Load Balancing with Migration Penalties

Vivek F. Farias*, Ciamac C. Moallemi*, and Balaji Prabhakar*†
*Electrical Engineering, Stanford University, Stanford, CA 94305, USA Emails: {vivekf,ciamac,balaji}@stanford.edu
†Computer Science, Stanford University, Stanford, CA 94305, USA

***Abstract*— Many practical systems perform load balancing. The main aim of load balancing is to utilize the capacity of a system of parallel processors efficiently and to reduce the delay of processing jobs. This paper is concerned with load balancing, or process migration, when there is a penalty associated with migration.**

We consider the following model: Jobs arrive at each of n parallel servers. An arriving job can either be processed in a unit of time, on average, at the server where it arrives, or it can migrate to another server where it creates $K \geq 1$ independent jobs. When $K = 1$, migrating jobs impose no extra cost and this problem is considered extensively in the literature. We are interested in the situation $K > 1$. The problem is to decide whether a job should migrate or not. On the one hand migration leads to load balancing and hence reduces backlogs. However, it also leads to the creation of extra work and, hence, to a potential loss of throughput. We ask: Do there exist simple migration policies that can reduce backlogs while providing the highest throughput? Somewhat surprisingly, we find that policies like "migrate to the least loaded server" are unstable: they cause a loss of throughput. However, we find that a simple variant of this rule is stable *and* leads to a reduction of backlogs.

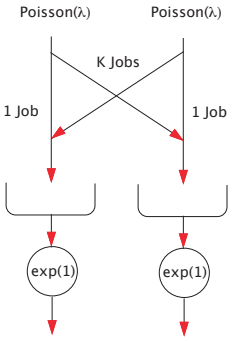
I. Introduction

Consider a system consisting of n servers, each with an associated queue. Jobs arrive according to a Poisson process of rate λ at each queue, and service times are IID exponential random variables with mean 1. In the canonical load balancing problem, an allocator can choose to re-route an arriving job to a queue different from the one it was intended for. An example of such a policy— SQ(d)—is sampling d of the n queues and moving the arriving job to the shortest of those queues. Using such a policy dramatically reduces the average delay for a job with no effect on throughput. Now, consider the case where job migration incurs a penalty.

One way to formalize such a notion is converting a single $\exp(1)$ job into $K > 1$ independent $\exp(1)$ jobs in the event the job is transferred to a queue different from its original destination (see Figure 1). Since load-balancing now adds work to the system it is intuitive to expect that policies that allow for too much migration might result in a loss of throughput. In fact, we will show that the natural extension of the SQ(d) policy, wherein in addition to picking the shortest of d sampled queues, a job is transferred only if the shortest sampled queue has at least K jobs less than the queue at the point of arrival, entails a loss of throughput.

The question now is whether it is possible to achieve the advantages of load balancing (viz. shorter delays) at *no* cost to throughput. We demonstrate such an algorithm: a job is transferred to the shortest of the d sampled queues only if the shortest sampled queue has at least a factor $\alpha > 1$ jobs less than the queue at the point of arrival. Thus we show that even though transferring jobs entails adding more work to the system, we may still realize the benefits of load balancing without sacrificing throughput.

Fig. 1. The model with $n = 2$.



Our model is a variant of the "supermarket" model considered in [1], [2], [3], [4], which corresponds to the case where $K = 1$. [1] and [3] independently studied the performance of SQ(d) in the large-n limit using mean field techniques wherein the behavior of the system is described by an infinite system of (deterministic) ODEs. [2] also uses mean-field analysis to study a load balancing algorithm that involves, in addition to sampling, the use of memory. The main thrust of the work is in justifying the correctness of the mean-field analysis which suggests that these algorithms offer drastic performance improvements. The supermarket model itself applies to a broad variety of interesting problems including dynamic resource allocation, hashing, etc. as pointed out in [5]. The variant we consider here lets us capture, in addition, the notion of a preferred allocation. One well studied example of such a problem is that of caching in multiprocessor systems wherein each processor has a preferred cache[6].

II. Model

For this and the following section, we shall consider the embedded discrete time chains corresponding to the systems of interest. For integer time indices $t \geq 0$, $q_i(t)$ is the amount of work (measured in the number of $\exp(1)$ job equivalents) in the queue for the ith server immediately after the tth event. We assume that the system starts empty. When a job arrives to server i, the server must decide whether to serve the job itself, in which case the job joins the server's queue, or to "bounce"

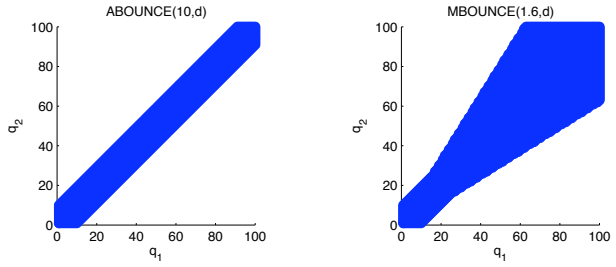
the job and send K jobs to the queue of another server. We will examine two routing strategies:

- ABOUNCE(C,d): This policy decides to share based on an additive threshold.
  When a job arrives at a server i at time t, the server samples d other queues with replacement. Let i^* be the index of the shortest of these queues. If
  $$q_i(t-1) > q_{i^*}(t-1) + C,$$
  then the job is bounced to queue i^*. Otherwise, the job joins the queue at server i. We require that $C \geq K$.
- MBOUNCE(α,d): This policy employs a multiplicative threshold to decide when to share. This threshold has been slightly modified so as not to bounce jobs to another queue when that queue would subsequently have more jobs than the original queue.
  In this scheme, d queues are also sampled as above. However, the job is only bounced to queue i^* if
  $$q_i(t-1) > \max(\alpha q_{i^*}(t-1), q_{i^*}(t-1) + K).$$
  Here, we require $\alpha > 1$.

The switching region under the ABOUNCE policy is far greater than that under MBOUNCE (refer to the caricature in Figure 2) in the following sense. The fraction of states in which no migration occurs under the ABOUNCE policy goes to 0 whereas it remains strictly positive under the MBOUNCE policy. Intuitively, this suggests that the ABOUNCE policy creates too much work and can therefore be unstable while the MBOUNCE policy can be stable. One major purpose of this paper is to rigorously prove these statements.

Note that in both strategies we allow for $d = \infty$; in which case the shortest sampled queue is assumed to be the shortest queue in the system.

Fig. 2. No-migration region (shaded) for $n = 2$ under ABOUNCE and MBOUNCE respectively.



III. Stability

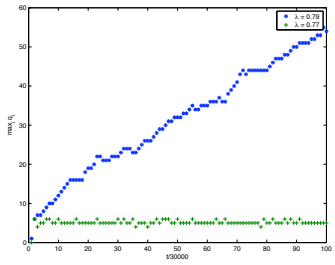
In this section, we will examine the stability policies of the various load balancing strategies. In particular, for each strategy, we are concerned with establishing the set of arrival rates λ such that the resulting system in positive recurrent.

First, consider the case where no sharing is performed. Here, the system trivially splits into n independent $M/M/1$ queues with arrival rate λ and service rate 1. Hence, the system is positive recurrent if $\lambda < 1$. Ideally, the system with sharing should not sacrifice throughput and, thus, we would hope for the same stability region.

Additive Thresholds

For the ABOUNCE(C,d) strategy, by numerical simulation, we can see that the stability region is strictly smaller than the system with no sharing. For example, For example, in Figure 3, we see two sample paths for the ABOUNCE(2,10) system with $K = 2$ and $n = 100$. The system is clearly stable when $\lambda = 0.77$, but unstable when $\lambda = 0.79$.

Fig. 3. Two sample paths for ABOUNCE(2, 10) with $n = 100$ that suggest instability for $\lambda \geq 0.79$.



The ABOUNCE(C,d) policy, however, is dominated by a system with no sharing where each job requires service time which is the sum of K exp(1) random variables. Such a system would be stable if $\lambda < 1/K$. By making this coupling precise, we can establish the following theorem.

Theorem 1: If $\lambda < 1/K$, then the ABOUNCE(C,d) system is positive recurrent.

Further, in the case of 2 queues and $d = \infty$, we can theoretically establish shrinkage of the stability region.

Theorem 2: If $n = 2$, the ABOUNCE(C,∞) system is null recurrent or transient if $\lambda > \lambda^*$, where $\lambda^* < 1$ is defined by

$$\lambda^* = \frac{KC + 1 + \sqrt{(KC+1)^2 + 4(K+1)(C+2)(C+3)}}{2(K+1)(C+2)}.$$

Proof: For the $\lambda > \lambda^*$ case, we appeal to the variation of Foster's Criterion in Proposition 5.4 in [7]. In particular, given a function $V : \mathbb{Z}_+^2 \to \mathbb{R}$, define the drift operator

$$(\Delta V)(q) = \mathbb{E}\left[V(q(1)) | \, q(0) = q\right] - V(q).$$

To establish that the system is null recurrent or transient, it suffices to find a test function V and a finite set $E_0 \subset \mathbb{Z}_+^2$ such that:

(I) For all $q \notin E_0$, $\Delta V(q) \geq 0$.
(II) There exists some $q^* \notin E_0$ so that for all $q \in E_0$, $V(q^*) > V(q)$.
(III) V is bounded below and for some constant $A < \infty$ and all $q \in E_0$,
$$\mathbb{E}\left[|V(q(1)) - V(q(0))| \, | \, q(0) = q\right] < A.$$

Consider the test function

$$V(q) = q_1 + q_2 + \frac{1-\lambda}{1+\lambda}\sum_{\ell=0}^{C} ||q_1 - q_2| - \ell|,$$

and set $E_0 = \{0\}$. This function clearly satisfies conditions (II) and (III). Define the following sets, for $0 \le \ell \le C$,

$$A = \left\{q \in \mathbb{Z}_+^2 - E_0 :\ |q_1 - q_2| > C\right\},$$

$$B_\ell = \left\{q \in \mathbb{Z}_+^2 - E_0 :\ |q_1 - q_2| = \ell,\ q_1, q_2 > 0\right\}.$$

By separately considering the drift $(\Delta V)(q)$ in each of these sets, it can be verified that for $\lambda > \lambda^*$, condition (I) also holds. ∎

In a system with $n = 2$ and $K = 2$ under the ABOUNCE(2,∞) policy, for example, Theorems 1 and Theorem 2 indicate stability when $\lambda < 0.5$ and instability when $\lambda > 0.89$.

Multiplicative Thresholds

The MBOUNCE(α,d) strategy, on the other hand, intuitively bounces less and less as the system gets more and more loaded (see Figure 2). To see this, consider the $n = 2$ case. If one queue is more loaded than the other, bounces will occur and the queues will be equalized. As the levels of the queues increase, it takes more and more time for them to become unequal by a constant fraction again, and hence bouncing decreases. As the following theorem establishes that, because of this behavior, MBOUNCE(α,d) maintains the same stability region as not sharing. The proof is deferred until the appendix.

Theorem 3: MBOUNCE(α,d) is stable for all $\lambda < 1$.

IV. Delay - Formal Mean Field Analysis

We now turn our attention to queuing delay. We approach this in two ways. We will present simulation results in the next section. Here we will present a formal mean-field analysis. Such an analysis permits us to answer questions about rare events in large systems (which is difficult to do via simulation). In particular, we study tail behavior for systems with n large. In this section we revert to considering continuous time Markov-chains as opposed to the embedded chains of the previous section.

Multiplicative Thresholds

Consider the Markov chain corresponding for a system of n queues under the MBOUNCE(d,α) policy. Let $m_i(\tau)$ denote the number of queues with at least i jobs at time τ and observe that the sequence $(m_i)_{i\ge 0}$ suffices to describe the state of the chain at time τ. For ease of exposition we make the assumption that α is integral. It then follows that for $i > K$ and $\Delta\tau$ small,

$$\begin{aligned}
&\mathbb{E}[m_i(\tau+\Delta\tau) - m_i(\tau)|m_i(\tau)] \approx \\
&(\lambda\Delta\tau)\,(m_{i-1}(\tau) - m_i(\tau))\,(m_{\lfloor i/\alpha\rfloor+1}(\tau)/n)^d \\
&+ (\lambda\Delta\tau)\sum_{m=0}^{K-1} m_{\alpha(i-K+m)}(\tau)\Big[(m_{i-K+m}(\tau)/n)^d - \\
&\quad (m_{i-K+m+1}(\tau)/n)^d\Big] - (\Delta\tau)\,(m_i(\tau) - m_{i+1}(\tau)),
\end{aligned}$$

where the first term in the summation corresponds to arrivals at a queue with $i-1$ jobs, the second term corresponds to jobs that are bounced to queues with between $i-K$ and $i-1$ jobs, whereas the last term corresponds to departures from a queue with i jobs. Setting $\Delta\tau = 1/n$, we have:

$$\begin{aligned}
&\frac{\mathbb{E}[m_i(\tau+\Delta\tau)/n - m_i(\tau)/n|m_i(\tau)]}{\Delta\tau} \approx \\
&(\lambda\Delta\tau)\,(m_{i-1}(\tau)/n - m_i(\tau)/n)\,(m_{\lfloor i/\alpha\rfloor+1}(\tau)/n)^d \\
&+ (\lambda\Delta\tau)\sum_{m=0}^{K-1} m_{\alpha(i-K+m)}(\tau)/n\Big[(m_{i-K+m}(\tau)/n)^d - \\
&\quad (m_{i-K+m+1}(\tau)/n)^d\Big] - (\Delta\tau)\,(m_i(\tau)/n - m_{i+1}(\tau)/n).
\end{aligned}$$

Letting $\lim_{n\to\infty}\mathbb{E}[m_i(\tau)/n] = s_i(\tau)$, and noting that by the Law of Large Numbers, $\lim_{n\to\infty} m_i(\tau)/n \to s_i(\tau)$, we (formally) have that for large n, the MBOUNCE(d, α) system is described by the following system of ODEs:

$$\begin{aligned}
&\dot{s}_i(\tau) = \\
&\lambda\,(s_{i-1}(\tau) - s_i(\tau))\,s^d_{\lfloor i/\alpha\rfloor+1}(\tau) \\
&\quad + \lambda\sum_{m=0}^{K-1} s_{\alpha(i-K+m)}(\tau)\left(s^d_{i-K+m}(\tau) - s^d_{i-K+m+1}(\tau)\right) \\
&\quad - (s_i(\tau) - s_{i+1}(\tau)), \quad \forall\ i > K,\ \tau > 0,
\end{aligned}$$

$$\begin{aligned}
\dot{s}_i(\tau) &= \lambda\,(s_{i-1}(\tau) - s_i(\tau)) \\
&\quad + \lambda\sum_{m=0}^{i-1} s_{\max(m+K,\alpha m)}(\tau)\left(s^d_m(\tau) - s^d_{m+1}(\tau)\right) \\
&\quad - (s_i(\tau) - s_{i+1}(\tau)), \quad \forall\ 0 < i \le K,\ \tau > 0, \qquad (1)\\
\dot{s}_0(\tau) &= 0, \quad \forall\ \tau > 0, \\
s_0(0) &= 1, \\
s_i(0) &= 0, \quad \forall\ i > 0.
\end{aligned}$$

Fixed points of this system of ODEs have the following property which suggests that for large n, the MBOUNCE(d, α) system has far lighter tails than a system with no sharing.

Lemma 1: Any fixed point of the system of equations (1), satisfying $\sum_{i=0}^{\infty} s_i < \infty$ and $s_i \ge 0\ \ \forall i$ must also satisfy:

$$s_i = O\left((\lambda(1+\epsilon))^{\frac{i^{\log_\alpha(d+1)}-1}{d}}\right).$$

for arbitrary $\epsilon > 0$.

Additive Thresholds

Similar reasoning as the multiplicative case suggests the following system of differential equations for the tails of an ABOUNCE(d) system with n large:

$$\begin{aligned}
\dot{s}_i(\tau) &= \lambda\,(s_{i-1}(\tau) - s_i(\tau))\,s^d_{i-K+1}(\tau) \\
&\quad + \sum_{m=0}^{K-1} \lambda s_{i+m}(\tau)\left(s^d_{i-K+m}(\tau) - s^d_{i-K+m+1}(\tau)\right) \\
&\quad - (s_i(\tau) - s_{i+1}(\tau)), \quad \forall\ i > K,\ \tau > 0,
\end{aligned}$$

$$
\begin{aligned}
\dot{s}_i(\tau) &= \lambda\left(s_{i-1}(\tau) - s_i(\tau)\right) \\
&\quad + \sum_{m=0}^{i-1} \lambda s_{m+K}(\tau)\left(s_m^d(\tau) - s_{m+1}^d(\tau)\right) \\
&\quad - \left(s_i(\tau) - s_{i+1}(\tau)\right), \quad \forall\ 0 < i \leq K,\ \tau > 0, \qquad (2) \\
\dot{s_0}(\tau) &= 0, \quad \forall\ \tau > 0, \\
s_0(0) &= 1, \\
s_i(0) &= 0, \quad \forall\ i > 0.
\end{aligned}
$$

Fixed points of this system of ODEs have the following property which suggests that for large n, the ABOUNCE(d) system has lighter tails than both the system with no sharing as well as the MBOUNCE(d, α) system for large n. Of course, we also know from the previous section that such an improvement in tail behavior comes at a cost to throughput.

Lemma 2: Any fixed point of the system of equations (2), satisfying $\sum_{i=0}^{\infty} s_i < \infty$ and $s_i \geq 0\ \ \forall i$ must also satisfy:

$$
s_i = O\left((\lambda(1+\epsilon))^{\frac{(d+1)\lfloor i/2(K+1)\rfloor - 1}{d}}\right).
$$

for arbitrary $\epsilon > 0$.

V. Delay - Numerical Results

In Figures 4 and 5, we can see empirical queue size tails for the ABOUNCE(C,d) and MBOUNCE(α,d) policies as compared with no sharing. In both cases, a significant improvement is seen versus the policy of not sharing. Further, the multiplicative threshold policies offer better performance than additive policies for this system.

Fig. 4. Queue-size tails for ABOUNCE($10, C$) vs. tails under no sharing. ($n = 100, \lambda = 0.75, K = 2$)

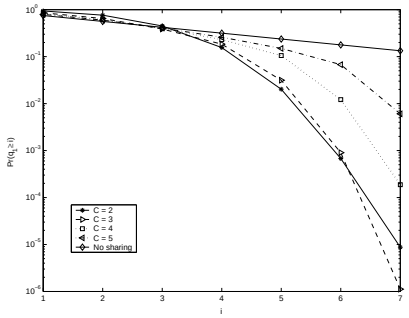


Fig. 5. Queue-size tails for MBOUNCE($10, \alpha$) vs. tails under no sharing. ($n = 100, \lambda = 0.75, K = 2$)

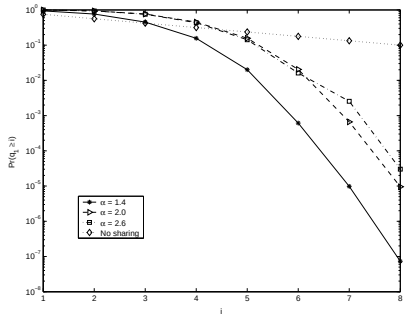
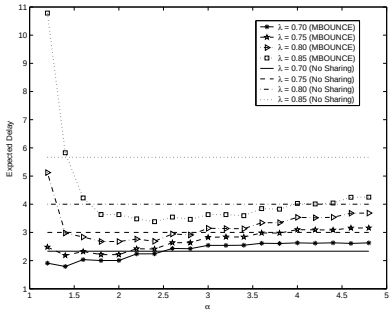


Fig. 6. Empirical expected delay for MBOUNCE(α,10) policy with varying α versus no sharing. ($n = 100$, $K = 2$)



VI. Conclusion

We have presented two algorithms for load balancing when there is a cost to transferring jobs. The use of the MBOUNCE algorithm provably entails no loss in throughput. Simulation results, as well as a formal mean-field analysis strongly suggest that this algorithm affords improvements in waiting time as well as lighter queue-tails than the no-sharing case. Several issues remain:

1) Can the mean field analysis of section IV be rigorized? The large deviations techniques used in [2] do not appear to extend.
2) Is there an 'optimal' decision boundary to decide between sharing and not-sharing a job?

Acknowledgments

The first author was supported by a supplement to NSF Grant ECS-9985229 provided by the MKIDS Program. The second author was supported by a Benchmark Stanford Graduate Fellowship. The authors wish to thank Devavrat Shah and Chandra Nair for helpful discussions.

References

[1] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, University of California, Berkeley, 1996.
[2] M. Mitzenmacher, B. Prabhakar, and D. Shah, "Load balancing with memory," in *Proceedings of FOCS*, 2002.
[3] N. Vvedenskaya, R. Dobrushin, and F. Karpelevich, "Queueing system with selection of the shortest of two queues: An asymptotic approach," *Problems of Information Transmission*, vol. 32, no. 1, pp. 15–29, 1996.
[4] C. Nair, B. Prabhakar, and D. Shah, "The randomness in randomized load balancing," in *Proceedings of the 39th Annual Allerton Conference on Communication, Control and Computing*, October 2001, pp. 912–921.
[5] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced allocations," *SIAM J. Comput.*, vol. 29, no. 1, pp. 180–200, 2000.
[6] M. S. Squillante and E. D. Lazowska, "Using processor-cache affinity information in shared-memory multiprocessor scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 2, pp. 131–143, 1993.
[7] S. Asmussen, *Applied Probability and Queues*, 2nd ed. Springer-Verlag, 2003.

APPENDIX: PROOF OF THEOREM 3

We use the Lyapunov function $V(q) = V_1(q)+V_2(q)$ where,

$$V_1(q) = \sum_{i=1}^{n} q_i^2,$$

and

$$V_2(q) = \frac{1}{\alpha - 1} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (q_i - q_j)^2 .$$

To prove positive recurrence (i.e. stability) for the MBOUNCE Markov chain (see Proposition 5.3 in [7]), it will suffice to show for some $\epsilon > 0$ that $(\Delta V)(q) \leq -\epsilon$ for all $q \notin K$ where K is some compact set.

For ease of exposition, we will assume for the balance of this proof that $\alpha = 2$. The same technique can be applied for other α. We also assume, without loss of generality, that the queues are numbered so that at time 0, $q_i \geq q_j$ for $i \leq j$; in the event of ties we assign the queue with a higher number prior to re-numbering, the higher number. In particular, after renumbering q_1 is the longest queue, while q_n is the shortest. We define the event $\mathcal{E}_n = \{q_1 < 2q_n\} \cup \{q_1 - q_n < K\}$. That is, $\mathcal{E}_n$ is the event that no transfer or arriving jobs is possible at the next epoch.

It is then straightforward to check that for $q \in \mathcal{E}_n$,

$$(\Delta V)(q) = \frac{2}{N} \frac{\lambda - 1}{\lambda + 1} \sum_{i=1}^{n} q_i + C_1, \tag{3}$$

where C_1 is a constant depending on n, λ.

Now, for $m \in \{1, \ldots, n-1\}$, define $\mathcal{E}_m = \{q_m \geq 2q_n, q_m - q_n \geq K\} \cup \{q_{m+1} < 2q_n\} \cup \{q_{m+1} - q_n < K\}$. Essentially, $\mathcal{E}_m$ is the event wherein a transfer of job is possible and the shortest queue from which a transfer is possible is q_m. Further, define,

$$p_{j,m} = P\left(\operatorname*{argmin}_{i \in \mathcal{S}} q_i = j \middle| \mathcal{E}_m \right),$$

where $\mathcal{S}$ is the set of indexes of the d queues sampled in the event of an arrival. Thus $p_{j,m}$ is the probability that the shortest of the d queues sampled is queue j (breaking ties in favor of the higher numbered queue). An elementary argument then gives $p_{j,m} \leq p_{j+1,m}$. Also, let $p_B = \sum_{j=m+1}^{n} p_{j,m}$. It is readily checked that for $q \in \mathcal{E}_m$:

$$\begin{aligned}(\Delta V_1)(q) \leq & \sum_{i=m+1}^{n} \frac{1}{n} \frac{\lambda - 1}{\lambda + 1} q_i \\ & + \sum_{i=1}^{m} \Big[\frac{\lambda/n}{\lambda + 1} \Big[\sum_{j=m+1}^{n} 2p_{j,m} K q_j + 2(1 - p_B) q_i \Big] \\ & \qquad - \frac{1/n}{\lambda + 1} 2q_i \Big] + C_{1,m}.\end{aligned} \tag{4}$$

We now consider $(\Delta V_2)(q)$. In evaluating $(\Delta V_2)(q)$ for $q \in \mathcal{E}_m$, we make the following observations:

(I) It is straightforward to check that departures from $q_i, 1 \leq i \leq n$ or arrivals to $q_i, m+1 \leq i \leq n$ contribute at most a constant to the expectation.

(II) Consider an arrival to $q_i, 1 \leq i \leq m$. We get the following positive contribution to the expectation for when it is unsuccessful in transferring its job:

$$(1 - p_B) \sum_{j=i+1}^{n} \frac{\lambda/n}{\lambda + 1} 2(q_i(0) - q_j(0)).$$

However, this contribution is negated by arrivals to queues numbered $j > i$,

$$\frac{-2(1 - p_B)\lambda}{n(\lambda + 1)} \Big[\sum_{j=i+1}^{m} (q_i(0) - q_j(0)) - \sum_{j=m+1}^{n} (q_i(0) - q_j(0)) \Big]$$

Now, consider the situation where i is successful at transferring its job to some queue. This transfer reduces the difference in lengths between i and the queue it transfers its job to, which results in the following negative contribution

$$-\frac{\lambda/n}{\lambda + 1} \sum_{j=m+1}^{n} 2p_{j,m} K q_j(0).$$

The difference in queue lengths between the queue to which i transferred its job and the queues below it increases, which contributes the following positive term to the expectation:

$$\frac{\lambda/n}{\lambda + 1} \sum_{j=m+1}^{n} p_{j,m} \sum_{\ell=j+1}^{n} 2K(q_j(0) - q_\ell(0)). \tag{5}$$

This term is negated since transfers from i reduce the difference in queue lengths between queues receiving a transfer and larger queues in $\{m+1, \ldots, n\}$ contributing:

$$\frac{-\lambda/n}{\lambda + 1} \sum_{j=m+1}^{n} p_{j,m} \sum_{\ell=m+1}^{j-1} 2K(q_j(0) - q_\ell(0)). \tag{6}$$

(We conclude (5) + (6) ≤ 0 using $p_{j,m} \leq p_{j+1,m}$.)

Putting these observations together, we have that an arrival to $q_i, 1 \leq i \leq m$ contributes to the expectation, a term upper bounded by:

$$-\left[\frac{\lambda/n}{\lambda + 1} \sum_{j=m+1}^{n} 2p_{j,m} K q_j(0) \right] + C.$$

Putting our observations together yields that for $q \in \mathcal{E}_m$

$$(\Delta V_2)(q) \leq -\left[\sum_{i=1}^{m} \frac{\lambda/n}{\lambda + 1} \sum_{j=m+1}^{n} 2p_{j,m} K q_j \right] + C_{2,m}. \tag{7}$$

From (4) and (7), we have that for $q \in \mathcal{E}_m$,

$$\begin{aligned}(\Delta V)(q) \leq & \sum_{i=m+1}^{n} \frac{1}{n} \frac{\lambda - 1}{\lambda + 1} q_i + \sum_{i=1}^{m} \left[-\frac{p_B/n}{\lambda + 1} 2q_i \right] \\ & + C_{1,m} + C_{2,m}.\end{aligned} \tag{8}$$

Of course, (8) holds for all $m, 1 \leq m \leq n-1$, and so together with (3) we have

$$(\Delta V)(q) \leq -\epsilon,$$

for $\sum_{i=1}^{n} q_i$ sufficiently large ($\{\sum_{i=1}^{n} q_i \geq N_\epsilon\}$, say). Since the complement of this set is compact, we are done.