Optimal Load Balancing on Distributed Homogeneous Unreliable Processors

Zhen Liu, Rhonda Righter

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Optimal Load Balancing on Distributed Homogeneous Unreliable Processors

Zhen Liu and Rhonda Righter

N° 2659

Septembre 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Optimal Load Balancing on Distributed Homogeneous Unreliable Processors*

Zhen Liu and Rhonda Righter

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués

Projet MISTRAL

Rapport de recherche n° 2659 — Septembre 1995 — 22 pages

Abstract: We consider optimal load balancing in a distributed computing environment with several homogeneous unreliable processors that have limited state information. Each processor receives its own arrival process of tasks from outside users, some of which can be redirected to the other processors. Processing times are *iid* and arbitrarily distributed. The arrival process of outside tasks to each processor may be arbitrary as long as it is independent of the state of the system. Processors may fail, with arbitrary failure and repair processes that are also independent of the state of the system. The only information available to a processor is the history of its decisions for routing work to other processors, and the arrival times for its own arrival process. We show that the round-robin policy, in which each processor sends the tasks that can be redirected to each of the processors in turn, *stochastically* minimizes the n^{th} task completion time for all n, and minimizes response times and queue lengths in a separable increasing convex sense, among all policies that balance workload. We also show that if there is a single centralized controller, round-robin is the optimal policy, and a single controller using round-robin routing is better than the optimal distributed system, where "optimal" and "better" are in the sense of stochastically minimizing task completion times and minimizing response times and queue lengths in the separable increasing convex sense.

Key-words: Load Balancing; Optimal Routing; Traffic Allocation; Scheduling; Round-Robin; Stochastic Ordering; Sample Path Analysis

*Correspondence: Zhen LIU, INRIA, Centre Sophia Antipolis, 2004 route des Lucioles, B.P. 93, 06902 Sophia-Antipolis, France. e-mail: liu@sophia.inria.fr

(Résumé : tsvp)

Equilibre de Charge sur Processeurs Distribués

Résumé : Le problème d'équilibre de charge optimal est étudié pour un système réparti composé de plusieurs processeurs homogènes non fiables. Chaque processeur reçoit individuellement les tâches des utilisateurs extérieurs dont certaines peuvent être redirigées vers les autres processeurs. Les temps de traitement des tâches sont *iid* avec une distribution générale. Le processus d'arrivées des tâches extérieures à chacun des processeurs est arbitraire pourvu qu'il soit indépendant de l'état du système. La seule information dont un processeur dispose est l'histoire de ses propres décisions de routage des tâches vers les autres processeurs ainsi que les dates d'arrivées de ses tâches. Il est démontré que, parmi les politiques d'équilibre de charge, la politique "round-robin" dans laquelle chaque processeur envoie les tâches redirigeables vers les autres processeurs d'une manière cyclique, minimise stochastiquement les dates de fin de traitement des tâches, et minimise, dans le sens convexe croissant séparable, les temps de réponse des tâches et les nombres de tâches en attente dans les processeurs. Il est aussi prouvé que s'il y a un seul contrôleur centralisé, la politique "round-robin" est optimale, et qu'un seul contrôleur centralisé appliquant "round-robin" est meilleur que les politiques de routage distribuées.

Mots-clé : équilibre de charge, routage optimal, allocation de trafic, ordonnancement, ordre stochastique, analyse trajectorielle.

1 Introduction

An important architecture for multiprocessor systems is the distributed architecture in which relatively autonomous processors are networked together (e.g., Gelenbe, 1989). Such systems are becoming increasingly prevalent with the proliferation of personal computers and workstations and of local area networks. Thus, efficient allocation of the multiple resources, or load balancing, in distributed computing systems is becoming increasingly important. Load balancicng in such systems is complicated by the fact that there is usually not a centralized controller, and distributed processors typically do not have full state information for other processors. It is further complicated if processors are unreliable. See Wang and Morris (1985), Boel and Van Schuppen (1989), and Gelenbe and Pekergin (1993), for an overview of load balancing protocols and issues.

In our distributed computing model, we have a set of homogeneous unreliable processors with limited information. By homogeneous we mean that the offered traffic, processing capabilities, and failure and repair times for all processors are stochastically the same. Each processor receives its own arrival process of tasks from outside users. Some of these tasks are "local", that is, they must be executed in first-come-first-served (FCFS) order by the processor that receives them. This may be because of precedence constraints among tasks or excessive communication between them. The other tasks are "general" and can be redirected to the other processors. General tasks that are received by a processor are executed by that processor in FCFS order. We consider how each processor should allocate its "general traffic" to the other processors when the only information available to a processor is the history of its decisions for routing work to other processors, and the arrival times for its own general arrival process.

Most prior work on load balancing for homogeneous processors has been done assuming there is a single centralized controller and processors are perfectly reliable. It is often the case for these systems that routing tasks to the processor with shortest expected workload is optimal. The main feature that distinguishes different problems is the information available to the controller when it makes its decision. In the most restrictive case, the controller knows only that it has an arrival, and therefore has no information on workloads. In this case Bernoulli routing with equal probabilities, which will stochastically balance the workloads, has been shown to be optimal under a variety of assumptions about distributions and cost functions (Chang, 1992, Chang, Chao, and Pinedo, 1990, Gün and Jean-Marie, 1993, Koole, 1994).

In our model each processor knows only the history of its decisions and its general arrival times, and the system starts with stochastically equal workload at all of the processors. In this case, for a centralized controller routing to homogeneous processors, Ephremides, Varaiya, and Walrand (1980) showed that the round-robin policy minimizes the sum of the completion times for the first n tasks, for all n, when processing times are exponential and there are no processor failures. Liu and Towsley (1994a) showed that the round-robin policy minimizes response times and queue lengths in a separable increasing convex sense, when processing times have increasing failure rates and processors are perfectly reliable. Comparisons between Bernoulli and round-robin policies, which are both considered static policies, were made in Stoyan (1983) and Jean-Marie and Liu (1992).

When the centralized controller also has queue length information dynamic policies can be considered. When processing times are exponentially distributed, shortest queue routing (which again is equivalent to shortest expected workload routing) is optimal for a variety of models and objective functions (e.g., Winston, 1977, Weber, 1978, Menich, 1987, Walrand, 1988, Hordijk and Koole, 1990, Sparaggis, Towsley, and Cassandras, 1993, and Towsley, Sparaggis, and Cassandras, 1993). Whitt (1986) gave a counterexample showing that shortest queue routing is not optimal for general processing time distributions.

If anticipative information (on processing times for future arrivals) is not permitted, the centralized controller has the most possible information when it knows the processing times of all tasks upon arrival. In this case, routing tasks to the processor with the least work, which is equivalent to FCFS with a shared queue, has been shown to be optimal for various objectives (Kingman, 1970, Vasicek, 1977, Foss, 1980 and 1981, Wolff, 1977 and 1987, Daley, 1987, Liu and Towsley, 1994b and 1994c).

When processors are not homogeneous the problem is considerably more difficult. See Combé and Boxma (1993) for a good discussion of the problem and solution approaches, and review of the literature.

For our model, with distributed control and unreliable homogeneous processors that know only earlier decisions and their own general task arrival times, we show that the round-robin policy, in which each processor sends general tasks to each of the other processors in turn, in a cyclic fashion, is optimal. By optimal, we mean that round-robin *stochastically* minimizes the n^{th} task completion time for all n, and minimizes response times and queue lengths in a separable increasing convex

sense, among all policies that balance workload. We also show that if there is a single centralized controller, round-robin is the optimal policy, and a single controller using round-robin routing is better than the optimal distributed system, where "optimal" and "better" are in the sense of stochastically minimizing task completion times and minimizing response times and queue lengths in the separable increasing convex sense. We assume processing times are *iid* (independent and identically distributed), but may have an arbitrary distribution. The arrival process of outside tasks to each processor may be arbitrary as long as it is independent of the state of the system. Processors may fail, with arbitrary failure and repair processes that are also independent of the state of the system.

In section 2 we recall some definitions for stochastic orderings and majorization, and in section 3 we review some properties of general single-server queues and prove a useful result for the round-robin policy. In section 4, we prove that for a centralized controller and perfectly reliable processors the round-robin policy is optimal. We extend this result to distributed control with unreliable processors and discuss other extensions in section 5.

2 Preliminaries

Let us first give some definitions and useful facts for stochastic orderings and majorization. See, for example, Marshall and Olkin (1979).

Throughout we will use smaller, before, earlier, increasing, etc. in the nonstrict sense.

Recall that for two random variables X and Y, X is stochastically larger than Y, $X \geq_{st} Y$, if and only if $E(g(X) \geq E(g(Y))$ for all increasing functions $g(t)$, provided the expectations exist. Also, $X \geq_{st} Y$ if and only if it is possible to construct two coupled random variables, that is, two random variables on the same probability space, $\hat{X}$ and $\hat{Y}$, so that $\hat{X} =_{st} X$ and $\hat{Y} =_{st} Y$ and $\hat{X} \geq \hat{Y}$ with probability 1.

For two m dimensional random vectors, $S = (S_1, ..., S_m)$ and $T = (T_1, ..., T_m)$, we say that S is larger than T in the separable stochastic (sep-st) sense if $E(\sum_{i=1}^{m} g_i(S_{[i]})) \geq E(\sum_{i=1}^{m} g_i(T_{[i]}))$ for any increasing functions $g_i(t)$, $i = 1, \ldots m$, provided the expectations exist, where $x_{[i]}$ indicates the i^{th} largest component of x. This is equivalent to $S_{[i]} \geq_{st} T_{[i]}$ for all $i = 1, \ldots m$. We will show that round-robin minimizes departure

times in the separable stochastic sense, which means that the n^{th} departure is stochastically earlier under round-robin than under any other policy for all n. Note that this is weaker than saying the entire departure process, i.e., all departures *jointly*, are stochastically minimized.

Lemma 2.1 *S is larger than T in the sep-st sense if we can construct two random vectors on the same probability space, $\hat{S}$ and $\hat{T}$, so that $\hat{S}_{[i]} =_{st} S_{[i]}$ and $\hat{T}_{[i]} =_{st} T_{[i]}$ and $\hat{S}_{[i]} \geq \hat{T}_{[i]}$ with probability 1, $i = 1, ..., m$.*

We say that S is larger than T in the separable increasing convex (sep-icx) sense if $E(\sum_{i=1}^{m} g(S_i)) \geq E(\sum_{i=1}^{m} g(T_i))$ for any increasing and convex function $g(t)$, provided the expectations exist. This is also written as $S \geq_{E_3^{\uparrow}} T$ (Marshall and Olkin, 1979).

For two m dimensional real vectors x and y, x weakly majorizes y, $x \succ_w y$, if $\sum_{i=1}^{k} x_{[i]} \geq \sum_{i=1}^{k} y_{[i]}$, for $k = 1, \ldots m$.

Lemma 2.2 *$x \succ_w y$ if and only if $\sum_{i=1}^{m} g(x_i) \geq \sum_{i=1}^{m} g(x_i)$ for any increasing and convex function $g(t)$.*

Corollary 2.3 *S is larger than T in the sep-icx sense if we can construct two random vectors on the same probability space, $\hat{S}$ and $\hat{T}$, so that $\hat{S}_i =_{st} S_i$ and $\hat{T}_i =_{st} T_i$, $i = 1, ..., m$ and $\hat{S} \succ_w \hat{T}$ with probability 1.*

In the following $x^1$and y^1 ($x^2$and y^2) are two m (n) dimensional real vectors, and (x^1, x^2) and (y^1, y^2) are the $m + n$ dimensional concatenations of x^1 and x^2, and y^1 and y^2, respectively.

Lemma 2.4 *If $x^1 \succ_w y^1$ and $x^2 \succ_w y^2$, then $(x^1, x^2) \succ_w (y^1, y^2)$.*

3 Properties of single-server queues and the round-robin policy

We now give some properties of $G/GI/1$ queues and of the round-robin (RR) policy that will be useful in the sequel.

Lemma 3.1 *Consider a $G/GI/1$ queue with initial workload W and arrival times T_n, service times S_n, and completion or departure times C_n for the n^{th} task. Let $W(t)$ be the workload and $Q(t)$ be the queue length (including the task in service) at time t. Then*

(C) $C_n = \max\{T_n, \max_{1\le m\le n-1} \sum_{i=m}^{n-1} S_i + T_m, W + \sum_{i=1}^{n-1} S_i\} + S_n.$

Also, if there have been n arrivals to a $G/GI/1$ queue by time t,

(W) $W(t) = \max\{0, \max_{1\le m\le n} \sum_{i=m}^{n} S_i + T_m - t, W + \sum_{i=1}^{n} S_i - t\},$

(Q) $Q(t) = \sum_{i=1}^{n} \mathbf{I}(W(t) - \sum_{j=i+1}^{n} S_i > 0),$

where $\mathbf{I}(\cdot)$ is the indicator function. In particular, C_n and $W(t)$ are increasing in W and T_m, $1 \le m \le n$, and $Q(t)$ is increasing in $W(t)$.

Lemma 3.2 *Consider two $G/GI/1$ queues that are identical except for their arrival times. Let T_n^i be the n^{th} arrival time and let $W^i(t)$ be the work at time t for queue i, where work is assumed to be right continuous. Let S be a generic service time. Suppose that the arrival times are ordered so that $0 = T_1^1 \le T_1^2 \le T_2^1 \le T_2^2 \le \cdots \le T_n^1 \le T_n^2 \le T_{n+1}^1 \cdots$. Then*

(a) If $T_n^1 \le t < T_n^2$, then $W^2(t) \le_{st} W^1(t) \le_{st} W^2(t) + S$.

(b) If $T_n^2 \le t < T_{n+1}^1$, then $W^1(t) \le_{st} W^2(t) \le_{st} W^1(t) + S$.

Proof. Our proof is by induction on n. Assume (a) and (b) hold for $n-1$, and consider n. (Properties (a) and (b) hold trivially for $n = 0$, with $T_0^i = 0$ for $i = 1, 2$.)

Let S_n be the n^{th} service time for both systems. Choose an arbitrary t such that $T_n^1 \leq t < T_n^2$. Then, using the induction hypothesis for (b) at $t = T_{n-1}^2$,

$$\begin{aligned} W^1(t) &= [[W^1(T_{n-1}^2) - (T_n^1 - T_{n-1}^2)]^+ + S_n - (t - T_n^1)]^+ \\ &\leq [[W^1(T_{n-1}^2) - (T_n^1) - T_{n-1}^2)]^+ - (t - T_n^1)]^+ + S_n \\ &\leq [[W^2(T_{n-1}^2) - (T_n^1 - T_{n-1}^2)]^+ - (t - T_n^1)]^+ + S_n \\ &= W^2(t) + S_n \end{aligned}$$

and

$$\begin{aligned} W^1(t) &= [[W^1(T_{n-1}^2) - (T_n^1 - T_{n-1}^2)]^+ + S_n - (t - T_n^1)]^+ \\ &= [\max\{S_n, W^1(T_{n-1}^2) + S_n - (T_n^1 - T_{n-1}^2)\} - (t - T_n^1)]^+ \\ &\geq [\max\{0, W^1(T_{n-1}^2) + S_n - (T_n^1 - T_{n-1}^2)\} - (t - T_n^1)]^+ \\ &\geq [\max\{0, W^2(T_{n-1}^2) - (T_n^1 - T_{n-1}^2)\} - (t - T_n^1)]^+ \\ &= W^2(t) \end{aligned}$$

where $[x]^+ = \max\{0, x\}$. We thus have (a), and (b) follows in a similar way from (a). □

Corollary 3.3 *Assume we have identical queues and the initial workloads are stochastically identical. Then under the round-robin policy, at each arrival epoch the workloads at all the queues will be stochastically ordered, and round-robin will route to the queue with the stochastically smallest workload.*

Proof. Let $W = (W^1, W^2, \ldots, W^K)$ be the initial workload in the queues. Since our result only concerns the marginal distributions of workloads, and since the round-robin policy is unaffected by actual workloads, we may assume, without loss of generality, that $W^1 = W^2 = \cdots = W^K$ with probability 1. Assume, again without loss of generality, that RR routes to the queues in the order $1, 2, \ldots K, 1, 2, \ldots$, and that it is about to route task n to queue l, $1 \leq l \leq K$, after having completed m full cycles, so $n = mK + l$. Let $V = (V^1, V^2, \ldots, V^K)$ be the workload just before task n's arrival. Then from lemma 3.2(b) $V^1 \leq_{st} V^2 \leq_{st} \cdots \leq_{st} V^{l-1}$ and $V^l \leq_{st} V^{l+1} \leq_{st} \cdots \leq_{st} V^K$, and from lemma 3.2(a) $V^K \leq_{st} V^1$, so the result follows. □

4 Centralized control

We have K homogeneous processors, and for each processor the service discipline is FCFS (first-come-first-served) and there is an infinite waiting buffer. Processing times are *iid*. In this section we assume processors do not fail. Upon arrival a task must be routed to one of the processors by the centralized controller, and the only information available is the history of prior routing decisions and arrival times. Thus, our admissible policies are effectively open-loop policies, in which all decisions are made in advance. We assume the system is initially empty, or, more generally, that the initial work at all the processors is stochastically identical. We show, with the help of several lemmas, that the round-robin policy (RR) minimizes response times and queue lengths in the separable increasing convex sense, where a task's response time is the difference between its completion time and its arrival time. We also show that completion times are minimized in the separable stochastic sense by round-robin.

Let $R = (R_1, R_2, \ldots, R_N)$ be the response times, and let $C = (C_1, C_2, \ldots, C_N)$ be the departure or completion times, of the first N tasks under an arbitrary policy, where tasks are labeled in the order of their arrival. Let $Q(t) = (Q_1(t), Q_2(t), \ldots, Q_K(t))$ be the queue lengths, including the task in service, at time t under an arbitrary policy, assuming there will only be N arrivals to the system. We use the superscript RR to denote the same quantities under the round-robin policy.

Theorem 4.1 *For all N,*

$$C^{RR} \leq_{sep-st} C, \text{ and } R^{RR} \leq_{sep-icx} R,$$

and for all t,

$$Q^{RR}(t) \leq_{sep-icx} Q(t).$$

Proof. Fix N. Let $\pi^{(k)}$, $k = 0, ..., N$, be a policy that follows RR for the first k routing decisions, and follows an arbitrary policy thereafter. We show that for any such policy $\pi^{(k)}$, there exists a policy $\pi^{(k+1)}$ that follows RR for the first $k+1$ routing decisions, such that the response times, queue lengths, and departure times are smaller in the appropriate sense under $\pi^{(k+1)}$ than under $\pi^{(k)}$. Our result will then follow since RR is $\pi^{(N)}$, and an arbitrary policy can be labeled $\pi^{(0)}$.

For all our policies we will assume the sequence of arrival times is the same. We also couple processing times so that the n^{th} processing time for each processor is the

same for all processors and all policies. That is, letting $S_n(k,\rho)$ be the processing time of the n^{th} task to be served at processor k under policy ρ, we have $S_n(k,\rho) \equiv S_n$ for all k and ρ. Under this coupling each processor will have the correct *marginal*, though not joint, distributions, which is sufficient since our objective is to minimize completion times, response times, and queue lengths in the *separable* increasing convex or stochastic sense. Also, since our decisions cannot depend on the states of the processors, our policies are unaffected by this coupling. A similar coupling was also used by Ephremides et al. (1980) and Liu and Towsley (1994).

Let W_i be the work at processor i just before the $k+1^{st}$ decision under the policies $\pi^{(k)}$ and $\pi^{(k+1)}$. From lemma 3.3, these workloads can be stochastically ordered. Without loss of generality, assume $W_1 \leq_{st} W_i$, $i = 2, ..., K$. If $\pi^{(k)}$ routes the $k+1^{st}$ arrival to processor 1, then by letting $\pi^{(k+1)}$ agree with $\pi^{(k)}$ for the remaining decisions, we are done, again from corollary 3.3. Therefore assume that $\pi^{(k)}$ routes the $k+1^{st}$ arrival to some other processor, call it processor 2 without loss of generality, and suppose that we construct W_1 and W_2 on the same probability space so that $W_1 \leq W_2$ with probability 1. Of course $\pi^{(k+1)}$ routes the $k+1^{st}$ arrival to processor 1.

Algorithm for constructing π'_R

Set $C = 0$;

DO for $i = l+1$ to N;

Let r be the i^{th} routing decision of π_R;

If $r = 1$ then $C \leftarrow C + 1$;

Else if $r = 2$ then $C \leftarrow C - 1$;

If $r = 1$ or 2 then

If $C > 0$ then $r' = r$;

Else if $C < 0$ then $r' = inv(r)$;

Else if $C = 0$ then $r' = 2$;

Else $r' = r$;

END DO LOOP;

Now $\pi^{(k+1)}$ is fully specified. The response times and departure times of tasks routed to processors other than processors 1 and 2 are the same under both policies $\pi^{(k)}$ and $\pi^{(k+1)}$, as are the queue lengths for the other processors. In lemma 4.3 and corollary 4.4 we show that for the tasks in each chunk, the response times under π'_R are weakly majorized by those under π_R, and the departures are earlier. We also show that during each chunk the queue lengths under π'_R are weakly majorized by those under π_R. These results follow from the fact that at the beginning of each chunk the workloads at processors 1 and 2 under the two policies are ordered in a strong sense (lemma 4.2). The result then follows from lemmas 2.1 and 2.4 and corollary 2.3. □

In the following, $\pi^{(k)}$, $\pi^{(k+1)}$, π_R, π'_R, processors 1 and 2, and W_1 and W_2 are as defined in the proof of theorem 4.1. Services at both processors under both policies are coupled as in that proof, and $W_1 \leq W_2$ with probability 1. By "complete chunk", we mean that the chunk ends because $C = 0$ in the algorithm.

We first show, in lemma 4.2, that workloads at processors 1 and 2 under π_R and π'_R at the end of each chunk are ordered in a sense that is stronger than the weak majorization sense defined earlier. We then show that response times and queue lengths are also ordered in a strong sense, from which the ordering in the sep-icx sense follows from corollary 2.3. We also show that departure times are ordered with probability 1.

Lemma 4.2 *Let U_i (U'_i) be the workload at processor i under policy $\pi^{(k)}$ ($\pi^{(k+1)}$) just after the last task in a chunk is routed. Then for each complete chunk, with probability 1,*

$$U'_1 \leq \min(U_1, U_2), U'_2 = \max(U_1, U_2).$$

Proof. Our proof is by induction on the chunks. The result for the "zero'th chunk" follows from the fact that $W'_1 = W_1 \leq W_2 = W'_2$. Let V_i (V'_i) be the workload at processor i under policy $\pi^{(k)}$ ($\pi^{(k+1)}$) just after the last task in the previous chunk is routed, so, by the induction hypothesis,

$$V'_1 \leq \min(V_1, V_2),\ V'_2 = \max(V_1, V_2).$$

Assume for the moment that for the current chunk the algorithm is such that $\pi'_R = \pi_R$. Then within the chunk, the following is guaranteed by our algorithm:

(a) The n^{th} arrival to processor 1 occurs before the n^{th} arrival to processor 2 under both policies π_R and π'_R for all arrivals n during the chunk, and

(b) An equal number of arrivals are sent to both processors under both policies.

We have that $U'_1 \leq U_1$ from lemma 3.1 (W), since by the induction hypothesis $V'_1 \leq V_1$. We also have $U'_1 \leq U_2$ from lemma 3.1 (W) because of property (a) and $V'_1 \leq V_2$. Therefore, $U'_1 \leq \min(U_1, U_2)$.

We consider the following cases to show $U'_2 = \max(U_1, U_2)$.

Case I: $V'_2 = V_2$ (i.e., $V'_1 \leq V_1 \leq V'_2 = V_2$). In this case processor 2 is identical under both policies during the chunk, so $U'_2 = U_2$. Moreover, the relation $V_2 \geq V_1$, together with property (a) and lemma 3.1 imply that $U_2 \geq U_1$. Therefore, $U'_2 = \max(U_1, U_2)$.

Case II: $V'_2 = V_1$ (i.e., $V'_1 \leq V_2 \leq V'_2 = V_1$). Here we consider two subcases depending on whether queue 2, that is the queue at processor 2, empties at some time during the chunk or not under policy π'_R. If it empties, then so does queue 2 under policy π_R by lemma 3.1 (W) for work, and from the point that it empties under policy π'_R the workload at processor 2 will be identical under the two policies, so $U'_2 = U_2$. Also, $U'_2 \geq U_1$, from property (a) and lemma 3.1 (W), since $V'_2 = V_1$. Again, $U'_2 = \max(U_1, U_2)$.

If queue 2 under policy π'_R never empties during the chunk, then neither will queue 1 under policy π_R because $V_1 = V'_2$, the arrivals to queue 1 occur earlier than the arrivals to queue 2 (property (a)) and there are the same number of arrivals to both chunks (b). Hence, at the end of the chunk the work will still be the same,

$$U'_2 = V'_2 + \sum S_i - T = V_1 + \sum S_i - T = U_1 \geq V_2 + \sum S_i - T = U_2,$$

where T is the total time for the chunk, and $\sum S_i$ is the sum of the processing times for the arrivals during the chunks (which are coupled for both policies and both processors). Hence, $U'_2 = \max(U_1, U_2)$ again.

Now let us suppose that π'_R switches the routings to processors 1 and 2 relative to π_R for this chunk, that is, that $C \leq 0$ in the algorithm during the chunk. In this case π_R routes a task to processor 2 before routing one to processor 1, and the reverse is true for π'_R. Let us relabel the processors under π_R for this chunk so that processor 2 will now be called processor 1 and vice-versa. Then, π_R and π'_R

agree, and properties (a) and (b) again hold, so the result follows from the previous argument. □

Now we are ready to consider response times, departure times, and queue lengths. Choose an arbitrary complete chunk of routing decisions. Let σ be the time just after the last task of the last chunk is routed, and let τ be the time just after the last task of the current chunk is routed. In the following, by n^{th} arrival to a processor we mean the n^{th} task to arrive to that processor during the chunk. Let n be arbitrarily fixed. Let R_i (R_i') be the response time and C_i (C_i') be the completion or departure time of the n^{th} arrival to processor i, $i = 1, 2$, under policy π_R (π_R') during the chunk, where the dependency on n is suppressed. Since from property (b) in lemma 4.2 there are the same number of arrivals to each processor in each complete chunk under both policies, the pairs (R_1, R_2) and (C_1, C_2) are well defined within each complete chunk. Let $Q_i(t)$ ($Q_i'(t)$) be the queue lengths under policy π_R (π_R') for processor i at time t.

Lemma 4.3 *For all n, with probability 1,*

$$C_1' \le \min(C_1, C_2), \text{ and } C_2' \le \max(C_1, C_2),$$

$$\max(R_1', R_2') \le \max(R_1, R_2), \text{ and } R_1' + R_2' \le R_1 + R_2,$$

and for $t \le \tau$,

$$Q_1'(t) \le Q_1(t),\ Q_2'(t) \le \max(Q_1(t), Q_2(t)), \text{ and } Q_1'(t) + Q_2'(t) \le Q_1(t) + Q_2(t).$$

Therefore $C_{[1]}' \le C_{[1]}$, $C_{[2]}' \le C_{[2]}$, $(R_1', R_2') \prec_w (R_1, R_2)$, and $(Q_1'(t), Q_2'(t)) \prec_w (Q_1(t), Q_2(t))$.

Proof. We will show the results for the completion times and the first two results for queue lengths first. The remaining results will follow from the ordering of the departures.

Suppose for the moment that for the current chunk $\pi_R' = \pi_R$. Let U_i (U_i') be the workload at processor i under policy $\pi^{(k)}$ ($\pi^{(k+1)}$) at time σ, so, by lemma 4.2,

$$U_1' \le \min(U_1, U_2),\ U_2' = \max(U_1, U_2).$$

Then $C_1' \le C_1$ by lemma 3.1 (C), since the arrival times for both policies at processor 1 are the same.

We have that $Q'_1(t) \leq Q_1(t)$ for $t = \sigma$ from lemma 3.1 (Q) and the fact that $U'_1 \leq U_1$. We therefore also have $Q'_1(t) \leq Q_1(t)$ for $t \leq \tau$ since the arrival times at processor 1 are the same under both policies, and we have just shown that departures are earlier under $\pi^{(k+1)}$ than under $\pi^{(k)}$.

Since $\pi'_R = \pi_R$ during the chunk, properties (a) and (b) of lemma 4.2, are again guaranteed by our algorithm. From property (a) and the fact that $U'_1 \leq U_2$, we have that $C'_1 \leq C_2$, from lemma 3.1 (C), and therefore $C'_1 \leq \min(C_1, C_2)$.

We consider the same cases as in lemma 4.2 to show the remainder of the results.

Case I: $U'_2 = U_2$ (i.e., $U'_1 \leq U_1 \leq U'_2 = U_2$). In this case queue 2 is identical under both policies during the chunk, so $C'_2 = C_2$ and $Q'_2(t) \leq Q_2(t)$ for $t \leq \tau$ from lemma 3.1.

Case II: $U'_2 = U_1$ (i.e., $U'_1 \leq U_2 \leq U'_2 = U_1$). Let $\hat{\tau}$ be the time just after the $n^t h$ arrival to processor 2. (If the $n^t h$ arrival is the last arrival, $\hat{\tau} = \tau$.) Again we consider two subcases depending on whether queue 2 under policy π'_R empties at some time during the chunk before time $\hat{\tau}$ or not. If it empties, then so does queue 2 under policy π_R by lemma 3.1 (W), since $U_2 \leq U'_2$. From the point that it empties under policy π'_R queue 2 will be identical under the two policies, so $C'_2 = C_2$, and $Q'_2(t) \leq Q_2(t)$ for $t \leq \hat{\tau}$.

If queue 2 under policy π'_R never empties during the chunk before the n^{th} arrival to queue 2, then neither will queue 1 empty before the n^{th} arrival to queue 1 under policy π_R because $U_1 = U'_2$, and the arrivals to queue 1 occur earlier than the arrivals to queue 2 from property (a). Therefore, $C'_2 = C_1$ and $Q'_2(t) \leq Q_1(t)$ for $t \leq \hat{\tau}$, from lemma 3.1.

Note that since we have shown that for all n the n^{th} departures from processors 1 and 2 occur earlier under π'_R than under π_R in all cases, i.e., $\min(C'_1, C'_2) \leq \min(C_1, C_2)$ and $\max(C'_1, C'_2) \leq \max(C_1, C_2)$, we have that the total number in the two queues must be smaller under π'_R than under π_R, i.e., $Q'_1(t) + Q'_2(t) \leq Q_1(t) + Q_2(t)$, for $t \leq \tau$.

Now let us consider the response times. Fix n. We have that

$$R'_1 = C'_1 - T_1,\ R'_2 = C'_2 - T_2,$$

$$R_1 = C_1 - T_1,\ R_2 = C_2 - T_2,$$

where T_i is the arrival time of the n^{th} arrival to processor i, $i = 1, 2$. Then $R_1' \leq R_1$ since $C_1' \leq C_1$, and $R_2' \leq \max(R_1, R_2)$ since $C_2' \leq \max(C_1, C_2)$ and $T_1 \leq T_2$ (property (a)), so $\max(R_1', R_2') \leq \max(R_1, R_2)$. Also $R_1' + R_2' \leq R_1 + R_2$, because $C_1' \leq \min(C_1, C_2)$, and $C_2' \leq \max(C_1, C_2)$.

By reversing the roles of processors 1 and 2 under π_R we also have the same results when π_R' switches the routings to processors 1 and 2 relative to π_R. □

Note that from lemmas 4.3 and 2.4 it follows that for all tasks within a complete chunk, the response times under π_R' are weakly majorized by those under π_R, and the departures are earlier.

Finally, let us consider the very last, possibly incomplete, chunk. Suppose for the moment that $\pi_R' = \pi_R$ for that chunk. Again let σ be the time just after the last task of the last chunk is routed, and let τ be the time just after the last task of the current chunk is routed. Let l (m) be the number of tasks routed to processor 1 (2) under policy π_R', and let ρ be the time of the m^{th} arrival to processor 1. By the structure of the algorithm for constructing π_R' in chunks, $l \geq m$, and $\rho \leq \tau$. Therefore, we can match up the first $m$arrivals to the two processors under the two policies, and for those arrivals lemma 4.3 still holds for response times and departure times, and it holds for queue lengths for $t \leq \rho$. The remainder of the arrivals will be routed to processor 1, and arguing as in lemma 4.3, we have for each of those tasks $C_1' \leq C_1$by lemma 3.1, since the arrival times for both policies at processor 1 are the same. We therefore also have $Q_1'(t) \leq Q_1(t)$for $t \leq \tau$. Also, since $Q_2'(t) \leq \max(Q_1(t), Q_2(t))$for $t \leq \rho$, this will remain true for $t \leq \tau$because after time ρthere will only be arrivals to processor 1 under both policies. Finally, since departures from processors 1 and 2 occur earlier under π_R'than under π_R, we have that the total number at the two processors must be smaller under π_R'than under π_R, i.e., $Q_1'(t) + Q_2'(t) \leq Q_1(t) + Q_2(t)$, for $t \leq \tau$. The results for response times follow from the results for completion times as before. If π_R'switches the routings to processors 1 and 2 relative to π_R, we get the corresponding result by reversing the roles of processors 1 and 2 under π_R.

Corollary 4.4 *For the very last, possibly incomplete, chunk, for $n \leq m$,*

$$C_1' \leq \min(C_1, C_2), \text{ and } C_2' \leq \max(C_1, C_2),$$

$$\max(R_1', R_2') \leq \max(R_1, R_2), \text{ and } R_1' + R_2' \leq R_1 + R_2,$$

for $m \leq n \leq l$,

$$C'_1 \leq C_1, \text{ and } R'_1 \leq R_1,$$

and for $t \leq \tau$,

$$Q'_1(t) \leq Q_2(t),\ Q'_2(t) \leq \max(Q_1(t), Q_2(t)),\ \text{and } Q'_1(t) + Q'_2(t) \leq Q_1(t) + Q_2(t).$$

Note that from corollary 4.4 and lemma 2.4 it follows that for all tasks within the last chunk, the response times under π'_R are weakly majorized by those under π_R, and the departures are earlier.

5 Distributed control with unreliable processors

Let us now consider the more general model with unreliable processors and distributed control. We will discuss several increasingly more general models in turn.

5.1 Unreliable processors

First consider the central controller model in which processors may fail. Failure and repair processes for different processors are *iid*, and these processes are independent of the number in queue. In particular, a processor may fail, or go on vacation, even when there is no task in its queue. Then, by coupling the failure and repair times, as we did for the processing times, so that they are identical for all processors, all the arguments go through with only minor modifications to lemma 3.1. In particular, it will still be true that C_n and $W(t)$ are increasing in W and T_m, $1 \leq m \leq n$, and $Q(t)$ is increasing in $W(t)$. If processors only fail when there are tasks present, our result will still hold, but we must assume that the time to fail has an exponential distribution. In this case each task has an "effective processing time" which is the time between its start of processing and its completion, and which may include some processor down time. Again we can couple effective processing times so they are identical for the n^{th} service at all processors, and the result follows.

5.2 Multiple arrival processes

Now let us suppose we have a central controller that can route arrivals from one arrival process, but in addition each processor has its own local arrival process of

tasks, and processors process all tasks in FCFS order. The local arrival processes are stochastically identical for all processors, and all arrival processes are independent of the states of the system. We couple the local arrival processes for all processors, so that all local arrival processes are identical. Then 3.3 still holds for round-robin routing by the central controller, where it again follows from 3.2 using the combined arrival process at each processor. In lemma 4.2 properties (a) and (b) again hold for the combined arrival process, and the argument follows as before to show that the round-robin policy at the central controller stochastically minimizes task completion times and minimizes response times and queue lengths in the separable increasing convex sense.

5.3 Distributed control

Now we consider the model with distributed control. Each processor receives its own arrival process of tasks from outside users, some of which it redirects to the other processors. Processing times are assumed to be *iid* (independent and identically distributed), but may have an arbitrary distribution. The arrival process of outside tasks to each processor may be arbitrary as long as it is independent of the state of the system. Processors may fail, with arbitrary failure and repair processes that are also independent of the state of the system. The processors are homogeneous in the sense that the distribution of their processing times are identical, and the failure, repair, and arrival processes for each processor are stochastically the same. Upon arrival of a general, or redirectable, task from an outside user, the processor must decide which processor to send it to. (If a processor is permitted to send general tasks to itself, the arrival processes of general tasks do not have to be stochastically the same for all processors.) General tasks received by other processors and local tasks must be accepted by the processor. (Task migration can occur at most once.) Accepted tasks are processed according to the FCFS discipline. The only information available to a processor is the history of its decisions for routing work to other processors, and the arrival times for its own general arrival process.

First notice that since a centralized controller can implement any distributed policy, a centralized controller using round-robin routing is better than the optimal distributed system in the sense of stochastically minimizing task completion times and minimizing response times and queue lengths in the separable increasing convex sense.

We assume that for the distributed control system admissable policies for general tasks are those which balance the workload in the long run, and use only information on the history of routing decisions and local arrival times. Therefore admissable policies include random routing with identical probabilities and round-robin routing. However, they also include cyclic rounding where the order within a cycle can vary within each cycle, round-robin where m consecutive arrivals are routed to each processor, and more complicated variations. (If processors may send general arrivals to themselves, then in implementing these policies processors include themselves.) Now consider one processor and assume the other processors are following some arbitrary admissable policy. Then from that processor's point of view, it is a central controller with multiple arrival processes. Therefore, from the result in section 5.2, we have that the optimal policy for the given processor is round-robin. Repeating the argument for each processor in turn, we have that following the round-robin policy at each processor minimizes the n^{th} task completion time for all n, and minimizes response times and queue lengths in a separable increasing convex sense, among all distributed policies that balance workload.

5.4 Polling system

Another model in which our results hold is a polling system in which a single processor serves several queues. The processor is available at each queue (and serving tasks if any are present) for a random time that is *iid* for each visit to each queue, and it visits queues either in a Bernoulli or cyclic fashion, where the first queue to visit is chosen at random. By Bernoulli visits we mean that for each visit the processor randomly chooses a new queue to visit such that each queue is equally likely to be chosen each time, and the choices are independent from visit to visit. Indeed, any visit strategy that is stochastically identical for all queues and is independent of the state of the system is permissible. There may be *iid* random switchover times. Our argument follows again by coupling the visit and intervisit times so that they are identical for all queues. That is, in our new system, the processor visits all queues at the same time and is away from all queues at the same time for all policies. This is obviously very different from the original system, but again, since our objective functions are separable, and since our admissible policies cannot depend on the states of the queues, the systems are equivalent for each queue *marginally*. Our system is therefore equivalent to the original centralized control system with processor failures. Thus, the round-robin policy for routing arrivals to queues will again be optimal in our separable sense for this polling system.

6 Acknowledgment

We would like to thank Ger Koole for helpful discussions.

References

[1] R. K. Boel, J. H. van Schuppen, "Distributed routing for load balancing", *Proc. IEEE*, **77**, 210-221.

[2] C. S. Chang, "A New Ordering for Stochastic Majorization: Theory and Applications", *Advances in Applied Probability*, **24**, pp. 604-634.

[3] C. S. Chang, X. L. Chao, M. Pinedo, "A Note on Queues with Bernoulli Routing", *Proc. 29th Conf. on Decision and Control*, Hawaii, December 1990.

[4] M. B. Combé, O. J. Boxma, "Optimization of Static Traffic Allocation Policies", *Proceedings of the Workshop on Formalisms, Principles, and State-of-the-Art*, Götz, Herzog, and Rettelbach, eds., Institut für Mathematische Maschinen und Datenverarbeitung (Informatik), Erlangen, 1993.

[5] D. J. Daley, "Certain Optimality Properties of the First Come First Served Discipline for $G/G/s$ Queues", *Stochastic Processes and their Applications*, **25**, pp. 301-308, 1987.

[6] A. Ephremides, P. Varaiya and J. Walrand, "A simple dynamic routing problem", *IEEE Trans. on Aut. Control*, **AC-25**, pp. 690-693, 1980.

[7] S. G. Foss, "Approximation of Multichannel Queueing Systems" (in Russian), *Sibirski Mat. Zh.*, Vol. 21, pp. 132-140, 1980. (Transl.: *Siberian Math. J.*, **21**, pp. 851-857, 1980.)

[8] S. G. Foss, "Comparison of Servicing Strategies in Multichannel Queueing Systems" (in Russian), *Sibirski Math. Zh.*, Vol. 22, pp. 190-197, 1981. (Transl.: *Siberian Math. J.*, **22**, pp. 142-147, 1981.)

[9] E. Gelenbe, *Multiprocessor Performance.* Wiley, New York, 1989.

[10] E. Gelenbe, F. Pekergin "Load balancing pragmatics." Technical report, EHEI Université René Descartes, February, 1993.

[11] L. Gün, A. Jean-Marie, "Parallel Queues with Resequencing", *J. ACM.* **40**, pp. 1188-1208, 1993.

[12] A. Hordijk, G. Koole, "On the optimality of the generalized shortest queue policy", *Prob. in the Engin. and Info. Sciences*, **4**, pp. 477-487, 1990.

[13] A. Jean-Marie, Z. Liu, "Stochastic Comparisons for Queueing Models via Random Sums and Intervals", *Adv. Appl. Prob.*, **24**, pp. 960-985, 1992.

[14] G. Koole, "On the pathwise optimal Bernoulli routing policy for homogeneous parallel servers", preprint, 1994.

[15] Z. Liu, D. Towsley, "Optimality of the Round-Robin Routing Policy", *J. Appl. Prob.* **31**, pp. 466-478, 1994a.

[16] Z. Liu, D. Towsley, "Effects of Service Disciplines in $G/G/s$ Queueing Systems", *Ann. Opns. Res.* **48**, special issue on Queueing Networks. To appear, 1994b.

[17] Z. Liu, D. Towsley, "Stochastic Scheduling in In-Forest Networks", *Adv. Appl. Prob.* **26**, pp. 222-241, 1994c.

[18] A. W. Marshall, I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, Academic Press, 1979.

[19] R. Menich, "Optimality of Shortest Queue Routing for Dependent Service Stations", *Proc. 26th Conf. on Decision and Control*, pp. 1069-1072, 1987.

[20] P.D. Sparaggis, D. Towsley, C.G. Cassandras, "Extremal properties of the SNQ and the LNQ policies in finite capacity systems with state-dependent service rates," *J. Appl. Prob.* **30**, pp. 223-236, 1993.

[21] D. Stoyan, *Comparison Methods for Queues and Other Stochastic Models.* English translation (D.J. Daley editor), J.Wiley and Sons, New York, 1983.

[22] D. Towsley, P.D. Sparaggis and C.G. Cassandras, 'Optimal routing and buffer allocation for a class of finite capacity queueing systems," IEEE Trans. Autom. Control. **37,** pp. 1446-1451, 1993.

[23] O. A. Vasicek, "An Inequality for the Variance of Waiting Time Under a General Queueing Discipline." *Opns. Res.*, **25**, pp. 879-884, 1977.

[24] Y.-T. Wang, R. J. T. Morris, "Load sharing in distributed systems," *IEEE Trans. Comp.*, **C-34**, 204-217.

[25] J. Walrand, *An Introduction to Queueing Networks.* Prentice Hall, 1988.

[26] R. R. Weber, "On the optimal assignment of customers to parallel queues", *J. Appl. Prob.*, **15**, pp. 406-413, 1978.

[27] W. Whitt, "Deciding Which Queue to Join: Some Counterexamples." *Opns. Res.*, Vol. 34, No. 1, pp. 55-62, 1986.

[28] W. Winston, "Optimality of the shortest line discipline", *J. Appl. Prob.*, **14**, pp. 181-189, 1977.

[29] R. W. Wolff, "An Upper Bound for Multi-Channel Queues," *J. Appl. Prob.*, Vol. 14, pp. 884-888, 1977.

[30] R. W. Wolff, "Upper Bounds on Work in System for Multichannel Queues," *J. Appl. Prob.*, Vol. 24, pp. 547-551, 1987.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399