Multi-Agent Adaptive Mechanism Leading to Optimal Real-Time Load Sharing

O. Gallay, M.-O. Hongler
EPFL, STI-IMT-LPM, Lausanne, Switzerland

Corresponding author: O. Gallay, Ecole Polytechnique Fédérale de Lausanne (EPFL), STI-IMT-LPM, Station 17, CH-1015 Lausanne, olivier.gallay@epfl.ch

Abstract. We propose a new real-time load sharing policy (LSP), which optimally dispatches the incoming workload according to the current availability of the operators. Optimality means here that the global service permanently requires the engagement of a minimum number of operators while still respecting due dates. To cope with inherent randomness due to operator failures as well as non-stationary fluctuating incoming workload, any optimal LSP rule will necessarily rely on real-time updating mechanisms. Accordingly, a permanent monitoring of the traffic workload, of the queue contents and of other relevant dynamic state variables is often realized by a central workload dispatcher. In this contribution, we abandon such a "classical" approach and we propose a fully decentralized algorithm which fulfils the optimal load sharing process. The underlying decentralized decisions rely on a "smart tasks" paradigm in which each incoming task is endowed with an autonomous routing decision mechanism. Incoming jobs hence possess, in this paper, the status of autonomous agents endowed with "local intelligence". Stigmergic interactions between these agents cause the optimal LSP to emerge. We emphasize that beside a manifest strict relevance for applications, our class of models is analytically tractable, a rather uncommon feature when dealing with multi-agent dynamics and complex adaptive logistics systems.

1 General Context

The reduction of manpower or other resource costs is an everlasting managerial challenge in any production and service network. Such contraction of the operating costs obviously relies on an optimized workload sharing between the available operators. Processing the full incoming load by using the minimum number of available operators, while still respecting given due dates, is clearly the basic optimization objective. The operator random failures as well as the non-stationary fluctuating incoming workload force the optimal load sharing policy (LSP) to be based on a permanent monitoring of the system state (*i.e.* queue contents, instantaneous traffic, etc). While this information updating process, on which our adaptive optimal LSP will be based, is often fulfiled by a central dispatcher, our present contribution shows how fully decentralized mechanisms, of multi-agent type, are also perfectly suitable to achieve the same objective.

A large body of the available literature focuses attention on the customer side. Thus, the problem consists in minimizing the customers' average waiting time and, therefore, the maximum total load on each server is minimized. Here, we adopt the complementary point of view of the service provider and try to minimize the number of engaged operators while nevertheless respecting due dates. Referred as adaptive load balancing, the above mentioned classical problem has first been addressed using centralized management (see [3] for instance) and more recently by using decentralized mechanisms (see [5] among others). Note that although our model does not, stricto sensu, optimize customer satisfaction, it allows however to bound the maximum waiting time in the system by an ad-hoc tuning of control parameters. While numerous aspects of load balancing and load sharing have been abundantly discussed over the last three decades, relatively little attention is devoted to information gathering costs. Along these lines, let us mention contribution [4], where with the aim to minimize the average waiting time, the authors take explicitly into account the monitoring costs. The ultimate goal in [4] is to find a tradeoff between the benefit and the costs of information gathering needed for any adaptive load sharing mechanism. To that purpose, an autonomous load sharing mechanism is derived, which adapts optimally the number of monitored servers to the current workload. This study is hence somehow related to the present work, where our aim is to optimally determine the number of servers to engage in order to face the current load.

The present work shares several similarities with well-known congestion control problems arising in the Internet [1, 2, 8, 12, 17] where one tries to regulate the data flows to avoid congestion at servers (operators in the present case and gateways in the Internet). In both cases, the ultimate goal is to simultaneously ensure queue stability and maximization of resource utilization (busy period here and throughput in the Internet framework). To that purpose, one implements feedback information flows to warn about server congestion. While the presence of randomness definitely favors flexible and decentralized management in both contexts, there exist however manifest differences. Indeed, the agent character is in the present paper carried by the tasks themselves while it is managed for the most part by the servers in the Internet. While in congestion control problems, fairness between the different users (in terms of throughput or delay) is essential, this feature is not required in the present case. Furthermore, it is common in congestion control mechanisms to use randomnization to discard packets when a buffer gets congested

and hence ensure that fairness between users. On the contrary, noise is used in our framework to dispatch the tasks between the servers, hence forming a noise-induced stability mechanism. Several congestion control rules [8, 12, 17] generate stable oscillations of the queue content and these will also be observed here. In both cases, the oscillations are due to the presence of information delay in the underlying controller, a phenomena thoroughly investigated for self-interested agents competing for common resources [14].

The dynamics to be discussed here exhibits optimal load sharing entirely due to decentralized history-based routing mechanisms. The complete lack of central management, the agents' autonomy, their ability to learn and the stigmergic agent type of interactions imply that our class of models belongs to the field of Complex Adaptive Logistics Systems (CALS). Instances of CALS might also emerge in the framework of supply chains. In [19], the need, in supply chain management, for coordination strategies leading to adaptive, flexible and collective behaviours is exhibited and it is showed that coherent global behaviour can be generated by using only elementary components with local interactions. This contribution explains how basic concepts and operational tools of Complex Adaptive Systems (CAS) fit naturally and efficiently to characterize the supply chains dynamics. In this context, contributions [18, 19] expose the dynamics of simple queueing systems (QSs) for which feedback loops and delays coexist and yield temporal oscillations of the queue contents. While the relevance and legitimacy of CAS have since long been emphasized in basic sciences (*i.e.* physics, chemistry and biology), it is remarkable that CAS also strongly enter into the engineering world, for example in logistics [13, 21], in traffic issues [15] and in production and service systems [9, 10, 11]. Note in addition that, besides its direct relevance to load sharing problems, the analytical tractability of the present class of models contributes to enrich the, so far, short list of analytically solvable CALS. In closing, we emphasize that among several possibilities to implement our algorithm, Radio-Frequency-Identification-Devices (RFID) attached to the incoming tasks provide a natural solution. The available RFID technology now directly allows for a wide implementation of local intelligence to circulating items in production systems, as it is testified in [6, 16, 20], which explore how this technology leads to an effective and efficient management of business processes.

In section 2, we describe our basic modelling framework. In section 3, we introduce our multi-agent type dynamic load sharing algorithm. In section 4, we study the emergence of self-organized stable load sharing, by using analytical considerations as well as simulation results. In the Appendix, we describe in more details the oscillatory behaviour that appears in the queue content dynamics.

2 Basic Modelling Framework

We consider a production center fed by an incoming flow of tasks modelled by a non-stationary, random renewal process with rate $\lambda(t)$. The production center is therefore a generic QS with N parallel servers. Each incoming task ζ_j ($j \in \mathbb{N}$) requires a specific amount $U_j > 0$ of processing time. The U_j's are characterized by i.i.d. random variables with general probability distribution, whose mean is fixed to 1.

The objective is to realize an optimal load sharing policy defined by:

($\mathscr{O}$): *Optimal Load Sharing Policy (LSP)*

i) **"Process the global incoming workload by permanently engaging the minimal number of available servers"** or equivalently, using QS terminology, **"maximize the busy period of the engaged servers".**

ii) **"Keep the average waiting time below a given level".**

To achieve the objective $\mathscr{O}$, one can rely either on a centralized solution (*i.e.* a central dispatcher) or on ad-hoc decentralized control mechanisms. Our aim here is to realize $\mathscr{O}$ by using a multi-agent decentralized framework. Such decentralization permanently ensures strong reactivity and high flexibility to cope with random and non-stationary environments. In the sequel, we assume non-preemptive LSP (*i.e.* a task cannot be transferred from one server to another after its execution has started).

Server Parameters. A server M_α, $\alpha \in \{1,2,...,N\}$, is characterized by:

i) its processing rate μ_α,

ii) its queue capacity parameter $\mathscr{C}_\alpha > 0$, that plays the role of a congestion threshold,

iii) a two states ("open" or "close") warning semaphore S_α, controlled by the queue capacity parameter $\mathscr{C}_\alpha$.

"Smart Task" Agent Character of Incoming Jobs. Each incoming task has the capability to:

i) record its sojourn time τ spent into the system (*i.e.* $\tau = W + V$, W being the waiting time in the queue and V the processing time),

ii) identify the server α, $\alpha \in \{1,2,...,N\}$, that did process the task,

iii) compute a set of individual dispatching probabilities p_α, $\alpha \in \{1,2,...,N-1\}$, that characterize, for each task, an autonomous routing strategy,

iv) read the state of the semaphores S_α attached to each server.

As we assumed the incoming task size to be i.i.d. with mean 1, the service times of M_α inherit the randomness and are hence also i.i.d. random variables with identical distribution and mean $\frac{1}{\mu_\alpha}$.

3 Multi-Agent Type Algorithm

A "smart task" behaves as follows:

i) *On entry:*
An incoming task first reads the state of S_1. If S_1 is open, the task enters M_1. If S_1 is closed, the task routing is: enter M_1 with probability p_1 and, with probability $(1-p_1)$, read the state of S_2 to tentatively join M_2. If S_2 is open then M_2 processes the task. If S_2 is closed, the task enters with probability p_2 into M_2 and with probability $(1-p_2)$ reads the state of S_3 to tentatively join M_3. The rule is then applied iteratively.

ii) *On exit:*
If the sojourn time $\tau_{j,\alpha}$ of the outgoing task ζ_j ($j \in \mathbb{N}$), processed by M_α, exceeds $\mathscr{C}_\alpha$ (*i.e.* $\tau_{j,\alpha} > \mathscr{C}_\alpha$) then ζ_j sets S_α to the state *"closed"*. In words, when a task processed by M_α has spent $\mathscr{C}_\alpha$ in the system, it triggers immediately the closing of S_α even if the processing is not yet completed. Conversely, on exit, provided $\tau_{j,\alpha} \leq \mathscr{C}_\alpha$, ζ_j sets S_α to the state *"open"*.

Note that, even when a semaphore S_α is closed, a p_α-based partial incoming traffic continues to be processed by a congested server M_α. The tasks joining such an overloaded server ultimately enable the reopening of the associated semaphore as soon as the workload becomes undercritical (*i.e.* $\tau_{j,\alpha} \leq \mathscr{C}_\alpha$).

The ability for each travelling task to monitor information left by predecessors and to process this information to autonomously decide its routing strategy confers to this dynamics a manifest adaptive multi-agent character (see Figure 1 for a summarizing sketch of the model).

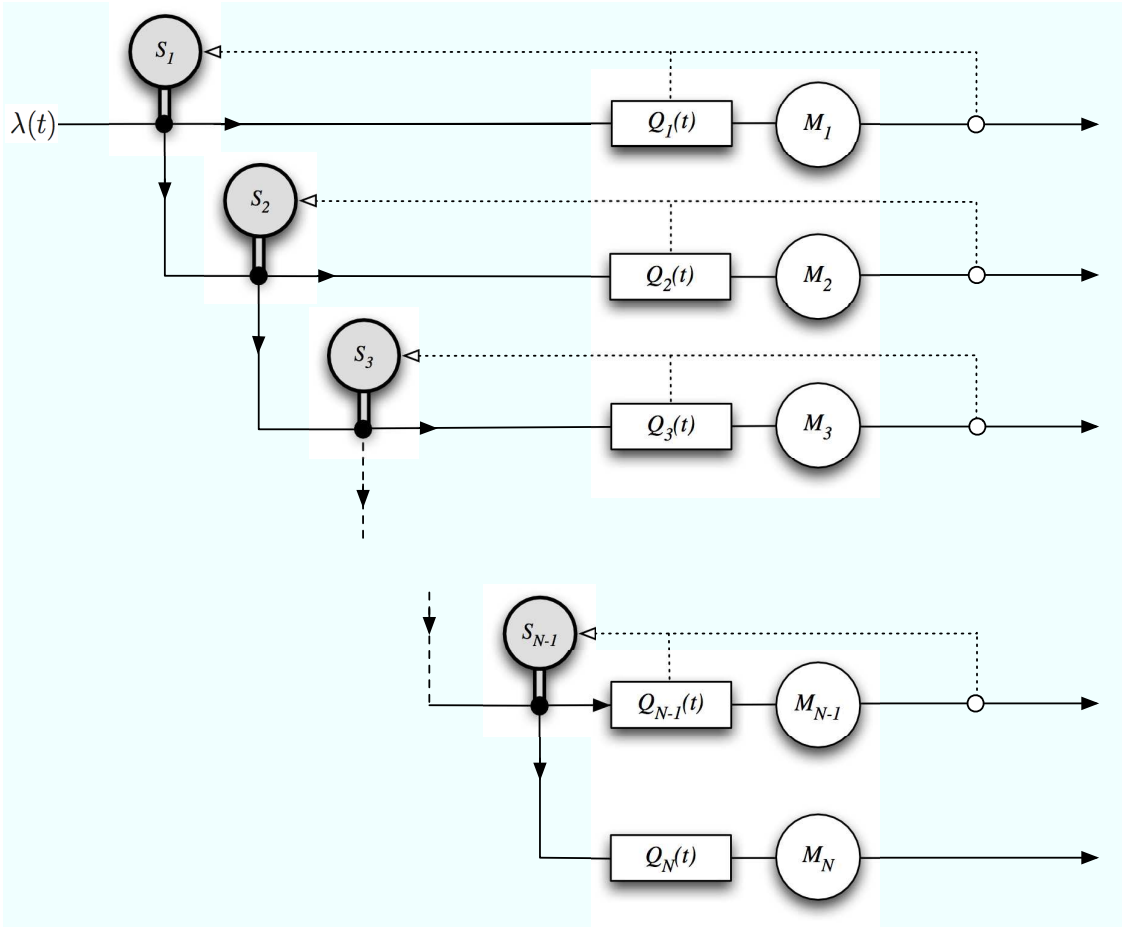

Figure 1: N-parallel servers queueing system with decentralized load sharing mechanism.

4 Emergence of Optimal Load Sharing Dynamics

From now on, we assume that the number N of potentially available servers is sufficient to always handle the offered workload, *i.e.*

$$\frac{\lambda(t)}{\sum_{\alpha=1}^{N}\mu_\alpha} < 1.$$

By the construction of the multi-agent dynamics, given in section 3, our LSP automatically ensures permanently *i)* the engagement of the minimal number of servers and *ii)* queue stability. As exposed in the Appendix, we observe (see Figure 2) that for large enough $\mathscr{C}_\alpha$'s, $\alpha \in \{1,...,N\}$, the queue contents exhibit stable temporal oscillations whose maximum values are given by:

$$Q_{\max,1} = \mathscr{C}_1\lambda(t) - 1; \qquad Q_{\max,\alpha} = \mathscr{C}_\alpha\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i) - 1,\ \alpha \in \{2,...,N\}. \tag{1}$$

Tuned by the control parameters $\mathscr{C}_\alpha$, $\alpha \in \{1,...,N\}$, these maximum possible queue contents can be used to

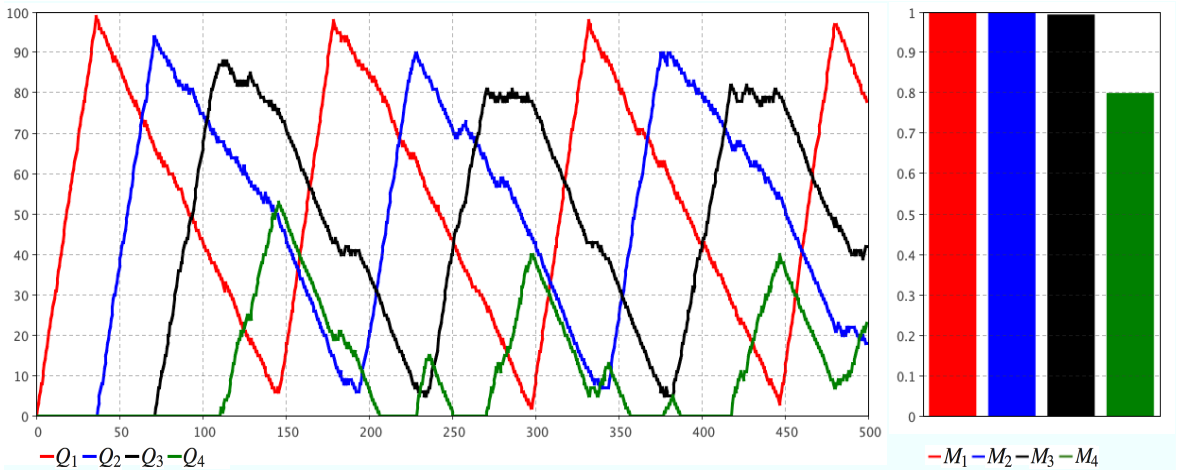

Figure 2: *Left*: Temporal evolution of the queue contents for interarrival times uniformly distributed in $[0.16;0.36]$ ($\lambda(t) = 3.8$), $K = 10$, service times uniformly distributed in $[0.5;1.5]$ ($\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$), $\mathscr{C}_1 = \mathscr{C}_2 = \mathscr{C}_3 = \mathscr{C}_4 = 26$ and $\varepsilon_\alpha = 0.22 - 0.7(\alpha - 1)$, $\alpha \in \{1,2,3\}$. *Right*: Corresponding server utilization.

calibrate the waiting room sizes and hence, due to Little's law, to limit the task waiting times. In particular, tasks subject to deadlines can be handled within the present framework. As shown in the Appendix, it is here worth to emphasize that the queue content of the engaged servers never vanishes, thus ensuring maximum busy period and hence optimal load sharing.

Let us now discuss in more details the role played by the dispatching probabilities p_α, introduced in Section 2. Remember that a congested server continues to be fed by a reduced incoming flow with rate:

$$\begin{cases} p_1\lambda(t) & \text{for server } M_1, \text{ and} \\ p_\alpha\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i) & \text{for server } M_\alpha,\ \alpha \in \{2,...,N\}. \end{cases}$$

Consequently, after a congestion occurs, the queue in front of the congested server will effectively decrease iff the following condition is satisfied:

$$\begin{cases} p_1\lambda(t) - \mu_1 < 0 & \text{for server } M_1, \text{ and} \\ p_\alpha\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i) - \mu_\alpha < 0 & \text{for server } M_\alpha,\ \alpha \in \{2,...,N\}. \end{cases} \tag{2}$$

To fulfil condition (2) the dispatching probabilities p_α, $\alpha \in \{1,...,N-1\}$, have to be chosen as:

$$p_1 = \frac{\mu_1}{\lambda(t)} - \varepsilon_1$$

and

$$p_\alpha = \frac{\mu_\alpha}{\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i)} - \varepsilon_\alpha, \quad \alpha \in \{2,...,N-1\},$$

with $\frac{\mu_1}{\lambda(t)} > \varepsilon_1 > 0$ and $\frac{\mu_\alpha}{\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i)} > \varepsilon_\alpha > 0$, $\alpha \in \{2,...,N-1\}$. This choice ensures stability of the queue contents. The smaller the value of the ε_α's is, the closer to optimality the load sharing is (*i.e.* the busy period of the engaged server M_α converges to 1 when $\varepsilon_\alpha \to 0$). A too drastic reduction of the partial traffic (ε_α large $\Rightarrow p_\alpha$

small) yields poor reactivity of the system. Indeed, the resulting long delay before the reopening of the semaphore is likely to empty the queue, thus leading to a decrease of the busy period.

As noted in [4], the incoming flow rate $\lambda(t)$ can itself be estimated, in real-time, by elementary agent-to-agent interactive mechanisms. This ultimately enables each task ζ_j, $j \in \mathbb{N}$, to estimate autonomously the ad-hoc dispatching probabilities $\overline{p}_{\alpha,j}$, $\alpha \in \{1,...,N-1\}$, which characterize its routing strategy. With this, our load sharing algorithm becomes fully decentralized, all routing decisions being taken by the circulating items themselves. One possibility to implement such a decentralized traffic estimation reads as follows:

Multi-Agent Type Traffic Load Estimator. Each task ζ_i ($i \in \mathbb{N}$) stores, upon arrival, its entry time t_i in the system on a register permanently accessible to the other tasks. The traffic estimator $\overline{\lambda}_j(t)$, computed by the incoming task ζ_j ($j \in \mathbb{N}$), relies on an observation window of size K. ζ_j reads the entry-time t_{j-K} of the K^{th} preceding task and estimates the instantaneous traffic by:

$$\overline{\lambda}_j(t) = \frac{t_j - t_{j-K}}{K}.$$

As clearly illustrated in Figure 2, this very rough estimation of the global incoming traffic is sufficient to complete the overall objective $\mathscr{O}$ and this, despite the underlying randomness. There is obviously an optimal trade-off to select an appropriate value of the observation window K. Valuing mostly the reactivity, we prefer small values of K for the observation window. As illustrated in Figure 3, small values of K lead indeed to highly reactive response (*i.e.* the length of the transient adaptive phase is almost negligible). This is in particular perfectly suitable for non-stationary incoming traffic load.

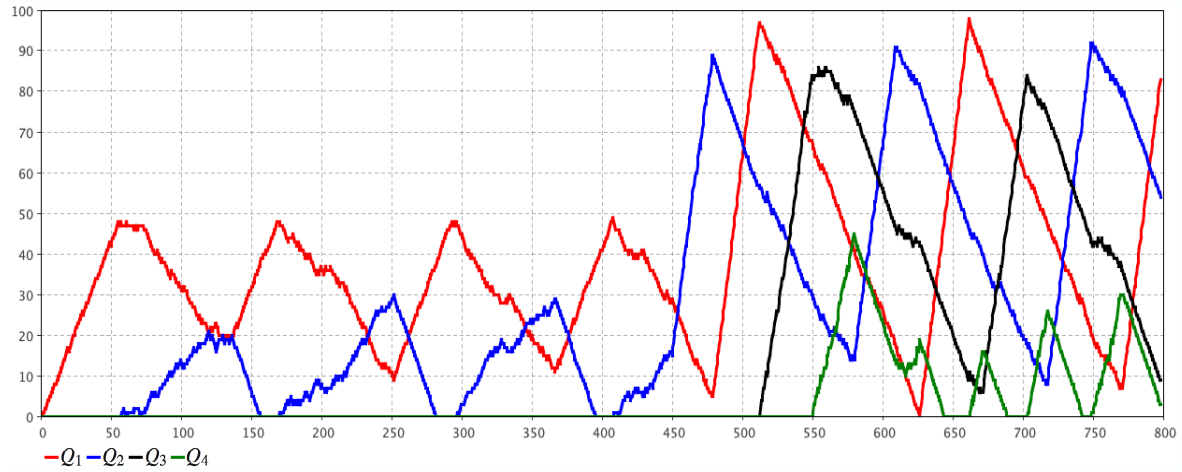

Figure 3: Temporal evolution of the queue contents for service times uniformly distributed in $[0.5;1.5]$ ($\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$), $\mathscr{C}_1 = \mathscr{C}_2 = \mathscr{C}_3 = \mathscr{C}_4 = 26$ and $\varepsilon_\alpha = 0.22 - 0.7(\alpha - 1)$, $\alpha \in \{1,2,3\}$. Interarrival times uniformly distributed in $[0.32;0.72]$ ($\lambda(t) = 1.9$) for $0 \leq t < 450$ and uniformly distributed in $[0.16;0.36]$ ($\lambda(t) = 3.8$) for $t \geq 450$, $K = 10$.

Quantitatively, the length of the adaptive phase thus only depends on the effective delay between the time a congestion effectively occurs and the time it is detected. This delay, for server M_α, is equal to $\mathscr{C}_\alpha$ (see the Appendix for more details). Note that depending on the specific management issues, larger observation windows K could also be selected whenever smooth reactions are required for system reliability or to avoid large set-up costs. Observe that most internet congestion control mechanisms rely on relatively large values of K for the observation window to smoothly react to bursty traffic.

5 Conclusion and Perspectives

In a competitive environment, to attract new and to keep loyal customers is the basic concern of any service provider, which definitely requires a high service customization to match all specific demands. Service customization affects both the quantity and the nature of the incoming demands. Focusing on quantitative aspects, one should clearly expect that high service customization necessarily leads to non-stationary and highly fluctuating rate of the service demand. Hence, constructing efficient service policies able, in such random and time-dependent environments, to entirely fulfil customer satisfaction while maintaining the operating costs at the lowest possible level is definitely a complex challenge. The ubiquitous non-stationary and fluctuating nature of the underlying demand imposes flexibility and reactivity to be key characteristics of any efficient algorithm required by the system management. Among other classical problems, we focus here on the construction of an optimal service policy

which enables the sharing of the global incoming workload between a set of available servers. In general, such a load sharing policy will be achieved by a central operator which dispatches, by an on-going real-time information gathering of a set of relevant system state variables, the incoming jobs among the available servers. Leaving aside such a centralized point of view and hence following a vigorous recent trend emerging in production and service systems, we explicitly show how the same task can also be perfectly realized using a fully decentralized algorithm. Our basic idea relies on a multi-agent perspective implying the service management to be performed by the jobs themselves (*i.e.* "smart tasks" paradigm). We are able to explicitly show how autonomous smart tasks can ensure, in real-time, that the global load is processed by the minimum number of engaged operators while permanently avoiding the system to get congested. The intrinsic simplicity of our algorithm, the analytical tractability of our models which offers an intimate understanding of the operating dynamics and, finally, the possibility, using RFID (*i.e.* Radio-Frequency-Identification-Devices), to confer an autonomous agent character to incoming jobs, clearly suggest how the implementation could actually be realized. Our present basic model offers several possibilities for refinements to cover realistic situations. For example, "fairness" issues could be addressed by adding an additional mechanism which would ultimately ensure that all incoming tasks wait in average the same time before being served.

6 Acknowledgments

This work is partially supported by the FNS (Fonds National Suisse pour la Recherche) under grant n^o 200020-117608/1.

7 Appendix: Queue Content Oscillatory Behaviour - Siphon Dynamics

We discuss here in more details the emergence of the temporal oscillations observed for the queue contents, as illustrated in Figure 2. We focus, for the discussion of this temporal oscillatory behaviour, on a deterministic approach. This approach, due to the law of large numbers (LLN), is also relevant in presence of fluctuations when the $\mathscr{C}_\alpha$'s, $\alpha \in \{1,...,N\}$, are sufficiently large (*i.e.* quasi-deterministic stable cyclo-stationary queue oscillations emerge independently of the inter-arrival and service time distributions). Note qualitatively that the relative importance of the fluctuations around the task average sojourn time (which is the sum of the preceding tasks individual processing times) decreases for large queue content $Q_\alpha(t)$ (a quantitative characterization is given in [7]). Figure 4 explicitly exhibits that the larger the $\mathscr{C}_\alpha$'s are, the smoother the oscillations are.

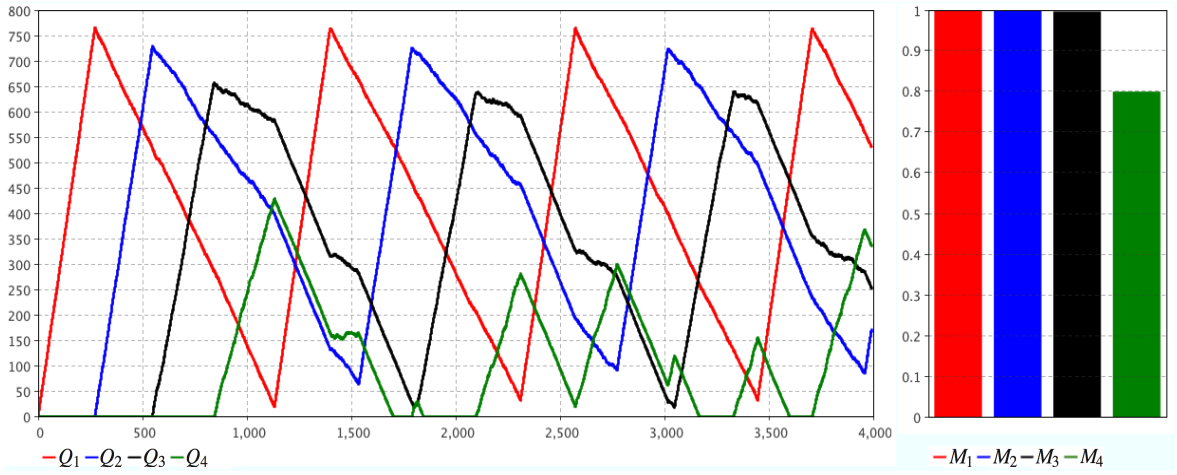

Figure 4: *Left*: Temporal evolution of the queue contents for interarrival times uniformly distributed in $[0.16;0.36]$ $(\lambda(t) = 3.8)$, $K = 10$, service times uniformly distributed in $[0.5;1.5]$ $(\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1)$, $\mathscr{C}_1 = \mathscr{C}_2 = \mathscr{C}_3 = \mathscr{C}_4 = 200$ and $\varepsilon_\alpha = 0.22 - 0.7(\alpha - 1)$, $\alpha \in \{1,2,3\}$. The smoothing effect due to the underlying LLN is manifestly observable by comparing Figures 2 and 4. *Right*: Corresponding server utilization.

Along the lines exposed in [7], we start by characterizing the oscillations of the first queue content $Q_1(t)$. During an initial phase, $Q_1(t)$ increases at rate $\lambda(t) - \mu_1$, the whole traffic $\lambda(t)$ is indeed dispatched to M_1, which is not yet overloaded. M_1 is considered as congested when $Q_1(t)$ reaches the level $\mathscr{C}_1\mu_1 - 1$. Indeed, at this time, a newly incoming task ζ_j will spend in average $\mathscr{C}_1$ in the system (*i.e.* its mean waiting time will be equal to $(\mathscr{C}_1\mu_1 - 1)\frac{1}{\mu_1} = \mathscr{C}_1 - \frac{1}{\mu_1}$ and its processing time will be equal in average to $\frac{1}{\mu_1}$). The queue $Q_1(t)$ reaches its auto-siphoning threshold when the congestion is first detected, which happens when ζ_j did wait $\mathscr{C}_1$ in the system. This then starts a second operating phase during which $Q_1(t)$ decreases at rate $p_1\lambda(t) - \mu_1$. This second phase lasts until a task detects that M_1 is not congested anymore, this happens after a time delay $\mathscr{C}_1$ initiated when $Q_1(t)$ reached again the level $\mathscr{C}_1\mu_1 - 1$. The alternance between these two operating phases creates queue content stable

oscillations whose amplitude Δ_1 and period Π_1 are respectively given by :

$$\Delta_1 = (1-p_1)\mathscr{C}_1\lambda(t)$$

and

$$\Pi_1 = \mathscr{C}_1\left[2+\frac{\mu_1-p_1\lambda(t)}{\lambda(t)-\mu_1}+\frac{\lambda(t)-\mu_1}{\mu_1-p_1\lambda(t)}\right].$$

The maximum and minimum values of these oscillations are given respectively by Eq. (1) and

$$Q_{\min,1} = p_1\mathscr{C}_1\lambda(t)-1.$$

To understand the underlying delay mechanism, it is enlightening to visualize the queue dynamics by using the hydrodynamic analogy sketched in Figure 5. We emphasize that contrary to the flow dynamics discussed in [7]

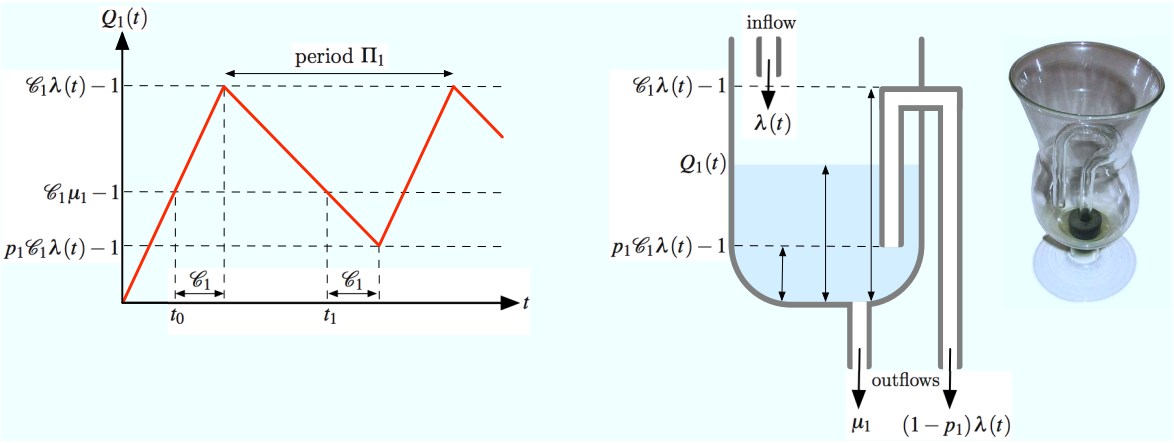

Figure 5: Hydrodynamic analogy. *Left:* The task entering at t_0 is the first one of a whole cluster $\mathscr{G}$ of tasks that will detect congestion. This task triggers the alternation of $Q_1(t)$ from the increasing to the decreasing state at $t_0+\mathscr{C}_1$. The last task belonging to cluster $\mathscr{G}$ is the one entering the system just before t_1 and triggers the switch of $Q_1(t)$ from the decreasing to the increasing state at $t_1+\mathscr{C}_1$. This simple delay dynamics repeats and creates stable oscillations of the queue content. *Right:* The "Tantalus glass" siphon model. The water level corresponds to the queue length $Q_1(t)$. The continuous inflow and outflow rates are respectively given by $\lambda(t)$ and μ_1. The periodic alternate siphoning outflow is $(1-p_1)\lambda(t)$. The siphon leaves a water residue of height $p_1\mathscr{C}_1\lambda(t)-1$, due to the continuous inflow during $\mathscr{C}_1$. The effective siphon length is $(1-p_1)\mathscr{C}_1\lambda(t)$.

where there is a feedback loop fed by physical items, here the feedback is purely informational (*i.e.* the items leave the system after service but deliver an indicative feedback to the following tasks).

The oscillation frequency of $Q_\alpha(t)$, $\alpha\in\{2,...,N\}$, are identical to $Q_1(t)$, hence

$$\Pi_\alpha = \Pi_1, \quad \alpha\in\{2,...,N\}.$$

This is illustrated in Figures 6 and 7 (respectively corresponding to Figures 2 and 4), where we exhibit the Fourier components (*i.e.* the spectrum) of the queue dynamics obtained by simulation for small, respectively large, values of $\mathscr{C}_\alpha$, $\alpha\in\{1,...,N\}$. As expected, for large values of $\mathscr{C}_\alpha$, the spectrum exhibits a sharp mode (*i.e.* the signal-to-noise ratio is enhanced).

For servers M_α, $\alpha\in\{2,...,N\}$, the oscillations exhibit an additional structure. Namely for $Q_\alpha(t)$, $\alpha\in\{2,...,N\}$, it exists an alternation between three distinct operating phases:

i) When all the servers M_β, $\beta\in\{1,...,\alpha-1\}$, are overloaded and hence their semaphores are closed, M_α receives a traffic with rate:

$$\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i).$$

Indeed, from the full incoming workload $\lambda(t)$, one has to substract the p_α-based partial traffics, that feed the congested servers. As a consequence, during this phase, $Q_\alpha(t)$ increases at rate $\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i)-\mu_\alpha$.

ii) Once server M_α becomes congested, $Q_\alpha(t)$ starts to empty. Provided all servers M_β, $\beta\in\{1,...,\alpha-1\}$, remain congested, the incoming traffic continues to be dispatched to M_α. As M_α is congested, it only receives a p_α-based part of this traffic and $Q_\alpha(t)$ hence decreases at rate $p_\alpha\lambda(t)\prod_{i=1}^{\alpha-1}(1-p_i)-\mu_\alpha$.

iii) Whenever one among the servers M_β, $\beta\in\{1,...,\alpha-1\}$, is no longer congested (therefore its semaphore has been reopened), this server attracts the full incoming workload and hence M_α is not fed anymore. Thus, $Q_\alpha(t)$ decreases at rate $-\mu_\alpha$.

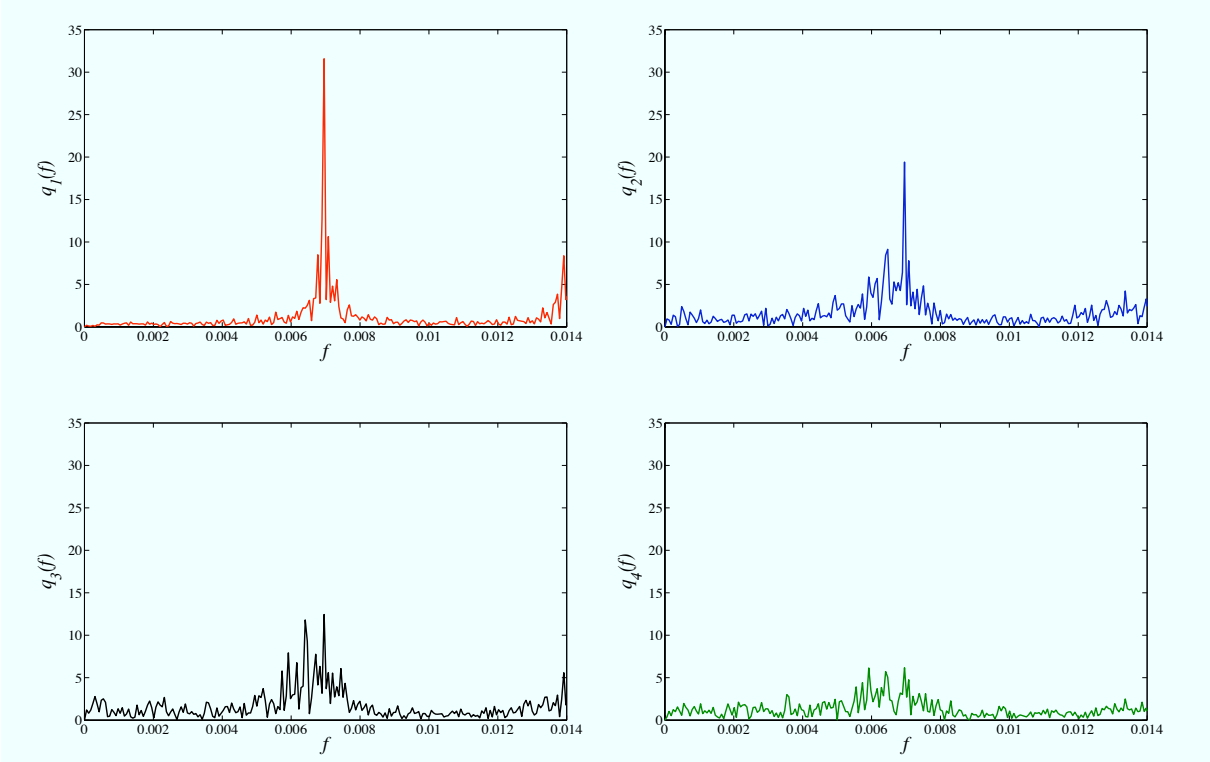

Figure 6: Spectrum of the queue content dynamics for interarrival times uniformly distributed in $[0.16;0.36]$ ($\lambda(t) = 3.8$), $K = 10$, service times uniformly distributed in $[0.5;1.5]$ ($\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1$), $\mathscr{C}_1 = \mathscr{C}_2 = \mathscr{C}_3 = \mathscr{C}_4 = 26$ and $\varepsilon_\alpha = 0.22 - 0.7(\alpha - 1)$, $\alpha \in \{1,2,3\}$.

The alternance between these three phases is completely determined by the queue dynamics of the yet engaged servers. Basically, the time at which a queue starts to empty triggers the feeding of the next server to be engaged. The oscillatory behavior of $Q_\alpha(t)$, $\alpha \in \{2,...,N\}$, is characterized by maximum and minimum values given respectively by Eq. (1) and:

$$Q_{\min,\alpha} = \mathscr{C}_\alpha p_\alpha \lambda(t) \prod_{i=1}^{\alpha-1} (1 - p_i) - 1, \quad \alpha \in \{2,...,N\}.$$

Consequently, the amplitude of these queue content oscillations is given by:

$$\Delta_\alpha = \mathscr{C}_\alpha \lambda(t) \prod_{i=1}^{\alpha} (1 - p_i), \quad \alpha \in \{2,...,N\}.$$

8 References

[1] Abdel-Jaber, H., Woodward, M., Thabtah, F. and Abu-Ali, A.: *Performance Evaluation for DRED Discrete-Time Queueing Network Analytical Model*. Journal of Network and Computer Applications, 31 (2008), 750-770.

[2] Aweya, J., Ouellette, M. and Montuno, D. Y.: *A Control Theoretic Approach to Active Queue Management*. Computer Networks, 36 (2001), 203-235.

[3] Bonomi, F. and Kumar, A.: *Adaptive Optimal Load Balancing in a Nonhogeneous Multiserver System with a Central Job Scheduler*. IEEE Transactions on Computers, 39/10 (1990), 1232-1250.

[4] Breitgand, D., Cohen, R., Nahir, A. and Raz, D.: *Cost Aware Adaptive Load Sharing*. Lecture Notes in Computer Science, 4725 (2007), 208-224.

[5] Breitgand, D., Cohen, R., Nahir, A. and Raz, D.: *On Fully Distributed Adaptive Load Balancing*. Lecture Notes in Computer Science, 4785 (2007), 74-85.

[6] Chow, H. K. H., Choy, K. L. and Lee, W. B.: *A Dynamic Logistics Process Knowledge-Based System - An RFID Multi-Agent Approach*. Knowledge-Based Systems, 20 (2007), 357-372.

[7] Filliger, R. and Hongler, M.-O.: *Syphon Dynamics - A Soluble Model of Multi-Agents Cooperative Behavior*. Europhysics Letters, 70/3 (2005), 285-291.

[8] Floyd, S. and Jacobson, V.: *Random Early Detection Gateways for Congestion Avoidance*. IEEE/ACM Transactions on Networking, 1/4 (1993), 397-413.

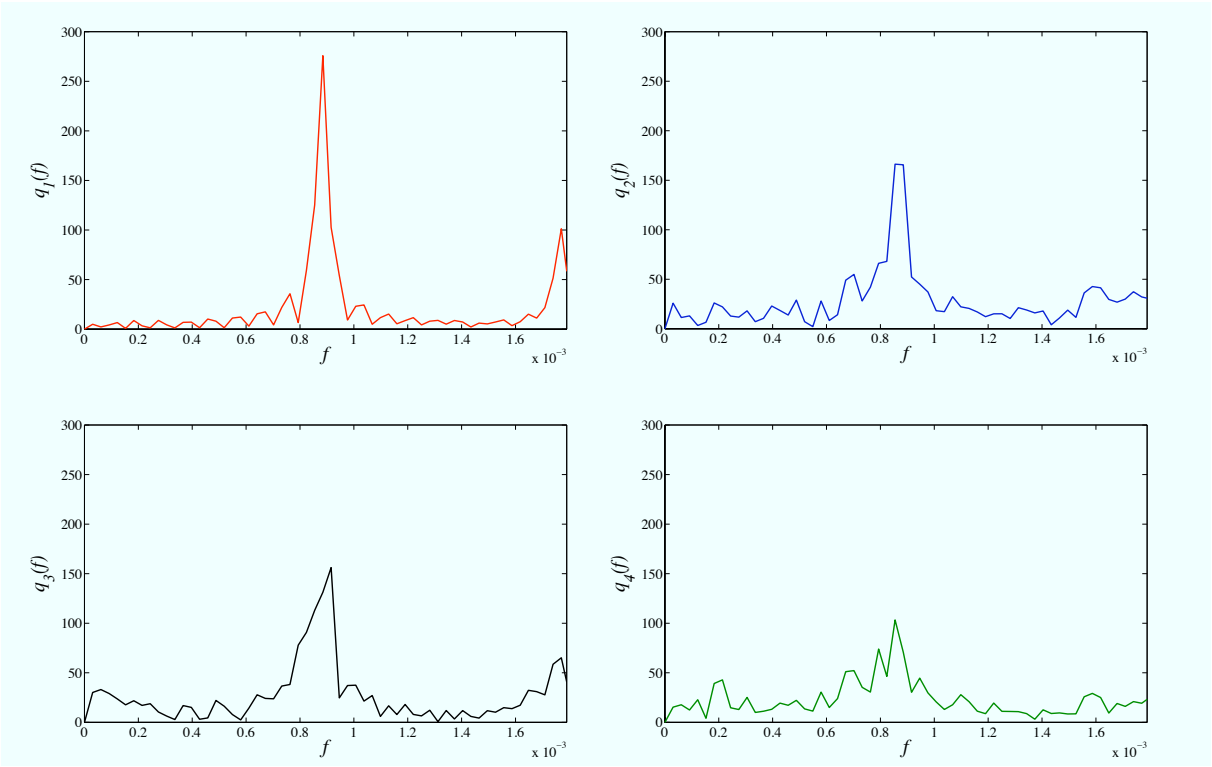

Figure 7: Spectrum of the queue content dynamics for interarrival times uniformly distributed in $[0.16; 0.36]$ $(\lambda(t) = 3.8)$, $K = 10$, service times uniformly distributed in $[0.5; 1.5]$ $(\mu_1 = \mu_2 = \mu_3 = \mu_4 = 1)$, $\mathscr{C}_1 = \mathscr{C}_2 = \mathscr{C}_3 = \mathscr{C}_4 = 200$ and $\varepsilon_\alpha = 0.22 - 0.7(\alpha - 1)$, $\alpha \in \{1, 2, 3\}$.

[9] Gallay, O. and Hongler, M.-O.: *Cooperative Dynamics of Loyal Customers in Queueing Networks*. Journal of Systems Science and Systems Engineering, 17/2 (2008), 241-254.

[10] Gallay, O. and Hongler, M.-O.: *Weariness and Loyalty Loss in Recurrent Service Models*. In: Proceedings of MOSIM'08, Paris, 2008, 2, 1011-1018.

[11] Gallay, O. and Hongler, M.-O.: *Circulation of Autonomous Agents in Production and Service Networks*. International Journal of Production Economics, in press.

[12] Hashem, E.: *Analysis of Random Drop for Gateway Congestion Control*. Report TR-465, MIT Laboratory of Computer Science, Boston, 1989.

[13] Hülsmann, M., Grapp, J. and Li, Y.: *Strategic Adaptivity in Global Supply Chains - Competitive Advantage by Autonomous Cooperation*. International Journal of Production Economics, 114 (2008), 14-26.

[14] Klein, M., Metzler, R. and Bar-Yam, Y.: *Handling Emergent Resource Use Oscillations*. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 35/3 (2005), 327-336.

[15] Lämmer, S. and Helbing, D.: *Self-Control of Traffic Lights and Vehicle Flows in Urban Road Networks*. Journal of Statistical Mechanics: Theory and Experiment, 4 (2008), art. no. P04019.

[16] Lee, L. S., Fiedler, K. D. and Smith, J. S.: *Radio Frequency Identification (RFID) Implementation in the Service Sector: A Customer-Facing Diffusion Model*. International Journal of Production Economics, 112 (2008), 587-600.

[17] Pingali, S., Tipper, D. and Hammond, J.: *The Performance of Adaptive Window Flow Controls, in a Dynamic Load Environment*. In: Proceedings of IEEE INFOCOM '90, 1990, 55-62.

[18] Kumara, S. R. T., Ranjan, P., Surana, A. and Narayanan, V.: *Decision Making in Logistics: A Chaos Theory Based Analysis*. Annals of the CIRP, 52/1 (2003), 381-384.

[19] Surana, A., Kumara, S. R. T., Greaves, M. and Raghavan U. N.: *Supply-Chain Networks: A Complex Adaptive Systems Perspective*. International Journal of Production Research, 43/20 (2005), 4235-4265.

[20] Tzeng, S.-F., Chen, W.-H. and Pai, F.-Y.: *Evaluating the Business Value of RFID: Evidence Form Five Case Studies*. International Journal of Production Economics, 112 (2008), 601-613.

[21] Wycisk, C., McKelvey, B. and Hülsmann, M.: *"Smart Parts" Supply Networks as Complex Adaptive Systems: Analysis and Implications*. International Journal of Physical Distribution & Logistics Management, 38/2 (2008), 108-125.