mathbf{N}$ has a set $\mathbf{Q}_n$ of parallel input queues. Let $\mathbf{Q} = \bigcup_{n\in\mathbf{N}} \mathbf{Q}_n$ be the set of *all* queues in the network. Each queue is FIFO.
5. For each queue $q \in \mathbf{Q}$, we denote by $\mathcal{N}(q)$ the node this queue belongs to. Therefore, $q \in \mathbf{Q}_n \Leftrightarrow \mathcal{N}(q) = n$.
6. There are F traffic flow traces, indexed by $f \in \mathbf{F} = \{1, 2, \ldots, F\}$.
7. The structure of each *flow trace* $\mathcal{T}^f$ is the following:
   (a) Each job of $\mathcal{T}^f$ visits consecutively a sequence of K^f queues $\mathcal{Q}^f = (q_1^f, q_2^f, \ldots, q_k^f, \ldots, q_{K^f}^f)$. Therefore, each $\mathcal{T}^f$ job visits consecutively the nodes $(\mathcal{N}(q_1^f), \mathcal{N}(q_2^f), \ldots, \mathcal{N}(q_k^f), \ldots, \mathcal{N}(q_{K^f}^f))$ which belong to the following levels

$(\mathcal{L}(\mathcal{N}(q_1^f)), \mathcal{L}(\mathcal{N}(q_2^f)), \ldots, \mathcal{L}(\mathcal{N}(q_k^f)), \ldots, \mathcal{L}(\mathcal{N}(q_{K^f}^f)))$. The condition about the network being *acyclic* (feed-forward) is expressed by the requirement that

$$1 \leqslant \mathcal{L}\big(\mathcal{N}\big(q_1^f\big)\big) < \mathcal{L}\big(\mathcal{N}\big(q_2^f\big)\big) < \cdots < \mathcal{L}\big(\mathcal{N}\big(q_k^f\big)\big) < \cdots < \mathcal{L}\big(\mathcal{N}\big(q_{K^f}^f\big)\big) \leqslant L. \tag{5.1}$$

That is, each job visits one node per level and one queue per node; or at most L queues ($K^f \leqslant L$), one per level. Jobs of the same flow visit exactly the same queues. The ordered queue list $\mathcal{Q}^f$ defines the *path/route* of each T^f job through the network.

(b) Let t_j^f be the arrival time of the jth job of flow T^f to the *first queue* q_1^f on its path $\mathcal{Q}^f$.

(c) Let σ_j^{fq} be the service time requirement of the jth job on flow trace $\mathcal{T}^f$ at queue q.

(d) Therefore, the flow trace structure is simply

$$\mathcal{T}^f = \big\{\big(t_j^f;\ \big\{\sigma_j^{fq},\ q \in \mathcal{Q}^f\big\}\big),\ j \in \mathbb{Z}_+\big\}. \tag{5.2}$$

(e) The *flow load* $\{\rho^{fq},\ q \in \mathcal{Q}^f\}$, $f \in \mathbf{F}$, is expressed through the time averages (assumed to exist)

$$\rho^{fq} = \lim_{t\to\infty} \frac{\Sigma^{fq}(0,t)}{t}, \tag{5.3}$$

where

$$\Sigma^{fq}(0,t) = \sum_{j\in\mathbb{Z}_+} \sigma_j^{fq} \mathbb{1}_{\{t_j^f \in (0,t]\}} \tag{5.4}$$

is the total service requirement that flow trace $f \in \mathbf{F}$ carries into the network for queue $q \in \mathcal{Q}^f$ in the time interval $(0,t]$.

8. Let $\mathbf{S}_n$ be the convex combination of the service vectors (and their projections on the axes) of node $n \in \mathbf{N}$, that is, the stability region of this node, as per the results of the previous sections.

Given the above setup, observe that *superposition of traces* of two or more traffic flows going through the same queue results in a (legitimate) trace for that queue. The aggregate (over all flow traces) incoming load for queue $q \in \mathbf{Q}_n$ of node $n \in \mathbf{N}$ is

$$\rho^q = \lim_{t\to\infty} \frac{\sum_{f\in\mathbf{F}:\ q\in\mathcal{Q}^f} \Sigma^{fq}(0,t)}{t} = \sum_{f\in\mathbf{F}:\ q\in\mathcal{Q}^f} \rho^{fq}, \tag{5.5}$$

summing over all flows going through that queue. Then, the load vector of node $n \in \mathbf{N}$, based on the load coming into the network, is simply

$$\vec{\rho}_n = \big(\rho^q,\ q \in \mathbf{Q}_n\big) = \Big\{ \sum_{f\in\mathbf{F}:\ q\in\mathcal{Q}^f} \rho^{fq},\ q \in \mathbf{Q}_n \Big\}. \tag{5.6}$$

But is this really the load vector that node $n \in \mathbf{N}$ sees? The answer is: yes, if all the upstream nodes are rate stable. Specifically, suppose that t_j^{fq} is the arrival time of the jth job of flow trace $f \in \mathbf{F}$ to queue $q \in \mathcal{Q}^f$ and d_j^{fq} this job's departure from the node the queue resides at, upon service completion; that is, $\theta_j^{fq} = d_j^{fq} - t_j^{fq}$ is the sojourn time of the job through the queue's node. Of course, all these times depend on the service policies used at the upstream nodes. Note that $t_j^{fq} = t_j^f + \sum_{q \in \mathcal{Q}_{\text{up}}^f} \theta_j^{fq}$, where $\mathcal{Q}_{\text{up}}^f$ is the set of queues that are upstream of q on the path $\mathcal{Q}^f$ of flow $f \in \mathbf{F}$.

We analyze the stability of the network inductively on its levels or stages. Consider first the load seen at a node $n \in \mathbf{N}_1$ at level 1. This is exactly $\vec{\rho}_n$ for all $n \in \mathbf{N}_1$. Therefore, if $\vec{\rho}_n \in \mathbf{S}_n$, then $\lim_{t\to\infty}(\overrightarrow{W}_n(t)/t) = 0$, where $\overrightarrow{W}_n(t) = (W^q(t),\ q \in \mathbf{Q}_n)$ and $W^q(t)$ is the workload in queue q at time t, given that the node is operated under the MaxProduct, or MaxEmpty, or BatchAdapt policy. From lemma 2.3, the node is rate stable and the sojourn times grow sublinearly, that is, $\lim_{j\in\infty}(\theta_j^{fq}/t_j^q) = 0$ for any flow $f \in \mathbf{F}$ that crosses queue $q \in \mathbf{Q}_n$ of that particular node $n \in \mathbf{N}_1$. Consider now the aforementioned flow trace f and the next (downstream) queue q' it visits on $\mathcal{Q}^f$; that cannot reside on a node at level 1, but only at a higher level. The workload fed into queue q' directly (as opposed to a network entry) by flow trace f over the time period $(0, t]$ is

$$\sum_{j\in\mathbb{Z}_+} \sigma_j^{fq'} \mathbb{1}_{\{t_j^{fq'} \in (0,t]\}} = \sum_{j\in\mathbb{Z}_+} \sigma_j^{fq'} \mathbb{1}_{\{t_j^{fq} + \theta_j^{fq} \in (0,t]\}}.$$

However, since queue q is FIFO and $\lim_{j\to\infty}(\theta_j^{fq}/t_j^q) = 0$ (sojourn times grow sublinearly by lemma 2.3(3)), we get

$$\begin{aligned}\lim_{t\to\infty} \frac{\sum_{j\in\mathbb{Z}_+} \sigma_j^{fq'} \mathbb{1}_{\{t_j^{fq'} \in (0,t]\}}}{t} &= \lim_{t\to\infty} \frac{\sum_{j\in\mathbb{Z}_+} \sigma_j^{fq'} \mathbb{1}_{\{t_j^{fq} \in (0,t]\}}}{t} = \rho^{fq'} \\ &= \lim_{t\to\infty} \frac{\sum_{j\in\mathbb{Z}_+} \sigma_j^{fq'} \mathbb{1}_{\{t_j^{f} \in (0,t]\}}}{t}, \end{aligned} \tag{5.7}$$

where the first equality follows from lemma 2.1.2(a), and the last equality is because $t_j^{fq} = t_j^f$, since q is at level 1. By induction, this flow behavior propagates to subsequent nodes at later stages/levels that the flow trace traverses as it crosses the network. The point is that, if all the upstream nodes that a flow passes through to reach a certain queue are stable, then there is no load deficit or load that enters the network but does not reach its target queue. Hence, the traffic intensity that the queue sees is what enters the network. From the above discussion, we reach the following conclusion.

Proposition 5.1 (Trace superposition and network stability calculus). Given the above model of an acyclic network of switched processing nodes, we have that if $\vec{\rho}_n \in \mathbf{S}_n$ for all nodes $n \in \mathbf{N}$, then each node is rate stable (in interaction with the others), when the MaxProduct, or FastEmpty, or BatchAdapt service policy is used to operate each node (different nodes could use different policies).

The proof of this proposition is a direct consequence of the above modeling framework and the analysis of switched nodes in isolation given in previous sections. The key facts used are: (1) that the superposition of several deterministic traces through a queue results in a legitimate aggregate trace again, (2) rate stable nodes induce at most 'sublinearly growing' job sojourn times and, hence, (3) stability of upstream nodes results in downstream ones seeing the full load that enters the network destined for them (and is not starved by any unstable node upstream). Proposition 5.1 provides a stability 'calculus' for analyzing acyclic networks.

6. Concluding remarks

A general *trace-based* formulation framework has been developed for studying the stability problem of switched processing systems. It is as 'light' in assumptions as possible, in the sense that only the existence of a *long-term load* carried by each trace is required. An interesting fact is that no probabilistic super-structure is needed in this modeling framework. The queueing stability concept is tied to long-term job *inflow–outflow balance* on the traffic trace. The system is analyzed as a *deterministic dynamical* one with discrete-event trajectory jumps. The analysis is based on studying the asymptotics of evolution trajectories at large times.

Two key families of service policies are designed and demonstrated to maximize the sustainable throughput of the system. The first one, that of *cone policies*, partitions the workload space into specially structured cones and chooses a service vector depending on which cone the workload is in at each point in time. Two cone policies, *MaxProduct* and *FastEmpty*, have been studied and shown to stabilize the system. They generate different cone partitionings of the workload space. The second family of policies, called *BatchAdapt*, is one that adaptively batches up incoming jobs to the system and schedules each batch 'asympotically' optimally (for large batches) using a robust low-overhead schedule, which can also tolerate substantial switching times without compromising throughput. Extension to acyclic networks has been investigated also.

Within the trace-based framework for investigating switched processing systems, several other issues deserve serious consideration. What is the more general class of stabilizing cone policies? How can the framework be extended to deal with networks of switched nodes with feedback? How can other relevant performance questions (delay etc.) – other than the fundamental one of stability – be addressed within this modeling framework? How can the obtained results be applied in various technology areas (wireless, switching, etc.) given more special system structure? How can additional system properties be established under more restrictive assumptions and especially by imposing a probabilistic super-structure? Some such issues are partially addressed in [1] and we are currently exploring the above in some detail.

Perhaps more important is the issue of developing further the general trace-based modeling framework, so as to capture and analyze new queueing and processing structures, involving dynamic control of resources and scheduling of tasks under complicated operational constraints. We are currently investigating this possibility.

Acknowledgements

The authors would like to thank Prof. Jim Dai of the Georgia Institute of Technology and Prof. George Michailidis of the University of Michigan at Ann Arbor for fruitful discussions regarding this research during their stay at Stanford.

Appendix A.

A.1. Proofs of lemmas in section 2

Proof of lemma 2.1. Recall that $\lim_{t\to\infty}(\vec{\Sigma}(0,t)/t) = \vec{\rho} > \vec{0}$. We then have:

$$\lim_{n\to\infty}\frac{\vec{\Sigma}(s_n,t_n)}{t_n-s_n} = \lim_{n\to\infty}\frac{\vec{\Sigma}(0,t_n)}{t_n}\frac{t_n}{t_n-s_n} - \lim_{n\to\infty}\frac{\vec{\Sigma}(0,s_n)}{s_n}\frac{s_n}{t_n-s_n} = \vec{\rho}\frac{1}{\phi} - \vec{\rho}\left(\frac{1}{\phi}-1\right)$$
$$= \vec{\rho},$$
$$\lim_{n\to\infty}\frac{\vec{\Sigma}(s_n,t_n)}{t_n} = \lim_{n\to\infty}\frac{\vec{\Sigma}(0,t_n)}{t_n} - \lim_{n\to\infty}\frac{\vec{\Sigma}(0,s_n)}{s_n}\frac{s_n}{t_n} = \vec{\rho} - \vec{\rho}(1-\phi) = \vec{\rho}\phi.$$

Similarly,

$$\lim_{n\to\infty}\frac{\vec{\Sigma}(s_n,t_n)}{s_n} = \lim_{n\to\infty}\frac{\vec{\Sigma}(0,t_n)}{t_n}\frac{t_n}{s_n} - \lim_{n\to\infty}\frac{\vec{\Sigma}(0,s_n)}{s_n} = \vec{\rho}\frac{1}{1-\phi} - \vec{\rho} = \vec{\rho}\frac{\phi}{1-\phi},$$
$$\lim_{n\to\infty}\frac{\vec{\Sigma}(s_n,t_n)}{t_n} = \lim_{n\to\infty}\frac{\vec{\Sigma}(0,t_n)}{t_n} - \lim_{n\to\infty}\frac{\vec{\Sigma}(0,s_n)}{s_n}\frac{s_n}{t_n} = \vec{\rho} - \vec{\rho}\cdot 1 = \vec{0}.$$

Similarly,

$$\lim_{n\to\infty}\frac{\vec{\Sigma}(s_n,t_n)}{s_n} = \lim_{n\to\infty}\frac{\vec{\Sigma}(0,t_n)}{t_n}\frac{t_n}{s_n} - \lim_{n\to\infty}\frac{\vec{\Sigma}(0,s_n)}{s_n} = \vec{\rho}\cdot 1 - \vec{\rho} = \vec{0}.$$

For any $\pi \in \Pi$, $m \in \mathbf{M}$, $q \in \mathbf{Q}$, we have

$$\int_{s_n}^{t_n} \mathbb{1}_{\{\vec{R}_\pi(\tau)=\vec{R}_m\}}\mathbb{1}_{\{W^q(\tau)>0\}}\,\mathrm{d}\tau \leqslant t_n - s_n,$$

hence,

$$\lim_{n\to\infty}\frac{\int_{s_n}^{t_n}\mathbb{1}_{\{\vec{R}_\pi(\tau)=\vec{R}_m\}}\mathbb{1}_{\{W^q(\tau)>0\}}\,\mathrm{d}\tau}{t_n} = 0.$$

From (1.4) we have

$$W^q(t_n) - W^q(s_n) = \Sigma^q(s_n,t_n) - \sum_{m=1}^{M} R^q_m \int_{s_n}^{t_n}\mathbb{1}_{\{\vec{R}_\pi(\tau)=\vec{R}_m\}}\mathbb{1}_{\{W^q(\tau)>0\}}\,\mathrm{d}\tau.$$

Dividing by t_n and letting $n \to \infty$, we get

$$\lim_{n\to\infty} \frac{W^q(t_n) - W^q(s_n)}{t_n} = \lim_{n\to\infty} \frac{\Sigma^q(s_n, t_n)}{t_n} - \lim_{n\to\infty} \frac{\sum_{m=1}^{M} R_m^q \int_{s_n}^{t_n} \mathbb{1}_{\{\vec{R}_\pi(\tau)=\vec{R}_m\}} \mathbb{1}_{\{W^q(\tau)>0\}} \, \mathrm{d}\tau}{t_n}$$
$$= 0.$$

Moreover,

$$\lim_{n\to\infty} \frac{W^q(t_n) - W^q(s_n)}{s_n} = \lim_{n\to\infty} \frac{W_\pi^q(t_n) - W_\pi^q(s_n)}{t_n} \frac{t_n}{s_n} = 0.$$

Without loss of generality, suppose that for each n a job arrives into queue $q \in \mathbf{Q}$ at t_n. Let j_n be the index of that job, so it is sufficient to show that $\lim_{n\to\infty}(\sigma_{j_n}^q / t_n) = 0$. For each n, choose $\varepsilon_n > 0$ so that there are no arrivals during the time interval $[t - \varepsilon_n, t_n)$ and $\lim_{n\to\infty}(\varepsilon_n / t_n) = 0$. Then part 2(a) of this lemma implies that $\lim_{n\to\infty}(\sigma_{j_n}^q / t_n) = \lim_{n\to\infty}(\Sigma^q(t_n - \varepsilon_n, t_n)/t_n) = 0$. □

Proof of lemma 2.2. The proof follows directly from similar arguments to those in the proof of lemma 2.1.3.

Arguing by contradiction, suppose that $\limsup_{t\to\infty}(\tau(t) - t/t) = \phi > 0$. In particular, there exists an increasing unbounded sequence $\{t_n\}_{n=1}^{\infty}$ such that $\lim_{n\to\infty}((\tau(t_n) - t_n)/t_n) = \phi$. Lemma 2.1.1(b) implies that $\lim_{n\to\infty}(\vec{\Sigma}(t_n, \tau(t_n))/\tau(t_n)) = \phi\vec{\rho}$. But $\phi\rho^q > 0$ for all $q \in \mathbf{Q}$, which contradicts part 1 of this lemma. □

Proof of lemma 2.3. Recall that any policy $\pi \in \Pi$ preserves the FIFO discipline in each individual queue, though this might not be the case across distinct queues.

Because the queue is FIFO, we have $W^q(d_j^q) = \sum^q(t_j^q, d_j^q)$. Therefore,

$$\frac{W^q(d_j^q)}{d_j^q} = \frac{\sum^q(0, d_j^q)}{d_j^q} - \frac{\sum^q(0, t_j^q)}{t_j^q} \frac{t_j^q}{d_j^q}. \tag{A.1}$$

Taking the limits as $j \to \infty$ and assuming that $\lim_{t\to\infty}\{W^q(t)/t\} = 0$, we see that $\lim_{j\to\infty}\{t_j^q / d_j^q\} = 1$, using (1.3). This implies that for an arbitrarily small $\varepsilon > 0$, we eventually have

$$\frac{1-\varepsilon}{t_j^q} \leqslant \frac{1}{d_j^q} \leqslant \frac{1+\varepsilon}{t_j^q} \tag{A.2}$$

for all large enough j. If $\lim_{k\to\infty}\{j_k / t_{j_k}^q\} = \lambda_{\text{in}}^q$ exists on an increasing unbounded subsequence of job indices $\{j_k\}_{k=1}^{\infty}$, then using (A.2), multiplying by j_k, taking the limits as $k \to \infty$, and finally relaxing ε to zero, we get $\lambda_{\text{out}}^q = \lim_{j\to\infty}\{j_k / d_{j_k}^q\} = \lambda_{\text{in}}^q$.

Suppose now that $\{s_n\}_{n=1}^{\infty}$ is an increasing unbounded time sequence, such that $\lim_{n\to\infty}(W^q(s_n)/s_n) = \phi > 0$. For all n, let j_n be the index of the *first* job to depart *after* s_n. Then, $W^q(d_{j_n}^q) \geqslant W^q(s_n) + \Sigma(s_n, d_{j_n}^q) - \sigma_{j_n}^q = W^q(s_n) + \Sigma^q(0, d_{j_n}^q) - \Sigma^q(0, s_n) - \sigma_{j_n}^q$ (the inequality is strict if at time s_n job j_n has already started being processed). Dividing through by $d_{j_n}^q$, using (1.3) and the facts that $d_{j_n}^q > s_n$ and $\lim_{n\to\infty}(\sigma_{j_n}^q / d_{j_n}^q) = 0$

(see the proof of lemma 2.1.3), we get that $\liminf_{n\to\infty}(W(d^q_{j_n})/d^q_{j_n}) \geqslant \min\{\phi, \rho^q\} > 0$. Let $\{j_k\}_{j=1}^{\infty}$ be a subsequence of $\{j_n\}_{n=1}^{\infty}$ such that $\lim_{k\to\infty}(W(d^q_{j_k})/d^q_{j_k}) = \tilde{\phi} \geqslant \min\{\phi, \rho^q\} > 0$. Then, (A.1) implies that

$$\lim_{k\to\infty} \frac{t^q_{j_k}}{d^q_{j_k}} = 1 - \frac{\tilde{\phi}}{\rho^q} = \xi \in [0, 1). \tag{A.3}$$

In particular, for any small $\varepsilon > 0$, eventually we have

$$\frac{\xi - \varepsilon}{t^q_{j_k}} \leqslant \frac{1}{d^q_{j_k}} \leqslant \frac{\xi + \varepsilon}{t^q_{j_k}}. \tag{A.4}$$

If there exists a subsequence $\{j_l\}_{l=1}^{\infty}$ of $\{j_k\}_{k=1}^{\infty}$ such that $\lim_{l\to\infty}(j_l/t^q_{j_l}) = \lambda$ for some $\lambda > 0$, then multiplying (A.4) through by j_l, taking the limit as $l \to \infty$, and relaxing ε to 0, we get that $\lim_{l\to\infty}(j_l/d^q_{j_l}) = \lambda\xi < \lambda = \lim_{l\to\infty}(j_l/t^q_{j_l})$.

The result follows immediately from (A.1) and (1.3), which yields $\lim_{j\to\infty}\{t^q_j/d^q_j\} = 1$. □

Proof of lemma 2.4. Arguing by contradiction, suppose there exists an increasing unbounded subsequence $\{t_n\}_{n=1}^{\infty}$ of the original one $\{t_k\}_{k=1}^{\infty}$ with $\lim_{n\to\infty}((t_n - s^q_n)/t_n) = 0$. From lemma 2.1.2(c) we get $\lim_{n\to\infty}((W^q(t_n) - W^q(s^q_n))/s^q_n) = 0$, which together with $\lim_{n\to\infty}(W^q(t_n)/t_n) = \eta^q$ gives $\lim_{n\to\infty}(W^q(s^q_n)/s^q_n) = \eta^q > 0$. Using lemma 2.1.3, we then get $\lim_{n\to\infty}(W^q(s^{q-}_n)/s^q_n) = \eta^q > 0$, covering the possible case where a job arrival occurs at s^q_n and so $W^q(s^{q-}_n) \neq W^q(s^q_n)$. This forces $W^q(s^{q-}_n)$ to be positive for large n, contradicting the fact that $W^q(s^{q-}_n) = 0$. □

A.2. Rate-stability proof of the FastEmpty policies

In this appendix we provide the detailed proof of the fact that, when $\vec{\rho} \in \mathbf{S}$, any FastEmpty policy π_Δ of section 3.3 with $\Delta \in (0, 1/M]$ will keep the system rate stable, under any traces (1.1) satisfying (1.3).

Referring to the setup of section 3.3, we start by reformulating it in terms of the *dual program* ($\mathrm{DP}_{\vec{w}}$) to ($\mathrm{LP}_{\vec{w}}$) and the dual variables $\vec{y}$ to $\vec{w}$. This serves the purpose of identifying a simple *dual* characterization of service vectors that FastEmpty uses at any point in time. Specifically, for any $\vec{w} \in \mathbb{R}^Q_+$ the dual program ($\mathrm{DP}_{\vec{w}}$) to the linear program ($\mathrm{LP}_{\vec{w}}$) of section 3.3 is given by:

$$\begin{aligned} &\text{maximize } \vec{y}'\vec{w} \qquad (\delta(\vec{w}) = \max_{\vec{y}'}\{\vec{y}'\vec{w}\}) \\ &\text{subject to:} \quad \vec{y}'\vec{R}_m \leqslant 1, \quad \forall\, m \in \mathbf{M}, \text{ and } \vec{y} \geqslant \vec{0} \end{aligned} \tag{$\mathrm{DP}_{\vec{w}}$}$$

where $'$ denotes the transpose co-vector operation, and $\vec{y}'\vec{w} = \sum_{q=1}^{Q} y^q w^q$ is the *standard* inner product between $\vec{y}$ and $\vec{w}$, and $\vec{y}'\vec{R}_m = \sum_{q=1}^{Q} y^q R^q_m$ similarly. By the complementary slackness condition (see, for example, [13, p. 136]) we have that, if $\vec{x}^*(\vec{w})$

and $\vec{y}^*(\vec{w})$ are corresponding optimal solutions of ($\mathrm{LP}_{\vec{w}}$) and ($\mathrm{DP}_{\vec{w}}$) respectively, then $x_m^*(\vec{w}) > 0$ implies that $\vec{y}(\vec{w})'\vec{R}_m = 1$. In particular, from (3.32) we see that

$$m \in \mu(\vec{w}) \quad \Rightarrow \quad \vec{y}^{\,*}(\vec{w})'\vec{R}_m = 1. \tag{A.5}$$

Moreover, we can see that, if $\vec{\rho} \in \mathbf{S}$, then $\vec{y}'\vec{\rho} \leqslant 1$, for every *feasible* solution $\vec{y} \in \mathbb{R}_+^Q$ of the DP. This is due to the fact that $\vec{\rho}$ is dominated in $\mathbf{S}$ by some convex combination of the service vectors $\vec{R}_1, \ldots, \vec{R}_M$, hence, if $\vec{y}$ is a feasible solution of DP, then $\vec{y}'\vec{\rho} \leqslant \max_{m\in\mathbf{M}}\{\vec{y}'\vec{R}_m\} \leqslant 1$ (using also the constraints of the DP). In summary, we now have the following *key property* of the FastEmpty policy π_Δ: if $\vec{y}^*(\vec{w})$ is an optimal solution of ($\mathrm{DP}_{\vec{w}}$), then

$$\vec{\rho} \in \mathbf{S} \quad \Rightarrow \quad \vec{y}^*(\vec{w})'\vec{R}_m - \vec{y}^*(\vec{w})'\vec{\rho} = \gamma\left(\vec{y}^{\,*}(\vec{w})\right) \geqslant 0, \quad \text{for any } m \in \mu(\vec{w}). \tag{A.6}$$

Based on the above, proving that FastEmpty is rate stable when $\vec{\rho} \in \mathbf{S}$ follows very similar steps to the corresponding proof for MaxProduct of section 3.1, albeit its own intricacies of course. To underline the similarities, we use some equivalent notation. The main difference between the two proofs is that here the optimal emptying time $\delta(\overrightarrow{W}(t))$ of the workload plays the role of the inner product $\langle \overrightarrow{W}(t), \overrightarrow{W}(t)\rangle$ in the MaxProduct proof. In particular, to achieve our *goal* of showing that under FastEmpty we have $\lim_{t\to\infty}(\overrightarrow{W}(t)/t) = 0$, it is sufficient to show instead that $\lim_{t\to\infty}\delta(\overrightarrow{W}(t)/t) = 0$. Indeed, following the *method* introduced in section 3.1, we show that $\limsup_{t\to\infty}\delta(\overrightarrow{W}(t)/t) > 0$ is *impossible* when $\vec{\rho} \in \mathbf{S}$, establishing the needed key contradiction.

Specifically, assuming that $\vec{\rho} \in \mathbf{S}$ and $\limsup_{t\to\infty}\delta(\overrightarrow{W}(t)/t) > 0$, let $\{t_a\}_{a=1}^\infty$ be an increasing unbounded sequence such that $\lim_{a\to\infty}\delta(\overrightarrow{W}(t_a)/t_a) > 0$. 'Thinning' out $\{t_a\}_{a=1}^\infty$ we can get a subsequence $\{t_b\}_{b=1}^\infty$ of that, with

$$\lim_{b\to\infty}\frac{\overrightarrow{W}(t_b)}{t_b} = \vec{\eta} \neq \vec{0} \quad \text{and} \quad \delta(\vec{\eta}) = \limsup_{t\to\infty}\delta\left(\frac{\overrightarrow{W}(t)}{t}\right) > 0, \tag{A.7}$$

for some nonzero vector $\vec{\eta} \in \mathbb{R}_+^Q$. We establish a contradiction, by constructing another increasing unbounded sequence $\{s_d\}_{d=1}^\infty$ such that $\lim_{d\to\infty}(\overrightarrow{W}(s_d)/s_d) = \vec{\psi}$ for some nonzero vector $\vec{\psi} \in \mathbb{R}_+^Q$ with $\delta(\vec{\psi}) > \delta(\vec{\eta})$.

As in section 3.1, the *core step* in the construction of the above sequence is the existence of a subsequence $\{t_c\}_{c=1}^\infty$ of $\{t_b\}_{b=1}^\infty$ and an associated increasing unbounded sequence $\{s_c\}_{c=1}^\infty$ with $s_c < t_c$ for all c, such that the following key *properties* 1–3 hold:

1. $\lim_{c\to\infty}((t_c - s_c)/t_c) = \varepsilon \in (0, 1)$, that is, the length of the time interval $(s_c, t_c]$ grows linearly with t_c.
2. $\mu(\overrightarrow{W}(z)) \subseteq \mu(\vec{\eta})$ for all $z \in (s_c, t_c]$, hence, only service vectors $\vec{R}_m$ with $m \in \mu(\vec{\eta})$ may be used during the time interval $(s_c, t_c]$.
3. $W^q(z) > 0$ for all $z \in (s_c, t_c]$ when $\eta^q > 0$, $q \in \mathbf{Q}$, hence, queue q with $\eta^q > 0$ never empties in $(s_c, t_c]$.

The construction of these sequences is based on the *intuition* that, since $\overrightarrow{W}(t_c) \approx \vec{\eta} t_c$ with $\vec{\eta} \neq \vec{0}$, the time it would take to get to the workload state at time t_c after entering the associated cone is of linear order (this is formulated explicitly in claim A.2 below).

Recall that $T_m(s_c, t_c)$ is the total time the service vector $\vec{R}_m$ is used during the time interval $(s_c, t_c]$. Properties 2 and 3 imply that $\sum_{m \in \mathbf{M}} T_m(s_c, t_c) = \sum_{m \in \mu(\vec{\eta})} T_m(s_c, t_c) = t_c - s_c$ for all c. In particular, for $q \in \mathbf{Q}$ such that $\eta^q > 0$, the workload change over the time interval $(s_c, t_c]$ can be expressed as

$$W^q(t_c) - W^q(s_c) = \Sigma^q(s_c, t_c) - \sum_{m \in \mu(\vec{\eta})} R_m^q T_m(s_c, t_c), \tag{A.8}$$

for all c. Let $\vec{y}^* \in \mathbb{R}_+^Q$ be an *optimal solution* of $(\mathrm{DP}_{\vec{\eta}})$ such that $\eta^q = 0 \Rightarrow y^{*q} = 0$. Note that such an optimal solution exists, since $\eta^q = 0$ would eliminate the impact of y^{*q} in the cost $\vec{y}^{*\prime}\vec{\eta}$ of $(\mathrm{DP}_{\vec{\eta}})$ and setting $y^{*q} = 0$ would produce an optimal solution anyway. Then, multiplying both sides by y^{*q} and summing over all $q \in \mathbf{Q}$, we get

$$\vec{y}^{*\prime}\big(\overrightarrow{W}(t_c) - \overrightarrow{W}(s_c)\big) = \vec{y}^{*\prime}\vec{\Sigma}(s_c, t_c) - \sum_{m \in \mu(\vec{\eta})} \vec{y}^{*\prime}\vec{R}_m T_m(s_c, t_c). \tag{A.9}$$

The condition $\eta^q = 0 \Rightarrow y^{*q} = 0$ guarantees that the terms corresponding to such q do no affect the overall summation above. Since $\vec{y}^{*\prime}\vec{R}_m$ is the same for all $m \in \mu(\vec{\eta})$ from (A.5), we have

$$\vec{y}^{*\prime}\big(\overrightarrow{W}(t_c) - \overrightarrow{W}(s_c)\big) = \vec{y}^{*\prime}\vec{\Sigma}(s_c, t_c) - \vec{y}^{*\prime}\vec{R}_m(t_c - s_c) \tag{A.10}$$

for some $m \in \mu(\vec{\eta})$. Dividing by $t_c - s_c$ and letting $c \to \infty$, and using lemma 2.1.1(a) and property 1, we get

$$\lim_{c \to \infty} \vec{y}^{*\prime} \frac{(\overrightarrow{W}(t_c) - \overrightarrow{W}(s_c))}{t_c - s_c} = \vec{y}^{*\prime}\vec{\rho} - \vec{y}^{*\prime}\vec{R}_m = -\gamma(\vec{y}^*) \leqslant 0. \tag{A.11}$$

Since $\lim_{c \to \infty}(\overrightarrow{W}(t_c)/t_c) = \vec{\eta}$, using again property 1, we get

$$\lim_{c \to \infty} \vec{y}^{*\prime} \frac{\overrightarrow{W}(s_c)}{s_c} = \frac{\vec{y}^{*\prime}\vec{\eta} + \varepsilon\gamma(\vec{y}^*)}{1 - \varepsilon} > \delta(\vec{\eta}). \tag{A.12}$$

The inequality is due to $\varepsilon \in (0, 1)$ and $\vec{y}^{*\prime}\vec{\eta} = \delta(\vec{\eta})$, because $\vec{y}^*$ is an optimal solution of $(\mathrm{DP}_{\vec{\eta}})$. From (A.12), we can obtain (by successive 'thinnings' over the individual components of the workload vector) a subsequence $\{s_d\}_{d=1}^{\infty}$ of $\{s_c\}_{c=1}^{\infty}$ for which $\lim_{d \to \infty}(\overrightarrow{W}(s_d)/s_d) = \vec{\psi}$ (for some vector $\vec{\psi} \in \mathbb{R}_+^Q$), so $\vec{y}^{*\prime}\vec{\psi} = \lim_{d \to \infty}(\vec{y}^{*\prime}\overrightarrow{W}(s_d)/s_d) > \delta(\vec{\eta})$. Since $\vec{y}^*$ is feasible in $(\mathrm{DP}_{\vec{\eta}})$ it is also feasible in $(\mathrm{DP}_{\vec{\psi}})$ and therefore $\delta(\vec{\psi}) \geqslant \vec{y}^{*\prime}\vec{\psi} > \delta(\vec{\eta})$. Therefore [7], $\lim_{d \to \infty} \delta(\overrightarrow{W}(s_d)/s_d) = \delta(\lim_{d \to \infty}(\overrightarrow{W}(s_d)/s_d)) = \delta(\vec{\psi}) > \delta(\vec{\eta}) = \limsup_{t \to \infty} \delta(\overrightarrow{W}(t)/t)$, establishing a contradiction.

Proposition A.1 (Rate stability under FastEmpty). Let $\overrightarrow{W}(t)$ be the workload vector of the system operating under the FastEmpty policy π_Δ with an arbitrarily fixed $\Delta \in (0, 1/M]$, on any arbitrarily fixed traces (1.1) satisfying (1.3). Then,

$$\vec{\rho} \in \mathbf{S} \quad \Rightarrow \quad \lim_{t\to\infty} \frac{\overrightarrow{W}(t)}{t} = 0 \tag{A.13}$$

which, by lemma 2.3(1), implies that the system is rate stable.

Proof. In view of the previously discussed methodology, it is enough to construct sequences $\{t_c\}_{c=1}^{\infty}$ and $\{s_c\}_{c=1}^{\infty}$ satisfying properties 1–3. This is done after proving the following two important intermediate results.

Claim A.1. Let $\vec{w} \in \mathbb{R}_+^Q$ be an arbitrarily fixed nonzero vector. Then, for $\mu(*)$ as defined in (3.30) for FastEmpty, there exists a number $\xi > 0$, such that $\vec{v} \in \mathbb{R}_+^Q$ with $\|\vec{v} - \vec{w}\| < \xi$ implies $\mu(\vec{v}) \subseteq \mu(\vec{w})$.

Proof. Arguing by contradiction, let $m_0 \notin \mu(\vec{w})$, and suppose that for all $n = 1, 2, \ldots$ there exists $\vec{v}_n \in \mathbb{R}_+^Q$ such that $\|\vec{v}_n - \vec{w}\| < 1/n$ and $m_0 \in \mu(\vec{v}_n)$. In particular, given the way FastEmpty operates, for each $n = 1, 2, \ldots$ there exists an optimal solution $\vec{x}^*(\vec{v}_n)$ such that $x^*_{m_0}(\vec{v}_n) \geqslant \Delta\delta(\vec{v}_n)$. Note that from the definition of $\{\vec{v}_n\}_{n=1}^{\infty}$ it follows that the sequence $\{\vec{x}^*(\vec{v}_n)\}_{n=1}^{\infty}$ is bounded by $\delta(\vec{w} + \vec{1})$, so it has a convergent subsequence. Let $\{\vec{v}_k\}_{k=1}^{\infty}$ be a subsequence of $\{\vec{v}_n\}_{n=1}^{\infty}$, such that $\lim_{k\to\infty} \vec{x}^*(\vec{v}_k)$ exists and is equal to some $\vec{x} \in \mathbb{R}_+^M$. From the continuity of $\delta(\cdot)$ [7], $\vec{x}$ is clearly an optimal solution of $(\mathrm{LP}_{\vec{w}})$, and $x_{m_0} \geqslant \Delta\delta(\vec{w})$. Hence, $m_0 \in \mu(\vec{w})$. This is clearly a contradiction. □

Claim A.2. Suppose that for some increasing unbounded time sequence $\{t_n\}_{n=1}^{\infty}$ we have $\lim_{n\to\infty}(\overrightarrow{W}(t_n)/t_n) = \vec{\eta}$, for some $\vec{\eta} \in \mathbb{R}_+^Q$. Let the time $\tau_n < t_n$ be such that $\mu(\vec{W}(\tau_n)) \not\subseteq \mu(\vec{\eta})$ (if there is no such time point, let $\tau_n = 0$). Then, $\liminf_{n\to\infty}((t_n - \tau_n)/t_n) > 0$.

Proof. The proof follows from claim A.1 and is analogous to the proof of claim 3.1. □

The construction of the two sequences is finally done as follows. Recall that $\{t_b\}_{b=1}^{\infty}$ satisfies $\lim_{b\to\infty}(\overrightarrow{W}(t_b)/t_b) = \vec{\eta}$, where $\vec{\eta}$ is a nonzero vector in $\mathbb{R}_+^Q$. By thinning out this sequence we can get a subsequence $\{t_c\}_{c=1}^{\infty}$ for which the following hold:

$\hat{s}_c = \sup\{t < t_c\colon \mu(\overrightarrow{W}(t)) \not\subseteq \mu(\vec{\eta})\}$ and $\lim_{c\to\infty} \dfrac{t_c - s_c^{\text{entry}}}{t_c} = \varepsilon_1 \in (0, 1]$

(by claim A.2).

$s_c^q = \sup\{t < t_c\colon W^q(t) = 0\}$ and $\lim_{c\to\infty} \dfrac{t_c - s_c^q}{t_c} = \varepsilon_2^q \in (0, 1]$,

for $q \in \mathbf{Q}$ such that $\eta^q > 0$ (by lemma 2.4).

Let $s_c = \max\{s_c^{\text{entry}}, \{s_c^q\colon\ \eta^q > 0\}, (1-\varepsilon_3)t_c\}$ for some $\varepsilon_3 \in (0,1)$. Then the following properties are satisfied:

$$\lim_{c\to\infty} \frac{t_c - s_c}{t_c} = \varepsilon = \min\bigl\{\varepsilon_1, \bigl\{\varepsilon_2^q\colon\ \eta^q > 0\bigr\}, \varepsilon_3\bigr\} \in (0,1),$$
$$\mu\bigl(\overrightarrow{W}(z)\bigr) \subseteq \mu(\vec{\eta}) \quad \text{for all } z \in (s_c, t_c] \text{ for all } c,$$
$$\text{for } q \in \mathbf{Q} \text{ with } \eta^q > 0, \quad W^q(z) > 0 \quad \text{for all } z \in (s_c, t_c], \text{ for all } c.$$

This implies that $\{t_c\}_{c=1}^{\infty}$ and $\{s_c\}_{c=1}^{\infty}$ satisfy properties 1–3 of appendix A.2, thereby completing the proof of the proposition. □

References

[1] M. Armony, Queueing networks with interacting service resources, Ph.D. Dissertation, Stanford University (1999).

[2] M. Armony and N. Bambos, Queueing networks with interacting service resources, in: *Proc. of Allerton Conference 1999*, Urbana, 1999, pp. 42–51.

[3] F. Baccelli and P. Bremaud, *Elements of Queueing Theory* (Springer, Berlin, 1994).

[4] N. Bambos and J. Walrand, Scheduling and stability aspects of a general class of parallel processing systems, Adv. in Appl. Probab. 25 (1993) 176–202.

[5] N. Bambos and K. Wasserman, On stationary tandem queueing networks with job feedback, Queueing Systems 15 (1994) 137–164.

[6] S.L. Bell and R.J. Williams, Dynamic scheduling of a system with two parallel servers in heavy traffic with complete resource pooling: Asymptotic optimality of a continuous review threshold policy, Ann. Appl. Probab. 11 (2001).

[7] V. Böhm, On the continuity of the optimal policy set for linear programs, SIAM J. Appl. Math. 28(2) (1975) 303–306.

[8] A. Budhiraja and P. Dupuis, Simple necessary and sufficient conditions for the stability of constrained processes, SIAM J. Appl. Math. 59(5) (1998) 1686–1700.

[9] H. Chen and A. Mandelbaum, Discrete flow networks: bottleneck analysis and fluid approximations, Math. Oper. Res. 16 (1991) 408–446.

[10] J.G. Dai, On positive Harris recurrence of multiclass queueing netowrks: A unified approach via fluid limit models, Ann. Appl. Probab. 5 (1995) 49–77.

[11] J.G. Dai, Stability of fluid and stochastic processing networks, MaPhySto 9 (1999).

[12] J.G. Dai and B. Prabhakar, The throughput of data switches with and without speedup, in: *Proc. of IEEE INFOCOM 2000*, 2000, pp. 556–564.

[13] G.B. Dantzig and M.N. Thapa, *Linear Programming 1: Introduction* (Springer, Berlin, 1997).

[14] P. Dupuis and R. Atar, Optimally stabilizing controls for a deterministic network model, in: *Proc. of the Allerton Conference 1999*, Urbana, 1999.

[15] N. Gans and G.J. van Ryzin, Optimal control of a multiclass, flexible queueing system, Oper. Res. 45(5) (1997) 677–693.

[16] N. Gans and G.J. van Ryzin, Optimal control of a parallel processing queueing system, Adv. in Appl. Probab. 30 (1998) 1130–1156.

[17] J.M. Harrison, Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies, Ann. Appl. Probab. 8 (1996) 822–848.

[18] J.M. Harrison, The BIGSTEP approach to flow management in stochastic processing networks, in: *Stochastic Networks: Theory and Applications*, eds. F. Kelly, S. Zachary and I. Ziedins (Oxford Univ. Press, Oxford, 1996) pp. 57–90.

[19] N. Kahale and P.E. Wright, Dynamic global packet routing in wireless networks, in: *Proc. of IEEE INFOCOM 1997*, 1997, pp. 1414–1421.
[20] R. Leelahakriengkrai and R. Agrawal, Scheduling in multimedia DS-CDMA wireless networks, Technical Report ECE-99-3, ECE Department, University of Wisconsin, Madison, IEEE Trans. Vehicular Technol. (1999) to appear.
[21] R. Leelahakriengkrai and R. Agrawal, Scheduling in multimedia wireless networks, in: *17th Internat. Teletraffic Congress*, Salvador da Bahia, Brazil, 2001.
[22] C. Maglaras, Dynamic scheduling in multiclass queueing networks: stability under discrete-review policies, Queueing Systems 31 (1999) 171–206.
[23] C. Maglaras, Discrete-review policies for scheduling stochastic networks: fluid asymptotic optimality, Ann. Appl. Probab. 10(3) (2000) 897–929.
[24] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, Achieving 100% throughput in an input-queued switch, IEEE Trans. Commun. 47(8) (1999) 1260–1267.
[25] S. Meyn, Feedback regulation for sequencing and routing in multiclass queueing networks, in: *2000 IEEE Internat. Symposium on Information Theory*, Sorrento, Italy, 1999.
[26] A. Stolyar, MaxWeight scheduling in a generalized switch: state space collapse and workload minimization in heavy traffic (2001) submitted.
[27] L. Tassiulas and P.P. Bhattacharya, Allocation of interdependent resources for maximal throughput, Stochastic Models 16(1) (1999).
[28] L. Tassiulas and A. Ephremides, Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, IEEE Trans. Automat. Control 37(12) (1992) 1936–1948.
[29] K.M. Wasserman and T. Lennon Olsen, On mutually interfering parallel servers subject to external disturbances, Oper. Res. 49(5) (2001) 700–709.