Queueing Syst (2012) 71:321–345
DOI 10.1007/s11134-012-9315-9

Load balancing via random local search in closed and open systems

Ayalvadi Ganesh · Sarah Lilienthal · D. Manjunath · Alexandre Proutiere · Florian Simatos

Received: 12 September 2011 / Revised: 16 January 2012 / Published online: 31 May 2012

Abstract In this paper, we analyze the performance of random *load resampling and migration* strategies in parallel server systems. Clients initially attach themselves to an arbitrary server, but may switch servers independently at random instants of time in an attempt to improve their service rate. This approach to load balancing contrasts with traditional approaches where clients make smart server selections upon arrival (e.g., Join-the-Shortest-Queue policy and variants thereof). Load resampling is particularly relevant in scenarios where clients cannot predict the load of a server before being actually attached to it. An important example is in wireless spectrum sharing where clients try to share a set of frequency bands in a distributed manner.

We first analyze the natural *Random Local Search (RLS)* strategy. Under this strategy, after sampling a new server randomly, clients only switch to it if their service rate is improved. In closed systems, where the client population is fixed, we derive tight estimates of the time it takes under RLS strategy to balance the load across

A. Ganesh (✉)
Dept. of Mathematics, University of Bristol, Bristol, UK
e-mail: a.ganesh@bristol.ac.uk

S. Lilienthal
Stats Lab., Cambridge University, Cambridge, UK
e-mail: s.lilienthal@statslab.cam.ac.uk

D. Manjunath
Dept. of Electrical Engineering, IIT Bombay, Mumbai, India
e-mail: dmanju@ee.iitb.ac.in

A. Proutiere
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: alepro@kth.se

F. Simatos
INRIA, Rocquencourt, France
e-mail: florian.simatos@m4x.org

servers. We then study open systems where clients arrive according to a random process and leave the system upon service completion. In this scenario, we analyze how client migrations within the system interact with the system dynamics induced by client arrivals and departures. We compare the load-aware RLS strategy to a load-oblivious strategy in which clients just randomly switch server without accounting for the server loads. Surprisingly, we show that both load-oblivious and load-aware strategies stabilize the system whenever this is at all possible. We use large-system asymptotics to characterize system performance, and augment this with simulations, which suggest that the average client sojourn time under the load-oblivious strategy is not considerably reduced when clients apply smarter load-aware strategies.

Keywords Mean field asymptotics · Stability analysis

Mathematics Subject Classification 60K25 · 68M14 · 68M20

1 Introduction

Load balancing is a key component of today's communication networks and computer systems in which resources are distributed over a wide area or across a large number of systems and have to be shared by a large number of users. Load balancing enables efficient resource utilization and thereby tends to improve the quality of service perceived by users. Traditionally, load balancing has been achieved by applying smart routing policies: when a new demand arrives, it is routed towards a particular resource depending on the current loads of the various resources; see [14] and references therein. In contrast, we are interested in systems where a new task is initially assigned to a resource chosen at random irrespective of the current resource loads, but where tasks can be re-assigned, i.e., migrate from one resource to another.

Our primary motivation stems from the increasing popularity of Dynamic Spectrum Access (DSA) techniques [1] as a potential mechanism for broadband access in future wireless systems. A common implementation platform for DSA is the use of reprogrammable Software-Defined-Radios (SDRs). These new radios are *frequency-agile* or *flexible*, and have the ability to rapidly jump from one frequency band to another in order to explore and exploit large parts of the spectrum. A central question in DSA is how multiple users may fairly and efficiently share spectrum in a distributed manner. Typically, the service rate of a user on a given frequency band is inversely proportional to the number of users transmitting on this band, i.e., to the load of the band. As new users entering the system have no way of determining the load on each frequency band, they have to initially select a band randomly. Should a user receive a quite poor quality of service on a given band, she may resample a new band at random and decide to switch to it. The overall system performance then strongly depends on the distributed resampling and switching strategy implemented by each user.

Though our primary motivation is DSA, our methods and results could provide insight into a number of other applications. One such pertains to wireline networks,

where there has recently been interest in multipath routing [15]. Here, users may use several path to download files, and have to select the appropriate path or the set of paths. Another application is in transport networks, where one might wish to understand how Wardrop equilibria, which correspond to the equalization of journey times across alternative routes, are achieved or approximated by network users acting on limited information. Our results could also shed insight on how quickly such equilibria can be re-established following major disruptions or other changes to the network. Finally, note that distributed load resampling can also be thought of as a game between selfish users. In fact, it is an instance of a congestion game (see, for example, [18]), and our results shed light on the time to reach a Nash equilibrium in such a game, but it also helps understanding the outcome of the game with a dynamic population of players.

We consider a generic system consisting of multiple servers (in DSA, frequency bands) employing the Processor Sharing (PS) service discipline, shared by clients who have to initially pick a server at random, and may later resample servers and migrate during their service. We restrict our attention to two natural distributed resampling and migration strategies, the *Random Local Search (RLS)* and *Random Load-Oblivious (RLO)* strategies. When implementing the RLS algorithm, a user resamples a new randomly chosen server at the instants of a Poisson process, and migrates to this new server if its load is smaller than that of the initial server. In contrast, under the RLO algorithm, a user hops between servers according to a random Markovian jump process irrespective of the loads of the visited servers.

We investigate both *closed* systems with fixed population of clients, and *open* systems with a population whose dynamics are governed by client arrivals and the completions of their services. In closed systems, we are interested in characterizing the time that it takes under the RLS algorithm to balance all server loads (note that here the RLO algorithm does not balance loads except in an average sense—so we do not study this algorithm in closed systems). In open systems, users arrive at the various servers according to independent stochastic processes of fixed intensities, and leave upon service completion. In this scenario, client migrations within the system interact in a complicated manner with the system dynamics induced by client arrivals and departures. We aim at characterizing system stability under the RLS and RLO strategies, as well as at deriving estimates of user sojourn times. Our contributions are as follows:

- *Closed systems.* We show that, starting from an arbitrary allocation of users to servers, the time τ it takes to achieve perfect balance of server loads (the system reaches perfect balance if the number of users at any pair of servers differs by at most 1) scales at most as $\log(m)(\frac{m^2}{n} + \log(m))$, where m and n denote the number of servers and users, respectively. This is a considerable improvement over the existing bounds that stated that τ scales at most as m^2 (see, for example, [12]). We also investigate the time τ_ϵ to reach an approximate ϵ-balance (a system reaches an approximate ϵ-balance if there exists p such that the number of users associated to any server lies between $(1-\epsilon)p$ and $(1+\epsilon)p$). We show that τ_ϵ scales at most as $\log(m)/\epsilon$.
- *Open systems.* We demonstrate that both RLS and RLO strategies achieve the largest stability region possible, i.e., that the system is stable under these two algo-

rithms provided that $\sum_{i=1}^{m} \lambda_i < \sum_{i=1}^{m} \mu_i$, where λ_i denotes the initial user arrival rate at server i and μ_i is the service rate of this server. The result is not surprising for RLS, but less intuitive for RLO since, under this algorithm, users take no account of server loads when migrating. For both RLS and RLO strategies, we derive approximate estimates of the average user sojourn time using large-system asymptotics. The estimates are shown to be exact when the number of servers grows large, but turn out to be quite accurate for systems of limited sizes as well. Our first numerical results suggest that again, surprisingly, the average client sojourn time under the load-oblivious RLO strategy is not considerably reduced when clients apply smarter load-aware RLS strategy. To our knowledge, this paper is the first to analyze the performance of RLS and RLO algorithms in open systems.

The paper is organized as follows. We describe our model and notation in Sect. 2, and related work in Sect. 3. Sections 4 and 5 are devoted to the analysis of closed and open systems, respectively. We provide concluding remarks in Sect. 6.

2 Model description and notation

We consider a set of m Processor Sharing servers of respective capacities $\mu_1, \ldots, \mu_m$. The system is *homogeneous* if $\mu_i = 1$ for all $i = 1, \ldots, m$. The system state at time t is represented by the number of clients associated to each server, $N(t) = (N_1(t), \ldots, N_m(t))$. The service rate of a client associated to server i at time t is then $\mu_i / N_i(t)$. Clients independently resample and switch servers to selfishly improve their service rate. They have a myopic view of the system in the sense that they are aware of their current service rates, but do not know the service rate they would achieve at other servers. Given this myopic view, it is natural to consider and analyze the two following random distributed resampling and migration algorithms:

- *Random Local Search (RLS) algorithm.* At the instants of a Poisson process of intensity $\beta > 0$, a client picks a new server uniformly at random and migrates to it if and only if this would increase her service rate. In other words, if at time t, a client associated to server i picks server j, she migrates to j if and only if $\mu_j / (N_j(t) + 1) > \mu_i / N_i(t)$.
- *Random Load-Oblivious (RLO) algorithm.* After arriving in the system, each client visits successive servers according to a continuous-time random walk with transition matrix $Q = \{q_{ij}, i, j = 1, \ldots, m\}$. The random walks are independent across clients, and irreducible. We denote by π the stationary distribution of this random walk. Note that as a consequence of irreducibility, clients visit all servers eventually, i.e., $\pi_i > 0$ for all $i = 1, \ldots, m$.

Note that under the RLO algorithm, clients do not take loads into account when switching servers. In particular, they may move to a server with a higher load. An example of such a resampling strategy is as follows. Each client has a Poisson clock of rate $\beta > 0$ and, when her clock ticks, she picks a new server uniformly at random and moves there irrespective of its load.

We analyze the performance of distributed resampling and migration strategies in closed and open systems. In closed systems, the total population of clients is fixed, equal to n. There are no arrivals or services. Such systems can be thought of as modelling persistent clients, who remain in the system forever (relative to some appropriate time-scale), and derive utility which is greater if their 'share' of a server (the reciprocal of the total number of clients at that server) is greater. For such systems, we investigate the time it takes under the RLS algorithm to balance clients across servers, starting from any arbitrary system state. In open systems, exogenous clients associate to server i according to a Poisson process of intensity λ_i (the arrival processes are independent across servers). Client service requirements are i.i.d. exponentially distributed with unit mean. Under RLS and RLO algorithms, $(N(t), t \geq 0)$ is a Markov process. In open systems, we are interested in characterizing the *stability region* of RLS and RLO strategies, defined as the set of arrival rates $\lambda = (\lambda_1, \ldots, \lambda_m)$ such that the system is stable, i.e., such that $(N(t), t \geq 0)$ is positive recurrent. We also aim at estimating the average client sojourn time.

3 Related work

There have been many studies on distributed, selfish load balancing algorithms and routing games in closed systems; e.g., [16] and references therein. Refer to [18] for a quite exhaustive survey. Much of the work in this area has concentrated on finding the fastest sequence of moves that would balance the system, also called Nashification [11]. One class of algorithms is the elementary step system, first described in [19] in which a sequence of best response moves are performed by the clients. Of course, this requires that the clients know the status of all the other servers. In [4, 5], the authors study closed systems with limited information about the servers' status. They consider a synchronous system where at each step, each server samples a new server randomly and if the load of the sampled server is smaller, then a client moves with probability $(N_c - N_n)/N_c$, where N_c is the load on the current server and N_n is the load of the sampled server. It is shown that the expected time to balance the system is $O(\log\log m + n^4)$. A modification of this load balancing algorithm is studied in [5], and it is shown that the expected time to balance the system is $O(\log m + n \log n)$. In [12], the author considers clients dynamics identical to those considered in this paper and uses the potential function introduced in [10] to quantify the time to achieve system balance. It is shown that the expected time to reach a balance scales at most as $O(m^2)$. We provide significant improvements on this bound.

In open systems, the client moves interact in a complicated manner with the client arrival and departure processes. There is very little work trying to understand this interaction. None of the existing work deals with a system similar to that studied here. For instance, [2] analyzes the interaction in a game-theoretical framework, where arrivals are *adversarial*, and where a central controller moves clients with the aim of stabilizing the system. The performance of the classical *work stealing* load-balancing scheme has also been studied; see, for example, [3] and references therein. Of course, there is an abundant literature on the performance of classical load-balancing schemes in open systems where clients are assigned to a given server for the entire duration

of their service; see, for example, the analysis of the supermarket model in [13, 17]. To our knowledge, the present paper provides the first analysis of natural distributed resampling and migration strategies in open systems.

4 Closed systems

In this section, we analyze the performance of the RLS resampling strategy in a closed homogeneous system, and obtain tight bounds on the expected time to balance the server loads.

Recall that there are n clients distributed among m servers. Let $n = qm + r$, $0 \le r \le m-1$. We now define the following:

- The state $N(t) = (N_1(t), \ldots, N_m(t))$ is *balanced* if $|N_i(t) - N_j(t)| \le 1$ for $1 \le i < j \le m$. The time to balance, τ, is defined as
$$\tau := \inf\bigl\{t > 0 : N(t) \text{ is balanced}\bigr\}.$$
- The state $N(t)$ is ϵ-*balanced* if $(1-\epsilon)p \le N_i(t) \le (1+\epsilon)p$ for all $i = 1, \ldots, m$, where $p = n/m$. The time, τ_ϵ, to ϵ-balance is defined as
$$\tau_\epsilon := \inf\bigl\{t > 0 : N(t) \text{ is } \epsilon\text{-balanced}\bigr\}.$$

Let $f, g : \mathbb{N} \to \mathbb{R}_+$. We say $f(k) = O(g(k))$ if there exists $c \in \mathbb{R}_+$ such that $f(k) \le cg(k)$ for all k. Similarly, for $f, g : \mathbb{N}^2 \to \mathbb{R}_+$, we say $f(k,l) = O(g(k,l))$ if there exists $c \in \mathbb{R}_+$ such that $f(k,l) \le cg(k,l)$ for all k and l.

4.1 Time to reach balance

We now characterize the time required by the RLS algorithm to reach perfect balance and ϵ-balance. Throughout, we shall consider a sequence of systems indexed by (m, n), the number of servers and clients, respectively. Our results apply if either of these tend to infinity along the sequence, but in practice we shall be interested in the case when both are large, which is captured by an asymptotic regime in which both quantities tend to infinity.

In practice, interest focuses on the case where $n \ge m$, i.e., there are more clients than servers and balancing the load is important because servers are a valuable resource. In the lightly loaded case of there being more servers than clients ($m > n$), load balancing may not be as important. Nevertheless, for completeness, we also treat this case.

Suppose first that $m \ge n$ along the sequence. Then, in a balanced configuration, every server will have 0 or 1 clients at it. (In this case, ϵ-balance can be more demanding than perfect balance, and hence impossible! So we only consider perfect balance.) The problem of reaching such a configuration from an arbitrary initial configuration is related to the coupon collector problem, which has been extensively studied.

Lemma 1 *Suppose that the number of servers, m, is no smaller than the number of clients, n. Then, the expected time, $\mathbb{E}[\tau]$, for randomized local search to achieve balance is $O(\frac{m}{m-n}\log n)$.*

In the remainder of this section, we shall focus on the more practically relevant case, where there are more clients than servers.

Theorem 1 *Consider a sequence of systems indexed by* (m,n), *and suppose that* $n > m$ *along this sequence. The expected time,* $\mathbb{E}[\tau]$, *for randomized local search to achieve balance is* $O(\log(m)(\frac{m^2}{n} + \log(m)))$.

Theorem 2 *Consider a sequence of systems indexed by* (m,n), *and suppose that* $n > (1+\alpha)m$ *along this sequence, for some constant* $\alpha > 0$. *The expected time,* $\mathbb{E}[\tau_\epsilon]$ *for randomized local search to achieve* ϵ*-balance is* $O(\log(m)/\epsilon)$.

Remarks

1. It is easy to see, applying Markov's inequality, that the same upper bounds on the time to balance hold in probability as in expectation.
2. We now compare our bounds on τ with that from [12]. From Theorem 2.7 of [12], the expected number of attempted moves before reaching balance is $O(m^2 n)$. Since move attempts (resampling) occur at rate n, this gives us a time complexity of $O(m^2)$. Our bound in Theorem 1 is much tighter.
3. Our bound is close to the best possible. To see this, suppose m divides n exactly. At some stage, the algorithm will reach an allocation in which one server has $n/m+1$ clients, one other server has $n/m-1$ clients and all others have exactly n/m clients. Each of the $n/m+1$ clients at the overloaded server attempts to move at rate 1, and each move attempt is successful with probability $1/m$. Hence, the mean time for just the final move is $m^2/(m+n) \geq m^2/(2n)$. Our bound is only a $\log m$ factor higher than the time for the last move.

   Alternatively, consider the situation when $m^2 = o(n)$ and all n clients are initially at the same server. Then, at least $n - \lceil n/m \rceil$ clients need to move out of this server to reach balance. When there are k clients at the server, the expected time to the next move is at least $1/k$ (possibly more, as the move attempt may not be successful). Hence, the expected time to reach balance is at least

$$\sum_{k=\lceil n/m \rceil + 1}^{n} \frac{1}{k} \geq \int_{n/m}^{n} \frac{1}{x}\,dx = \log m.$$

   Again our bound is only a $\log m$ factor higher than the above lower bound on the time to reach balance.

4.2 Proofs

Without loss of generality, we take the rate β of the independent Poisson clocks at each client to be unity. A client at server i whose clock has ticked at time t attempts to move by sampling a server uniformly at random from all m servers. It moves to the sampled server, say j, if and only if $N_i(t) - N_j(t) > 1$. Clearly, $N(t)$ evolves as a continuous time Markov chain.

4.2.1 *Proof of Lemma 1*

Recall that every server has 0 or 1 clients in a balanced configuration. Assume we don't start with a balanced configuration, and consider some time $t < \tau$, when the configuration is still unbalanced. Let $A_0(t)$ denote the number of empty servers, and $A_i(t)$, $i \geq 1$ the number of servers with exactly i clients. Then, $A_0(t) \geq m - n + 1$, and

$$\tau = \inf\bigl\{t \geq 0 : A_0(t) = m - n\bigr\}.$$

Noting that $A_0(t) + \sum_{i=1}^{\infty} A_i(t) = m$, while $\sum_{i=1}^{\infty} i A_i(t) = n$, we conclude that the number of clients at servers with two or more clients is given by

$$\sum_{i=2}^{\infty} i A_i(t) = n - A_1(t) \geq n - m + A_0(t). \tag{1}$$

Each of these clients attempts to move at unit rate, independent of other clients. If it chooses an empty server to move to, then the number of empty servers decreases by 1.

Define $T_k = \inf\{t \geq 0 : A_0(t) \leq k\}$ to be the first time that the number of empty servers is no more than k. When the number of empty servers hits $m - n$, the system is balanced; in other words, $\tau = T_{m-n}$.

The random time, $T_{k-1} - T_k$, for $A_0(t)$ to decrease from k to $k-1$, is stochastically dominated by an exponential random variable denoting the time for one of the clients in servers containing two or more clients to move to an empty server. By (1), the number of clients at such servers is at least $n - m + k$ at time T_k. Each of these moves at rate 1, and chooses an empty server to move to with probability k/m. Hence,

$$T_{k-1} - T_k \preceq \mathrm{Exp}\left(\frac{k(n-m+k)}{m}\right),$$

where we write $X \preceq Y$ to mean that X is stochastically dominated by Y. We shall also write $\mathrm{Exp}(\lambda)$ to denote an exponential random variable with parameter λ and mean $1/\lambda$. It follows that

$$\mathbb{E}[T_{k-1} - T_k] \leq \frac{m}{k(n-m+k)} = \frac{m}{m-n}\left(\frac{1}{k+n-m} - \frac{1}{k}\right). \tag{2}$$

We also note that $A_0(t)$ is monotone decreasing in t, and that $A_0(0) \leq m - 1$. Hence,

$$\begin{aligned}
\mathbb{E}[\tau] = \mathbb{E}[T_{m-n}] &\leq \sum_{k=m-n+1}^{m-1} \mathbb{E}[T_{k-1} - T_k] \\
&\leq \frac{m}{m-n} \sum_{k=m-n+1}^{m-1} \left(\frac{1}{k+n-m} - \frac{1}{k}\right) \\
&\leq \frac{m}{m-n}\left(1 + \int_{m-n+1}^{m-1} \frac{1}{x+n-m}\, dx\right) = \frac{m}{m-n}\bigl(1 + \log(n-1)\bigr).
\end{aligned}$$

This completes the proof of the lemma.

4.2.2 *Proof of Theorem 1*

Define $V(t) := \max_{1\le j\le m} N_j(t)$, i.e., $V(t)$ is the maximum number of clients associated with any server at time t. Define $C_v(t)$ to be the number of servers with exactly v clients, $B_v(t)$ to be the number with exactly $v-1$ clients and $A_v(t)$ to be the number with strictly less than $v-1$ clients, all at time t.

The idea of the proof is as follows. The evolution of $N(t)$ towards balance is divided into phases. If $V(t)=v$, then $N(t)$ is said to be in phase v. Thus, $C_v(t)$ is the number of maximally loaded servers in phase v. Since a client never moves to a server that has more clients than its current server, $V(t)$ is monotone decreasing and, in each phase, $C_v(t)$ is also monotone decreasing. Phase v ends when $C_v(t)=0$. Let τ_v denote the (random) length of phase v. Each phase can be further divided into sub-phases, say (v,c), when $C_v(t)=c$. Let $\tau_{v,c}$ denote the random length of time that it takes for $C_v(t)$ to decrease from c to $c-1$. Observe that $\tau_v=\sum_c \tau_{v,c}$ and $\tau=\sum_v \tau_v$. When $N(t)$ is balanced, $V(t)=\lceil \frac{n}{m}\rceil$, $C_{\lceil \frac{n}{m}\rceil}(t)=r$ if $r>0$ and $C_{\lceil \frac{n}{m}\rceil}(t)=m$ otherwise. This gives us the maximum range for v. The number of sub-phases in each phase is also similarly bounded. The theorem is proved by bounding the expected times of each of the sub-phases and phases.

Proof In phase v, observe that

$$vC_v(t)+(v-1)B_v(t)\le n,$$
$$m-B_v(t)-C_v(t)=A_v(t).$$

Furthermore, if $N(t)$ is not balanced, then there has to be at least one server with $v-2$ or fewer clients (i.e., $A_v(t)\ge 1$). Hence, while the system is unbalanced, we have

$$A_v(t)\ge \begin{cases} m-\frac{n}{v-1}, & v\ge \frac{n}{m-1}+1,\\ 1, & v\le \frac{n}{m-1}. \end{cases} \tag{3}$$

Each of the $vC_v(t)$ clients at one of the maximally loaded servers samples one of the m servers at random at unit rate. If the sampled server happens to be one of the $A_v(t)$ servers with $v-2$ or fewer clients, then the client moves to the sampled server and $C_v(t)$ decreases by 1. This event has probability $A_v(t)/m$. Hence, $C_v(t)$ decreases by 1 at rate $vC_v(t)A_v(t)/m$, and we obtain from (3) that

$$\tau_{v,c}\preceq \tilde{\tau}_{v,c}\sim \mathrm{Exp}(\lambda_{v,c}),\quad \text{where } \lambda_{v,c}=\begin{cases} vc\big(1-\frac{n}{m(v-1)}\big), & v\ge \frac{n}{m-1}+1,\\ \frac{vc}{m}, & v\le \frac{n}{m-1}. \end{cases} \tag{4}$$

Here we write $X\sim Y$ to mean that the random variables X and Y have the same distribution. In particular,

$$\mathbb{E}[\tau_{v,c}]\le \mathbb{E}[\tilde{\tau}_{v,c}]=\begin{cases} \frac{1}{vc}\frac{m(v-1)}{[m(v-1)-n]}, & v\ge \frac{n}{m-1}+1,\\ \frac{m}{vc}, & v\le \frac{n}{m-1}. \end{cases} \tag{5}$$

At any time t, $C_v(t)$ is bounded above by $\lfloor n/v \rfloor$, since there cannot be more than this many servers with v clients. Since phase v ends when $C_v = 0$, we have $\mathbb{E}[\tau_v] \le \sum_{c=1}^{\lfloor n/v \rfloor} E[\tau_{v,c}]$, and we obtain using (5) that

$$\mathbb{E}[\tau_v] \le \begin{cases} \frac{m(v-1)}{v[m(v-1)-n]} \sum_{c=1}^{\lfloor n/v \rfloor} \frac{1}{c}, & v \ge \frac{n}{m-1} + 1, \\ \frac{m}{v} \sum_{c=1}^{\lfloor n/v \rfloor} \frac{1}{c}, & v \le \frac{n}{m-1}. \end{cases} \tag{6}$$

Moreover, for all $v \ge \lceil \frac{n}{m} \rceil$, we readily see that,

$$\begin{aligned} \sum_{c=1}^{\lfloor n/v \rfloor} \frac{1}{c} &\le 1 + \int_1^{n/v} \frac{1}{x}\, dx \\ &\le 1 + \log \frac{n}{v} \le 1 + \log m. \end{aligned}$$

Hence, we can rewrite (6) as

$$\frac{\mathbb{E}[\tau_v]}{1 + \log m} \le \begin{cases} \frac{m(v-1)}{v[m(v-1)-n]}, & v \ge \frac{n}{m-1} + 1, \\ \frac{m}{v}, & v \le \frac{n}{m-1}. \end{cases} \tag{7}$$

Finally, the time τ it takes to reach perfect balance satisfies

$$\tau \le \sum_{v=\lceil n/m \rceil + 1}^{n} \tau_v + \sum_{c=r+1}^{m} \tau_{\lceil n/m \rceil, c}. \tag{8}$$

Now, we have by (5) that $\mathbb{E}[\tau_{\lceil n/m \rceil, c}] \le \frac{m}{\lceil n/m \rceil c}$, and so,

$$\begin{aligned} \sum_{c=r+1}^{m} \mathbb{E}[\tau_{\lceil n/m \rceil, c}] &\le \frac{m}{\lceil n/m \rceil} \sum_{c=r+1}^{m} \frac{1}{c} \\ &\le \frac{m^2}{n} \left(1 + \int_1^m \frac{1}{x}\, dx \right) \\ &= \frac{m^2}{n} (1 + \log m). \end{aligned} \tag{9}$$

Hence, from (7), (8), and (9), we obtain

$$\begin{aligned} \frac{\mathbb{E}[\tau]}{1 + \log m} \le \frac{m^2}{n} &+ \sum_{v=\lceil n/m \rceil + 1}^{\lceil n/(m-1) \rceil} \frac{m}{v} \\ &+ \sum_{v=\lceil n/(m-1) \rceil + 1}^{n} \frac{m}{m(v-1) - n}. \end{aligned} \tag{10}$$

The number of terms in the first sum above is at most $\max\{1, \frac{n}{m(m-1)}\}$. Each summand is no more than m^2/n. Hence, the first sum is bounded above by $\max\{\frac{m^2}{n}, 2\}$. The

second sum is bounded above by

$$\frac{m}{\frac{mn}{m-1}-n}+\int_{\frac{n}{m-1}+1}^{n}\frac{m}{m(x-1)-n}\,dx=\frac{m(m-1)}{n}+\log\frac{m(n-1)-n}{\frac{mn}{m-1}-n}.$$

Substituting these expressions in (10) and simplifying, we get

$$\begin{aligned}\mathbb{E}[\tau]&\le(1+\log m)\left(\max\left\{\frac{m^2}{n},2\right\}+\frac{m^2}{n}+\log\bigl(m^2\bigr)+\frac{m^2}{n}\right)\\&\le 3(1+\log m)\left(\frac{m^2}{n}+\log m+1\right).\end{aligned}$$

This completes the proof. □

4.2.3 Proof of Theorem 2

Recall that $p=n/m$. We need the following definitions.

- Server i is ϵ-balanced at time t if $(1-\epsilon)p\le N_i(t)\le(1+\epsilon)p$, underloaded if $N_i(t)<(1-\epsilon)p$ and overloaded if $N_i(t)>(1+\epsilon)p$. $M_C(t)$, $M_U(t)$, and $M_O(t)$ denote the number of ϵ-balanced, underloaded, and overloaded servers, respectively.
- The underflow from server i is defined to be

$$u_i(t)=\begin{cases}0 & \text{if } N_i(t)\ge p,\\ p-N_i(t) & \text{otherwise.}\end{cases}$$

Also, let $U(t):=\sum_{i=1}^{m}u_i(t)$. Similarly, define the overflow from server i as

$$o_i(t)=\begin{cases}0 & \text{if } N_i(t)\le p,\\ N_i(t)-p & \text{otherwise,}\end{cases}$$

and $O(t):=\sum_{i=1}^{m}o_i(t)$.

Proof Let $N_O(t)$ be the number of 'overflowing' clients defined as

$$N_O(t):=\sum_{i\in\mathcal{M}_O(t)}\bigl(N_i(t)-(1+\epsilon)p\bigr),$$

where $\mathcal{M}_O(t)$ is the set of overloaded servers at time t. We can write

$$\begin{aligned}U(t)&\le\bigl(M_U(t)\times p\bigr)+M_C(t)\times(\epsilon p),\\O(t)&\ge N_O(t)+\bigl(m-M_U(t)-M_c(t)\bigr)\times(\epsilon p).\end{aligned}$$

Since $O(t)=U(t)$, we obtain

$$pM_U(t)+(\epsilon p)M_C(t)\ge N_O(t)+\bigl(m-M_U(t)-M_C(t)\bigr)(\epsilon p),$$

which yields

$$\begin{aligned} N_O(t) &\le p\big(M_U(t)+M_C(t)\big)\big((1+\epsilon)-\epsilon m\big) \\ M_C(t)+M_U(t) &\ge \frac{N_O(t)}{(1+\epsilon)p}+\frac{\epsilon}{1+\epsilon}m \\ &\ge \max\left\{\frac{N_O(t)}{(1+\epsilon)p},\frac{\epsilon}{1+\epsilon}m\right\}. \end{aligned}$$

Now consider a client that is attempting to move at time t. We say that this attempt results in a good move if the attempt results in a migration that reduces $N_O(t)$. Let G denote the event corresponding to a good move. When the state of the system is (N_O, M_C, M_U), the probability of a good move is

$$\mathbb{P}(G) \ge \frac{N_O+(1+\epsilon)p}{n}\frac{M_C+M_U}{m},$$

and the number of attempts between successive good moves is geometric with mean at most $\frac{mn}{(N_O+p)M_U}$.

Let K_G denote the number of attempts before a good move occurs from the state (N_O, M_C, M_U). The expected number of attempts before a good move reduces N_O satisfies

$$\mathbb{E}[K_G] \le \frac{mn}{(N_O+(1+\epsilon)p)(M_C+M_U)}.$$

Let K_ϵ denote the number of attempts to achieve ϵ-balance. In the worst case, $N_O(t)$ starts at $(m-1)p$ and ends at 1. We can then bound $\mathbb{E}[K_\epsilon]$ as follows:

$$\begin{aligned} \mathbb{E}[K_\epsilon] &\le \sum_{i=1}^{(m-1)p}\frac{mn}{((1+\epsilon)p+i)\big(\max\{\frac{i}{(1+\epsilon)p},\frac{\epsilon}{1+\epsilon}m\}\big)} \\ &= \sum_{i=1}^{\epsilon n}\frac{mn}{((1+\epsilon)p+i)\big(\frac{\epsilon}{1+\epsilon}m\big)} \\ &\quad+\sum_{i=\epsilon n+1}^{(m-1)p}\frac{mn}{((1+\epsilon)p+i)\big(\frac{i}{(1+\epsilon)p}\big)} \\ &= \frac{1+\epsilon}{\epsilon}n\sum_{i=1}^{\epsilon n}\frac{1}{((1+\epsilon)p+i)} \\ &\quad+mn\sum_{i=\epsilon n+1}^{(m-1)p}\frac{1}{i}-\frac{1}{((1+\epsilon)p+i)} \\ &\le \frac{1+\epsilon}{\epsilon}n\log\left(\frac{(1+\epsilon)p+\epsilon n}{(1+\epsilon)p}\right) \\ &\quad+mn\log\left(\frac{(m-1)p}{\epsilon n}\frac{(1+\epsilon)p+\epsilon n}{(1+\epsilon)p+(m-1)p}\right) \end{aligned}$$

$$\leq \frac{n}{\epsilon}\log(1+\epsilon m) + mn\left(-\frac{1}{m} + \frac{1+\epsilon}{\epsilon m} - \frac{\epsilon}{m}\right)$$
$$\leq \frac{n}{\epsilon}\log(m).$$

Since each client is sampling at unit rate, the total sampling rate is n and the average time to reach ϵ-balance, $\mathbb{E}[\tau_\epsilon]$ is $\mathbb{E}[K_\epsilon]/n$. Thus $\mathbb{E}[\tau_\epsilon] = O((\log m)/\epsilon)$. □

5 Open systems

In open systems, we are interested in quantifying classical queueing performance metrics, such as the stability region and the mean client sojourn time. We first investigate the stability region achieved under RLO and RLS algorithms. Both algorithms are shown to stabilize the system whenever this is at all possible, which for load-oblivious RLO algorithm may be surprising. Then, we try to obtain more detailed estimates of the system performance. As it turns out, the system equilibrium distribution is difficult, if not impossible, to derive, and we rely on large-system asymptotics to provide insights into the way the system behaves.

5.1 Stability

In the following, we denote $\lambda = (\lambda_1, \ldots, \lambda_m)$ and $\mu = (\mu_1, \ldots, \mu_m)$ the vectors representing the arrival and departure rates at the various servers. $\|\cdot\|$ denotes the L_1-norm on $\mathbb{R}^m$. We first provide an upper bound on the maximum stability region defined as the set of λ such that there may exist a resampling and migration strategy stabilizing the system. This set is obtained by assuming that all servers' resources are pooled.

Proposition 1 *Assume that λ is such that $\sum_i \lambda_i > \sum_i \mu_i$. Then there is no resampling and migration strategy stabilizing the system.*

Proof The proof is straightforward. Remark that for any resampling and migration strategy, the total service rate is less than $\sum_i \mu_i$. Then if $\sum_i \lambda_i > \sum_i \mu_i$, the average number of clients in the system grows at a rate greater than $\sum_i \lambda_i - \sum_i \mu_i > 0$. The system is then unstable. □

The two following theorems state that both RLO and RLS strategies achieve maximum stability.

Theorem 3 *Assume that $\sum_i \lambda_i < \sum_i \mu_i$. Then the system is stable under RLO algorithm.*

Theorem 4 *Assume that $\sum_i \lambda_i < \sum_i \mu_i$. Then the system is stable under RLS algorithm.*

A result somewhat similar to that of Theorem 3 was first stated in [7] using heuristic fluid limits arguments. Fluid limits are powerful techniques to study ergodicity

of Markov processes [8]. They comprise the study of the system behavior in the following limiting regime: the initial condition is scaled up by a multiplicative factor k, time is accelerated by the same factor, and k tends to ∞. Often the system becomes tractable in this regime and even deterministic. If the system in the fluid regime reaches 0 in a finite time, then the process is ergodic. In the fluid regime, clients stay for very long periods of time in our system, and since, under RLO algorithm, the client random walks are ergodic, the probability that a given client is associated to server i *should* be proportional to π_i (the equilibrium distribution of the random walk). In such case, when the client population is large (as in the fluid regime), all servers should be occupied and active, ensuring that the system empties in finite time. This is the argument used in [7], but not justified. The problem arises because the client migration process actually interacts with arrivals and departures. Handling this interaction turns out to be extremely difficult. Recently however, in [21], the authors were able to formally derive the system fluid limits, and analyze its stability under very specific assumptions on the client random walk (its transition matrix Q has to be diagonalizable). Their proof is quite intricate. In the following, we prove Theorem 3 without the use of fluid limits, and for *any* random walk. Our proof is much more direct than that in [21], and hence is amenable to deal with more general cases and possible extensions. For the proof of Theorem 4, we use a rather classical method, namely, we exhibit a simple Lyapunov function.

5.1.1 Proof of Theorem 3

Recall that by definition, under RLO strategy, the process $(N(t), t \geq 0)$ is the Markov process with the following non-zero transition rates for $1 \leq i \neq j \leq m$:

$$\begin{cases} \Omega(n, n + e_i) = \lambda_i, \\ \Omega(n, n - e_i + e_j) = n_i q_{ij}, \\ \Omega(n, n - e_i) = \mu_i \mathbb{1}_{\{n_i > 0\}}, \end{cases} \tag{11}$$

where $n = (n_1, \ldots, n_m) \in \mathbb{N}^m$ and e_i is the m-dimensional vector with every coordinate equal to 0, except for the ith one equal to 1. The matrix $Q = (q_{ij})$ describes the migration of clients, and it is only assumed to possess a unique stationary distribution $\pi = (\pi_i)$ such that $\pi_i > 0$ for each $i = 1, \ldots, m$. The aim of the analysis is to use the following result, known as Foster's criterion [20].

Foster's criterion *If there exist K and $t \geq 0$ such that*

$$\sup_{n \in \mathbb{N}^m : \|n\| \geq K} \mathbb{E}_n\big(\big\|N(t)\big\| - \|n\|\big) < 0, \tag{12}$$

where $\mathbb{E}_n(\cdot) = \mathbb{E}(\cdot | N(0) = n)$, then $(N(t), t \geq 0)$ is ergodic.

Kolmogorov's equation is the first step that leads to (12): for any $t \geq 0$, the drift $\mathbb{E}_n(\|N(t)\| - \|n\|)$ is given by

$$\mathbb{E}_n\big(\big\|N(t)\big\| - \|n\|\big) = \|\lambda\| t - \int_0^t \mathbb{E}_n\left(\sum_{i=1}^m \mu_i \mathbb{1}_{\{N_i(u) > 0\}}\right) du.$$

This gives the following inequality, which is the basis of our drift analysis:

$$\mathbb{E}_n\big(\big\|N(t)\big\| - \|n\|\big) \le \|\lambda\|t - \|\mu\| \int_0^t \mathbb{P}_n\big(N(u) > 0\big)\,du, \tag{13}$$

where $\mathbb{P}_n[\cdot] = \mathbb{P}[\cdot|N(0) = n]$, and for $x \in \mathbb{N}^n$, $x > 0$ is to be understood coordinate-wise, i.e., $x_i > 0$ for each $i = 1, \ldots, m$.

The idea of the proof of (12) is that when the system starts with many clients, then the number of arrivals and departures is negligible on the time interval $[0, t]$ and the system behaves like the closed one. For a closed system, it is not difficult to show, using the fact that Q has an invariant measure, that $\mathbb{P}(N(u) > 0)$ for $u > 0$ is arbitrarily close to 1 as the number of clients in the system increases. In view of (13), this gives a negative drift when $\sum_i \lambda_i < \sum_i \mu_i$.

The following coupling initially proposed and formally justified in [21] is key to relate the open and closed systems. For $n \in \mathbb{N}^m$ and $\ell, \rho \in \mathbb{R}_+^m$, denote by $N^n_{\ell,\rho}$ the process under RLO strategy starting in the initial state n, with arrival rate ℓ_i at server i with capacity ρ_i. Then $(N^n_{\ell,\rho}(t))$ is the Markov process with $N^n_{\ell,\rho}(0) = n$, and with non-zero transition rates given by (11) with ℓ_i instead of λ_i and ρ_i instead of μ_i. Then the processes $N^n_{\ell,0}$ and $N^0_{\rho,0}$ can be coupled in such a way that for some process $Z(t) \ge 0$,

$$N^n_{\ell,\rho}(t) = N^n_{\ell,0}(t) - N^0_{\rho,0}(t) + Z(t), \quad t \ge 0.$$

Moreover, the processes $\|N^0_{\rho,0}\|$ and $N^n_{\ell,0}$ are independent, and $\|N^0_{\rho,0}\|$ is a Poisson process with parameter $\|\rho\|$. Essentially, this coupling realizes the process $N^n_{\ell,\rho}$ with arrivals and departures as the difference between two processes without departures. This coupling can be constructed as follows: consider a particle system with three kinds of particles, colored blue, red, and green. All the particles in the system are performing independent continuous-time random walks, going from i to j at rate q_{ij}, and the system starts with only blue particles.

Consider two independent Poisson processes $\mathcal{N}_\ell$ and $\mathcal{N}_\rho$ with respective parameters $\|\ell\|$ and $\|\rho\|$: at times of $\mathcal{N}_\ell$, add a new blue particle at server i with probability $\ell_i/\|\ell\|$. At times of $\mathcal{N}_\rho$, consider server i with probability $\rho_i/\|\rho\|$: if there is a blue particle, choose one at random and turn it into a red one. If there is no blue particle, add a green particle.

If $B_i(t)$, $R_i(t)$, and $G_i(t)$ are respectively the number of blue, red, and green particles at server i at time t, then it is easy to see that:

- B is distributed like $N^n_{\ell,\rho}$,
- $B + R$ is distributed like $N^n_{\ell,0}$,
- $R + G$ is distributed like $N^0_{\rho,0}$ and $\|B + G\| = \mathcal{N}_\rho$ is independent of $B + R$.

This proves the coupling with $Z(t) = G(t)$. The process $N^n_{\ell,0}$ can be seen as the superposition of the initial particles with the particles arriving at rate $\|\ell\|$, hence the additional coupling $N^n_{\ell,0} = N^n_{0,0} + N^0_{\ell,0}$ holds, and finally, $N^n_{\ell,\rho}$ can be written

$$N^n_{\ell,\rho}(t) = N^n_{0,0}(t) + N^0_{\ell,0}(t) - N^0_{\rho,0}(t) + Z(t), \quad t \ge 0,$$

with $N^n_{0,0}$ and $\|N^0_{\rho,0}\|$ independent, and $Z(t) \geq 0$. Starting from (13), we now turn our attention to proving the existence of constants K and t which satisfy (12). We have, using the coupling's notation, $\mathbb{P}_n(N(u) > 0) = \mathbb{P}(N^n_{\lambda,\mu}(u) > 0)$ and hence, for any $0 \leq u \leq t$ and $n \in \mathbb{N}^m$,

$$\begin{aligned}\mathbb{P}_n\big(N(u) > 0\big) &= \mathbb{P}\big(N^n_{0,0}(u) + N^0_{\lambda,0}(u) + Z(u) > N^0_{\mu,0}(u)\big)\\ &\geq \mathbb{P}\big(N^n_{0,0}(u) > \big\|N^0_{\mu,0}(u)\big\|\big).\end{aligned}$$

Since the process $N^n_{0,0}$ is independent of the random variable $\|N^0_{\mu,0}(t)\|$, we can work conditionally on the value of $\|N^n_{\mu,0}(t)\|$ and study the quantity $\mathbb{P}(N^n_{0,0}(u) > M)$. Thus we only need to consider the closed process $N^n_{0,0}$ henceforth, and so we simplify the notation and note $N^n_{0,0} = N^n$. Markov's inequality gives

$$\begin{aligned}\mathbb{P}\big(\exists i \in \{1,\dots,m\} : N^n_i(u) \leq M\big) &= 1 - \mathbb{P}\big(N^n(u) > M\big)\\ &\leq \sum_{i=1}^m \mathbb{P}\big(N^n_i(u) \leq M\big) \leq e^M \sum_{i=1}^m \mathbb{E}\big(e^{-N^n_i(u)}\big).\end{aligned}$$

For any $i \in \{1,\dots,m\}$,

$$\mathbb{E}\big(e^{-N^n_i(u)}\big) = \prod_{j=1}^m \big[\mathbb{E}_j\big(e^{-\mathbb{1}_{\{\xi(u)=i\}}}\big)\big]^{n_j}$$

where ξ under $\mathbb{P}_j$ is a continuous-time Markov chain with transition rates $Q = (q_{ij})$, and which starts at $\xi(0) = j$. If $p(j,i,u) = \mathbb{P}_j(\xi(u) = i)$, one gets for $u \geq t_0 > 0$ and $n \in \mathbb{N}^m$ with $\|n\| \geq K$

$$\begin{aligned}\mathbb{E}\big(e^{-N^n_i(u)}\big) &= e^{\sum_{j=1}^m n_j \log(1-(1-1/e)p(j,i,u))}\\ &\leq e^{-\|n\|(1-1/e)p(t_0)} \leq e^{-K(1-1/e)p(t_0)}\end{aligned}$$

with $p(t_0) = \inf_{u \geq t_0} \min_{1 \leq i,j \leq m} p(j,i,u)$. Note that since, for any $1 \leq i, j \leq m$, $p(j,i,u) > 0$ for any $u > 0$ and $p(j,i,u) \to \pi_i > 0$ as $u \to +\infty$, one has that $p(t_0) > 0$. Therefore, for $u \geq t_0$ and n with $\|n\| \leq K$, integrating with respect to the law of $\|N^0_{\mu,0}(t)\|$ gives

$$\mathbb{P}\big(N^n(u) > \big\|N^0_{\mu,0}(t)\big\|\big) \geq 1 - \varepsilon(t,K,t_0)$$

with $\varepsilon(t,K,t_0) = me^{\|\mu\|t(e-1)-K(1-1/e)p(t_0)}$. In particular, for $t \geq t_0$,

$$\sup_{n \in \mathbb{N}^m : \|n\| \geq K} \mathbb{E}_n\big(\big\|N(t)\big\| - \|n\|\big) \leq \|\lambda\|t - \|\mu\|(t-t_0)\big(1 - \varepsilon(t,K,t_0)\big).$$

Since by assumption $\|\lambda\| < \|\mu\|$, it is not difficult to choose constants t, t_0, and K such that the right hand side is strictly negative (for instance, $t = 1$, t_0 small enough, and K large enough), which gives the result.

5.1.2 Proof of Theorem 4

Intuitively, it is clear that the load-dependent RLS strategy performs better than the load-oblivious RLO policy, since it seems harder under RLS to see an empty server. This simple observation shows that the number of empty servers should be part of a Lyapunov function, and indeed this leads us to define the function $f : \mathbb{N}^m \to \mathbb{R}_+$ by:

$$\forall n, \quad f(n) = \sum_{i=1}^{m} \max(\epsilon, n_i) = \|n\| + \varepsilon k_0(n)$$

with $k_0(n) = \mathbb{1}_{\{n_1=0\}} + \cdots + \mathbb{1}_{\{n_m=0\}}$ the number of empty servers in state n. In order for f to be a Lyapunov function, the constant $0 < \epsilon < 1$ has to satisfy

$$\epsilon \times \sum_i \mu_i < \sum_i (\mu_i - \lambda_i) - \gamma$$

for some $\gamma > 0$.

Let $K_0(n)$ (resp., $K_1(n)$) be the set of servers that are empty (resp., have a single client). Denote by $k_0(n)$ and $k_1(n)$ the respective cardinalities of these sets. Let us compute the average drift $\Delta f(n)$ of the Markov process $N(t)$ under RLS strategy. We have

$$\Delta f(n) = \sum_i \lambda_i - \sum_{i \notin K_0(n)} \mu_i + \epsilon \left(\sum_{i \in K_1(n)} \mu_i - \sum_{i \in K_0(n)} \lambda_i - Y(n) \right),$$

where $Y(n)$ is the rate in state n at which empty servers are fed by migrating clients.

– If $k_0(n) = 0$, there are no empty servers in state n and in particular $Y(n) = 0$. We have

$$\Delta f(n) = \sum_i (\lambda_i - \mu_i) + \epsilon \sum_{i \in K_1(n)} \mu_i < -\gamma$$

because of our choice of ϵ.

– If $k_0(n) > 0$, there is at least one empty server in state n. Define $p(n) = \max_i n_i$. Considering migrations of the $p(n)$ clients from (one of) the server(s) with maximum size to one of the empty servers, we obtain $Y(n) \geq \frac{\beta \times p(n)}{m}$, which ensures that $\Delta f(n) < -\gamma$ when $p(n)$ is large enough, say greater than K.

We conclude the proof by considering the drift outside the set $F = \{n : f(n) < m(K + \epsilon)\}$. First, remark that F is finite. Then, when $n \notin F$, $p(n) \geq K$. We deduce that for all $n \notin F$: $\Delta f(n) < -\gamma$. The positive recurrence follows.

5.2 Approximate performance estimates

The system behavior in stationary regime under RLO and RLS strategies is extremely difficult to analyze. For example, $(N(t), t \geq 0)$ is unfortunately not reversible under

these strategies. To obtain estimates of the steady state distribution and client sojourn times, we use large-system asymptotics, i.e., we let m grow large. Recently, large-system asymptotics have been successfully applied in many contexts in communication systems. They have been used, for example, to understand load balancing issues such as those arising in the supermarket model [13, 17, 23]. In the rest of the section, we denote by $N^{(m)}(t)$ the vector representing the numbers of clients at time t at each server in a system with m servers under either RLO or RLS algorithm.

In what follows, we consider homogeneous systems where $\lambda_i = \lambda$ and $\mu_i = 1$ for all i. This restriction simplifies the notation and results, but is not essential. We discuss at the end of this subsection how to deal with heterogenous systems. We also assume that the number of clients associated to a given server is bounded by a (possibly very large) constant B. Again this assumption is not crucial, but relaxing this assumption would require a much more involved analysis, as explained below.

5.2.1 RLO algorithm

We first consider RLO algorithms. We assume that a client jumps from one server to another at the instants of a Poisson process of intensity β, and that the next server is chosen uniformly at random. The analysis can be extended to any random walk (see Sect. 5.2.3). We represent the system state at time t by $X_k^{(m)}(t)$ the proportion of servers with exactly k clients at time t. We also define $S_k^{(m)}(t) = \sum_{l \geq k} X_l^{(m)}(t)$.

Let us compute the average change in the system state in a small interval of time of duration dt, and more specifically the change in $X_k^{(m)}$. Arrivals occur at rate λm: An arrival increases $X_k^{(m)}$ if it occurs at servers with $k-1$ clients, and decreases $X_k^{(m)}$ if it occurs at servers with k clients. Hence the change in $X_k^{(m)}$ due to exogenous arrivals is $dt\lambda(X_{k-1}^{(m)} - X_k^{(m)})$. Departures can be analyzed similarly. Let us now compute the change due to client migrations. Clients migrating to server with $k-1$ (resp., k) clients increase (resp., decrease) $X_k^{(m)}$. In addition, clients migrating from servers with k (resp., $k+1$) clients decrease (resp., increase) $X_k^{(m)}$. The average change in $X_k^{(m)}$ due to client migrations is thus $dt\beta((X_{k-1}^{(m)} - X_k^{(m)}) \sum_j j X_j^{(m)} - k X_k^{(m)} + (k+1) X_{k+1}^{(m)})$. In summary, the average change in $X_k^{(m)}$ during dt is

$$dt \times \Bigg[\lambda\big(X_{k-1}^{(m)} - X_k^{(m)}\big) - \big(X_k^{(m)} - X_{k+1}^{(m)}\big) + \beta\Bigg[\big(X_{k-1}^{(m)} - X_k^{(m)}\big) \sum_j j X_j^{(m)} - k X_k^{(m)} + (k+1) X_{k+1}^{(m)} \Bigg] \Bigg].$$

There is no explicit dependence on m, and hence we expect the dynamics of $X_k^{(m)}(t)$ to be close to those of a deterministic solution x_k of the following sets of differential equations: for all $k \in \{0, \ldots, B\}$,

$$\dot{x}_k = \lambda(x_{k-1} - x_k) - (x_k - x_{k+1}) + \beta\Bigg[(x_{k-1} - x_k) \sum_j j x_j - k x_k + (k+1) x_{k+1} \Bigg], \tag{14}$$

with the convention that $x_{-1} = 0 = x_{B+1}$. We may write similar differential equations for the evolution of $S_k^{(m)}$. We obtain: for all $k = 0, \ldots, B$,

$$\dot{s}_k = \left(\lambda + \beta \sum_{j \geq 1} s_j\right)(s_{k-1} - s_k) - (1 + \beta k)(s_k - s_{k+1}), \tag{15}$$

with the convention that $s_{-1} = 0 = s_{B+1}$. Next we formally justify the above analysis and show that (14) gives an estimate of system behavior that becomes exact when $m \to \infty$.

Transient regime The next theorem states that the approximation is exact over finite time-horizons, and is a direct application of Kurtz's theorem, see Chap. 11 in [9].

Theorem 5 *Assume that* $\lim_{m\to\infty} X^{(m)}(0) = x(0)$ *almost surely. Fix* $t > 0$. *We have*: *almost surely*,

$$\lim_{m\to\infty} \sup_{u \leq t} \left\| X^{(m)}(u) - x(u) \right\| = 0, \tag{16}$$

where $x(\cdot)$ *is the unique solution of* (14) *with initial condition* $x(0)$.

Proof First, one can easily represent the family of processes $(X^{(m)}(t), t \geq 0)$ as a family of *density dependent population processes* as, for example, defined in [9]. Then, define $F : \mathbb{R}^{B+3} \to \mathbb{R}^{B+3}$ by: for all $x \in \mathbb{R}^{B+3}$, $F_{-1}(x) = 0 = F_{B+1}(x)$ and, for all $k = 0, \ldots, B$,

$$F_k(x) = x_{k-1}\left(\lambda + \beta \sum_j j x_j\right) - x_k(\lambda + \beta k + 1) + x_{k+1}.$$

Now (14) writes $\dot{x} = F(x)$. F is Lipschitz on $\mathcal{T} = \{x \in \mathbb{R}_+^{B+3} : x_{-1} = 0 = x_{B+1}, \sum_{k=0}^{B} x_k = 1\}$. As a consequence, the conditions of the theorem stated in [9, p. 456] are met, and we deduce the expected result. □

Stationary regime The above theorem holds for finite time-horizons only. It does not say anything about the long-term behavior of the system and in particular for example about the average stationary client sojourn time. To circumvent this difficulty, we may use the advanced framework formalized by Sznitman [22] and further developed in [13], and more recently in [6]. Here we follow step by step the arguments presented in [13] to prove the convergence of the steady-state behavior of finite systems towards the equilibrium point of dynamical system (14) when $m \to \infty$ and for finite B. Denote by $X_{\text{eq}}^{(m)}$ the stationary empirical distribution of the system with m servers (such distribution exists because $(N^{(m)}(t), t \geq 0)$ is an irreducible finite-state Markov process, and thus positive recurrent). Next, observe that the family of distributions $X_{\text{eq}}^{(m)}$ is tight (because we restrict our analysis to finite B). Now we can apply the same reasoning used in [13] to show that if (14) admits a unique equilibrium point, then any accumulation point of the family $(X_{\text{eq}}^{(m)}, m \geq 0)$ corresponds to this equilibrium point. More precisely we have the following theorem.

Theorem 6 *Assume that from any initial condition in $\mathcal{T}$, the solution of (14) converges to a unique equilibrium point ξ. Then $X_{\mathrm{eq}}^{(m)}$ converges to ξ when $m \to \infty$.*

It should be noted that the above result heavily relies on the assumption that B is finite. When $B = \infty$, a first difficulty is to actually prove the existence of solutions of the system of ODEs (14) (when $B = \infty$, the functions F_k are not uniformly Lipschitz, in k). A second difficulty is to prove the tightness of $(X_{\mathrm{eq}}^{(m)}, m \geq 0)$.

From the previous theorem, we know that in a system of m servers, the proportion of servers handling k clients in the stationary regime gets close to ξ_k as m grows large. We may also approximate the average number of clients in the system by $\sum_{k\geq 1} k\xi_k$ and deduce an estimate of the average sojourn time using Little's formula. It remains to show that the system of differential equations (14) converges to a unique equilibrium point ξ, and to characterize ξ.

Let ξ be a fixed point of (14), then we easily see that: for all $i = 1, \ldots, B$,

$$\xi_i = \xi_0 \times \frac{(\lambda + \beta y)^i}{\prod_{j=1}^{i}(1 + \beta j)},$$

where $y = \sum_j j\xi_j$. ξ_0 is obtained so that ξ is a probability measure. Finally, y must solve

$$y \times \left[1 + \sum_{i=1}^{B} \frac{(\lambda + \beta y)^i}{\prod_{j=1}^{i}(1 + \beta j)}\right] = \sum_{i=1}^{B} i \frac{(\lambda + \beta y)^i}{\prod_{j=1}^{i}(1 + \beta j)}. \tag{17}$$

One can check that if $\lambda < 1$, (17) indeed has a unique positive solution y: if $z = \lambda + \beta y$, z must solve $g(z) = 0$ with

$$g(z) = \frac{(z - \lambda)}{\beta}\left[1 + \sum_{i=1}^{B} \frac{z^i}{\prod_{j=1}^{i}(1 + \beta j)}\right] - \sum_{i=1}^{B} i \frac{z^i}{\prod_{j=1}^{i}(1 + \beta j)}.$$

We have

$$g(z) = \sum_{i=0}^{B+1} \alpha_i z^i,$$

where $\alpha_0 = -\lambda/\beta$, $\alpha_{B+1} > 0$, and for $i = 1, \ldots, B$,

$$\begin{aligned}\alpha_i &= \frac{1}{\beta}\left(\frac{1}{\prod_{j=1}^{i-1}(1 + \beta j)} - \frac{\lambda}{\prod_{j=1}^{i}(1 + \beta j)} - \frac{i\beta}{\prod_{j=1}^{i}(1 + \beta j)}\right) \\ &= \frac{1}{\beta \prod_{j=1}^{i}(1 + \beta j)}(1 - \lambda) > 0.\end{aligned}$$

The result follows from $g(\lambda) < 0$ and $g'(z) > 0$ for all $z \geq 0$. In summary, the unique equilibrium point of (14) is ξ.

Theorem 7 *From any initial condition $x(0) \in \mathcal{T}$, if $\lambda < \mu$, the system of differential equations (14) converges to the unique equilibrium point ξ.*

Proof The system enjoys the following important monotonicity property. Consider two initial conditions $x(0)$ and $x'(0)$ such that[1] $x(0) \leq_{\mathrm{st}} x'(0)$, then if x and x' are the solutions of (14) with respective initial conditions $x(0)$ and $x'(0)$, we have at any time $t \geq 0$, $x(t) \leq_{\mathrm{st}} x'(t)$. The proof of this property is based on a probabilistic interpretation of the dynamical system (14) as the Kolmogorov equations of a collection of birth–death processes of birth rate $\lambda + \beta \sum_j j x_j$ and death rates $(1+\beta k)$ in state k. The idea is that for any $s \geq 0$, $x(s) \leq_{\mathrm{st}} x'(s)$ implies that $\sum_j j x_j(s) \leq \sum_j j x'_j(s)$, so the birth rate at time s for x is smaller than that for x', and by a standard coupling argument, we deduce that just after time s, we still have $x(s+) \leq_{\mathrm{st}} x'(s+)$. We may further deduce that this ordering remains valid over time.

Denote by x^E (resp., x^F) the solution of (14) when the system is initially empty $x^E(0) = (1, 0, \ldots, 0)$ (resp., full $x^F(0) = (0, \ldots, 0, 1)$). A direct consequence of the above monotonicity property is that $x^E(t)$ (resp., $x^F(t)$) is stochastically increasing (resp., decreasing) over time. For example, for all $h, t \geq 0$, $x^E(t+h) \geq_{\mathrm{st}} x^E(t)$. This implies that both $x^E(t)$ and $x^F(t)$ converge to ξ when $t \to \infty$ (since the equilibrium point is unique). We deduce that such convergence also holds starting from any initial condition $x(0)$, since again due to the monotonicity property $x^E(t) \leq_{\mathrm{st}} x(t) \leq_{\mathrm{st}} x^F(t)$ for all t. □

5.2.2 RLS algorithm

The large-system approximation method developed above applies to RLS algorithms. We can similarly derive a deterministic approximation for the evolution of the system empirical measure $X^{(m)}$. When $m \to \infty$, this evolution is characterized by: for all $k = 0, \ldots, B$,

$$\begin{aligned}\dot{x}_k &= \lambda(x_{k-1} - x_k) - (x_k - x_{k+1}) \\ &\quad + \beta\Bigg[x_{k-1} \sum_{j \geq k+1} j x_j - x_k \sum_{j \geq k+2} j x_j \\ &\quad - k x_k \sum_{j \leq k-2} x_j + (k+1) x_{k+1} \sum_{j \leq k-1} x_j\Bigg], \end{aligned} \tag{18}$$

with by convention $x_{-2} = x_{-1} = x_{B+1} = x_{B+2} = 0$. Analyzing the dynamical system (18) is not straightforward and deserves a full study, which we skip here. In all numerical experiments presented below, we verified the convergence of (18) to a unique equilibrium point.

5.2.3 Extension to heterogenous systems and arbitrary random walks (for RLO algorithm)

The above asymptotic analysis has been simplified by considering homogenous systems and uniform random walks (for RLO) only. However, in the case of RLS al-

[1] $\leq_{\mathrm{st}}$ denotes the usual strong stochastic order, i.e., if x, y are probability measures on $\{0, \ldots, m\}$, $x \leq_{\mathrm{st}} y$ iff for all j, $\sum_{i=0}^{j} x_i \geq \sum_{i=0}^{j} y_i$.

gorithm, it can be easily extended to the case of heterogenous systems, where the arrival rates and server speeds are not identical. To do so, we may classify server according to their arrival rate and speed—servers of the same class have same arrival rate and speed. Then, we can derive a set of differential equations, similar to (14) or (18), approximating the evolution of the proportion of servers of a given class and handling a given number of clients. We obtain a dynamical system whose variables $x_{v,k}$ represent the proportion of servers of class v having k clients. In the case of RLO algorithm, the analysis may also be extended to arbitrary random walks; it suffices to include into the server class the rates at which clients jump towards other servers. For example, servers of class v have the same arrival rate and speed, and the rate at which a client at one of class-v servers jumps to a server of class v' depends on v and v' only. In [6], the authors present such multi-class asymptotic analysis in details.

5.3 Numerical experiments

We now illustrate the results derived in this section via simple numerical experiments. To evaluate the relative performance of RLO and RLS algorithms, we consider first an homogenous system (for all i, $\lambda_i = \lambda$, $\mu_i = 1$), and then an extreme heterogenous system where all clients arrive at the same server ($\lambda_1 = m\lambda$, and for all $i \geq 2$, $\lambda_i = 0$). The system performance is expressed in terms of the average client throughput, defined as the inverse of the average sojourn time.

Figure 1 gives the average client throughput as a function of λ in homogenous systems. We compare the results obtained through the large-system asymptotics $m = \infty$ and those obtained for $m = 10$ servers. The results for finite systems are obtained using Monte Carlo simulations. The underlying processes are Markovian jump processes: we first simulate the processes for 10^7 jumps to reach steady state, and then simulate again the system evolution over 10^7 more jumps to estimate the average throughput.

Note that the asymptotics results are pretty accurate even for small systems. Actually at a load of 0.8, the relative error made in our approximations of the average throughput under RLO and RLS algorithms is less than 4 % when $m = 5$, and becomes less than 0.5 % for $m = 20$. Note that RLO and RLS are both stable if and only

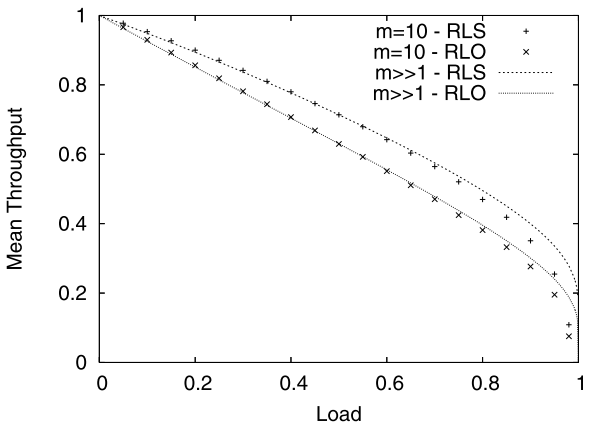

Fig. 1 Mean throughput under RLS and RLO in homogenous systems as a function of the load λ. $\beta = 0.5$

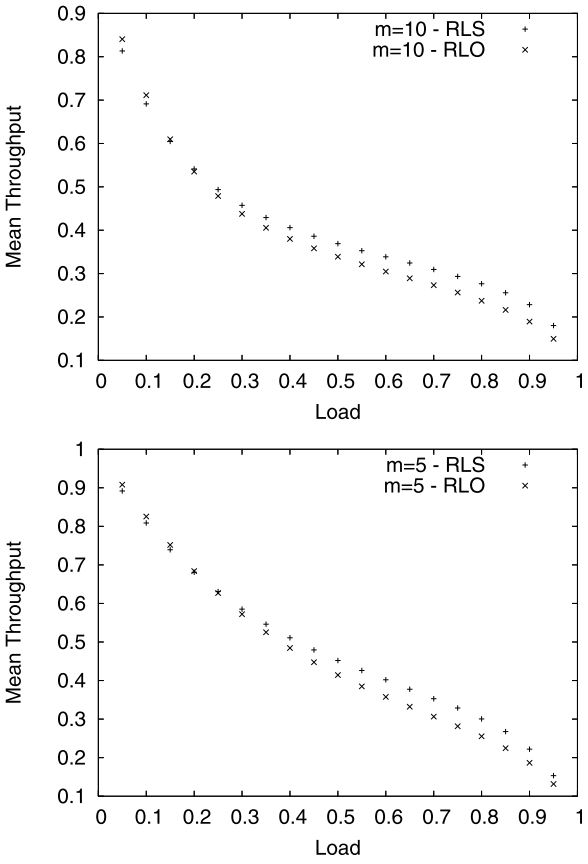

Fig. 2 Mean throughput under RLS and RLO in heterogenous systems as a function of the load λ. $\beta = 0.5$

if $\lambda < 1$. Surprisingly, the performance improvement achieved by the load-dependent RLS algorithm over that obtained under the load-oblivious RLO algorithm is not that significant, typically less than 20 %.

Figure 2 provides the performance in heterogenous systems with $m = 5$ and $m = 10$. We provide simulation results only, although, as explained above, we could have obtained analytic asymptotic results. Again as expected, even if all clients arrive at the same server, RLS and RLO stabilize the system whenever possible (when $\lambda < 1$). The difference between the throughput achieved by RLS and RLO is quite small irrespective of the number of servers considered. Hence it seems that implementing a load-dependent resampling and migration algorithm may not significantly improve the performance.

6 Conclusion

In this paper, we have analyzed the performance of distributed load balancing schemes where clients independently decide to resample and change server to improve their service rate. We considered two natural random resampling and migra-

tion strategies: A load oblivious strategy RLO where clients randomly move from one server to another without accounting for the actual server loads, and a load-dependent selfish strategy RLS where clients randomly resample servers and migrate only if their rate is improved.

In closed systems where the population of clients is fixed, we have provided a new tight bound on the time to balance server loads under RLS strategy. This time can be interpreted as the time to reach a Nash Equilibrium in this selfish routing game. Our bound considerably improves the bounds available in the literature. But it holds only in the case of homogenous systems where servers have identical service rates. It seems challenging and interesting to figure out how to apply our methodology to obtain bounds on the time to balance the system in the case of heterogenous systems. It might also be interesting to investigate the time it takes to balance the system in scenarios where client migrations are limited, in the sense that from a given server, clients can migrate to a restricted subset of servers (as, for example, specified via a graph).

In open systems where clients arrive at the various servers at different rates, we provided a first analysis of the system dynamics. These dynamics are complicated as the client arrival and departure processes interact with the client migration processes. We have shown that both RLO and RLS load balancing strategies are able to stabilize the system whenever this is at all possible. It may appear somehow surprising that a completely distributed and load-oblivious algorithm such as RLO can achieve maximum stability. Using large-system asymptotics, we also provided approximate estimates of the mean client sojourn time. The results show that again, surprisingly, the load-oblivious RLO strategy does not yield significant performance losses compared to the load-dependent RLS strategy. These findings are valid for exponential service requirements, and it would be interesting to know whether they remain valid for other service requirement statistics.

An interesting extension of the present work (especially relevant when considering spectrum sharing issues) is to analyze the case where clients may use resources from several servers simultaneously. There are some preliminary results in this direction in [15], but neither the time to reach equilibrium nor the population dynamics are studied.

References

1. IEEE Conference on Dynamic Spectrum Access. http://www.ieee-dyspan.org/
2. Anshelevich, D., Kleinberg, J.: Stability of load balancing algorithms in dynamic adversarial systems. SIAM J. Comput. **37**(5), 1656–1673 (2008)
3. Berenbrink, P., Friedetzky, T., Goldberg, L.A.: The natural work-stealing algorithm is stable. SIAM J. Comput. **32**(5), 1260–1279 (2003)
4. Berenbrink, P., Friedetzky, T., Goldberg, L.A., Goldberg, P., Hu, Z., Martin, R.: Distributed selfish load balancing. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 354–363 (2006)
5. Berenbrink, P., Friedetzky, T., Hajirasouliha, I., Hu, Z.: Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks. In: LNCS, vol. 4698, pp. 41–52. Springer, Berlin/Heidelberg (2007)
6. Bordenave, C., McDonald, D., Proutiere, A.: A particle system in interaction with a rapidly varying environment: mean field limits and applications. AMS J. Netw. Heterog. Media (2010). doi:10.3934/nhm.2010.5.31

7. Borst, S., Proutiere, A., Hegde, N.: Capacity of wireless data networks with intra- and inter-cell mobility. In: Proceedings of IEEE INFOCOM, Barcelona, Spain (2006)
8. Dai, J.G.: On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models. Ann. Appl. Probab. **5**, 49–77 (1995)
9. Ethier, S., Kurtz, T.: Markov Processes. Wiley, New York (1986)
10. Even-Dar, E., Kesselman, A., Mansour, Y.: Convergence time to Nash equilibrium in load balancing. ACM Trans. Algorithms **3**(3) (2007). doi:10.1145/1273340.1273348
11. Feldmann, R., Gairing, M., Lucking, T., Monien, F.B., Rode, M.: Nashification and the coordination ratio for a selfish routing game. In: Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP) (2003). doi:10.1007/3-540-45061-0_42
12. Goldberg, P.W.: Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In: Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing, pp. 131–140 (2004)
13. Graham, C.: Chaoticity on path space for a queueing network with selection of the shortest queue among several. J. Appl. Probab. **37**, 198–211 (2000)
14. Harchol-Balter, M.: Task assignment with unknown duration. J. ACM **49**(2), 260–288 (2002)
15. Key, P., Massoulie, L., Towsley, D.: Path selection and multipath congestion control. In: Proceedings of IEEE INFOCOM (2006)
16. Koutsoupias, E., Papadimitriou, C.H.: Worst-case equilibria. In: Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS), pp. 404–413 (1999)
17. Mitzenmacher, M.: The power of two choices in randomized load balancing. Ph.D. Thesis, University of California Berkeley (1996)
18. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
19. Orda, A., Rom, R., Shimkin, N.: Competitive routing in multi-user communication networks. IEEE/ACM Trans. Netw. **1**, 510–521 (1993)
20. Robert, P.: Stochastic Networks and Queues. Stochastic Modelling and Applied Probability Series. Springer, New York (2003)
21. Simatos, F., Tibi, D.: Spatial homogenization in a stochastic network with mobility. Ann. Appl. Probab. **20**(1), 312–355 (2010)
22. Sznitman, A.: Propagation of chaos. In: Ecole d'ete de probabilites de Saint Flour XIX. Lectures Notes in Mathematics, vol. 1464, pp. 165–251. Springer, Berlin (1991)
23. Vvedenskaya, N.D., Dobrushin, R.L., Karpelevich, F.I.: Queueing system with selection of the shortest of two queues: an asymptotic approach. Probl. Pereda. Inf. **32**(1), 20–34 (1996)