Economides, A.A. & Silvester, J.A.: **Priority load sharing: an approach using Stackelberg games.**
Proceedings 28th Annual Allerton Conference on Communications, Control and Computing, 1990.

PRIORITY LOAD SHARING: AN APPROACH USING STACKELBERG GAMES

A. A. ECONOMIDES and J. A. SILVESTER

Electrical Engineering - Systems Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract

In this paper we consider the load sharing problem in a multiprocessor, where different classes of jobs have different priorities and each priority class optimizes its own objective function. First, we formulate the problem as a leader-follower Stackelberg game, where the high priority jobs constitute the leader and the low priority jobs constitute the follower. Then we focus on a special case of two preemptive resume priority classes that share a two-processor system. In this case, we find the Stackelberg equilibrium solution that minimizes the average job delay of each class. An interesting result is that if both classes have equal mean service requirements, when both processors are used, then the low priority load sharing decisions are independent of the arrival rates. Also, for equal mean service requirements and constant total arrival rate, the overall average delay of jobs from both classes is constant, i.e. does not depend on the mix of high and low priority jobs. Finally, we comment on our approach for multiobjective optimization of distributed systems with multi-priority classes.

1. INTRODUCTION

The usual approach to distributed system design and control is the optimization of a single function. If multiple objectives are desired, then the usual approach is to combine the objectives as seen by the system administrator [2] into a single function. Thus, it is assumed that all customers in the system are treated similarly and they cooperate for the socially optimum, such as optimizing the average customer performance. However, in a real distributed environment there is a diversity of customer classes, each with possibly different objectives and different service and accounting requirements. In [6, 7], we have taken a game theoretic approach for performance optimization of competing classes in a distributed computing system. In those papers, we have formulated and solved the routing problem among competing classes of jobs as a Nash game.

It is quite common to require differentiated service among different classes by assigning different priorities to different classes, for example interactive jobs have higher priority than batch jobs. A high priority class may acquire most of the resources that it needs, while a low priority class should wait for the high priority class to complete service. Since the reason for having priorities is to give preferential treatment to the high priority jobs, it is *not* meaningful to define a *single* multi-objective function (ex. a convex combination of the objective functions of the different priority classes) for global optimization *across all the priority classes simultaneously*. However, we can still optimize the behavior of jobs within each priority class. Therefore a different approach should be taken for performance optimization of multipriority systems. In this paper, we formulate and optimize the performance of different priority classes as a Stackelberg game. In [4], we have considered the joint load sharing, routing and congestion control problem.

For simplicity of presentation, we consider two priority classes of customers which select between two servers. Jobs from the high priority class and jobs from the low priority class arrive to a two-processor system requiring execution. The problem of deciding to which processor each job will be assigned is the load sharing problem [9, 3] (Fig. 1).

2. QUEUEING MODEL

In this section, we introduce a simple queueing model of two servers that are shared by customers of two priority classes (Fig. 1). The problem is to assign these customers to the two servers so as to minimize the average delay of each class. An application is load sharing for a multiprocessor system, where interactive jobs (high priority) and batch jobs (low priority) may use two processors for execution. Another application is routing, where voice packets (high priority) and data packets (low priority) may use two different links for transmission between source-destination.

Let the high priority class α jobs arrive to the system with rate λ^α (Poisson arrivals) and require service times with mean $1/\mu^\alpha$ (exponential). On the other hand, the low priority class β jobs arrive to the system with rate λ^β (Poisson arrivals) and require service times with mean $1/\mu^\beta$ (exponential). Jobs of both classes may be served at either of the two processors, which have service rates C_1 and C_2, respectively. Furthermore, for stability reasons it is assumed that the total arrival rate of service requirements is less than the total service rate : $\dfrac{\lambda^\alpha}{\mu^\alpha} + \dfrac{\lambda^\beta}{\mu^\beta} \leq C_1 + C_2$.

The fraction of class $c \in \{\alpha, \beta\}$ jobs assigned to server $i \in \{1,2\}$ is ϕ_i^c, where $\phi_1^c + \phi_2^c = 1$, $\phi_1^c, \phi_2^c \geq 0$, such that its cost function $J^c(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$ is minimized.

In the following sections, we formulate and solve the load sharing problem of two processors among two priority classes as a Stackelberg game. Due to space limitation, we ommit the proofs which may be found in [5, 4].

3. STACKELBERG EQUILIBRIUM

In this section, we consider the load sharing problem, when two priority classes, with different objectives, share two processors. We formulate this priority multiobjective optimization problem as a non cooperative Stackelberg game [1] between the two priority classes.

Next, we give some definitions for a two-priority class (any kind of priorities) game similar to those in [1] for Stackelberg games:

Definition 1: *In a two-priority class finite game, with the high priority class α as the leader and the low priority class β as the follower, the set $R^\beta(\phi_1^\alpha, \phi_2^\alpha)$, defined for the high priority strategy $(\phi_1^\alpha, \phi_2^\alpha)$ that satisfies $\phi_1^\alpha + \phi_2^\alpha = 1$, $\phi_1^\alpha, \phi_2^\alpha \geq 0$, by:*

$$
\begin{aligned}
R^\beta(\phi_1^\alpha, \phi_2^\alpha) = \{ \ & (\phi_1^\beta, \phi_2^\beta) \ \textit{such that} \ \phi_1^\beta + \phi_2^\beta = 1, \ \phi_1^\beta, \phi_2^\beta \geq 0 : \\
& J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta) \leq J^\beta(\phi_1^\alpha, \phi_2^\alpha, \hat{P}_1^\beta, \hat{P}_2^\beta), \\
& \forall \ (\hat{P}_1^\beta, \hat{P}_2^\beta), \ \textit{such that} \ \hat{P}_1^\beta + \hat{P}_2^\beta = 1, \ \hat{P}_1^\beta, \hat{P}_2^\beta \geq 0\}
\end{aligned}
$$

is the optimal response (rational reaction) set of the low priority class β to the strategy of the high priority class α.

What the above definition says is that the low priority class β finds the set of its controls $(\phi_1^\beta, \phi_2^\beta)$, that minimize its cost function $J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$, for given strategy $(\phi_1^\alpha, \phi_2^\alpha)$ of the high priority class α.

Definition 2: *In a two-priority class finite game with the high priority class α as the leader, a strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, such that $\phi_1^{\alpha*} + \phi_2^{\alpha*} = 1, \quad \phi_1^{\alpha*}, \phi_2^{\alpha*} \geq 0$, is called a Stackelberg equilibrium strategy for the leader if*

$$\inf_{(\phi_1^\beta, \phi_2^\beta) \in R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})} J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^\beta, \phi_2^\beta) \leq \inf_{(\phi_1^\beta, \phi_2^\beta) \in R^\beta(\phi_1^{\alpha}, \phi_2^{\alpha})} J^\alpha(\phi_1^{\alpha}, \phi_2^{\alpha}, \phi_1^\beta, \phi_2^\beta)$$

$\forall\ (\phi_1^\alpha, \phi_2^\alpha)$ *such that* $\phi_1^\alpha + \phi_2^\alpha = 1, \quad \phi_1^\alpha, \phi_2^\alpha \geq 0.$

This means that the high priority class α finds the set of its optimal controls $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ that minimize its cost function $J^\alpha(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$, given the optimal response set $R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ of the low priority class β to its strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$.

Definition 3: *Let $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, such that $\phi_1^{\alpha*} + \phi_2^{\alpha*} = 1, \quad \phi_2^{\alpha*}, \phi_2^{\alpha*} \geq 0$, be a Stackelberg strategy for the leader α. Then any element $(\phi_1^{\beta*}, \phi_2^{\beta*}) \in R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ is an optimal strategy for the follower β that is in equilibrium with $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$. The strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$ is a Stackelberg solution for the game with the high priority class α as the leader and the cost pair $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta*}, \phi_2^{\beta*})$ is the corresponding Stackelberg equilibrium outcome.*

So, after the high priority class α has found its equilibrium strategy $(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, then the optimal response set of the low priority class β is given by $R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$. Any element $(\phi_1^{\beta*}, \phi_2^{\beta*}) \in R^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*})$ is an optimal strategy for the low priority class β.

4. PREEMPTIVE RESUME PRIORITY LOAD SHARING

In this section, we give a simple example for two preemptive resume priority classes of jobs that share two processors. When a high priority job is assigned to a processor, if there is another high priority job there, then it is put in the queue. If there are only low priority jobs there, then the low priority job is preempted and the high priority one starts been executing immediately. When all the high priority jobs have finished receiving service, then the low priority job that was preempted resumes and continues receiving service [8, 2]. The high priority class α (leader) assigns its jobs to the two processors, such that the average delay of its jobs is minimized. On the other hand, the low priority class β (follower) assigns its jobs to the two processors, such that the average delay of its jobs is minimized, after the high priority class α has optimally assigned its jobs. Thus a Stackelberg equilibrium is achieved.

4.1 General Two Class Solution

The cost function that we use for the high preemptive resume priority class α is its average job delay [8]: $J^\alpha(\phi_1^\alpha, \phi_2^\alpha) = \sum_{i=1}^{2} \frac{\phi_i^\alpha * \frac{1}{\mu^\alpha C_i}}{1 - \frac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i}}$.

Similarly, the cost function for the low preemptive resume priority class β is its average job delay [8]: $J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta) = \sum_{i=1}^{2} \frac{\phi_i^\beta * [\frac{1}{\mu^\beta C_i} - \frac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i * \mu^\beta C_i} + \frac{\lambda^\alpha \phi_i^\alpha}{(\mu^\alpha C_i)^2}]}{(1 - \frac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i}) * (1 - \frac{\lambda^\alpha \phi_i^\alpha}{\mu^\alpha C_i} - \frac{\lambda^\beta \phi_i^\beta}{\mu^\beta C_i})}$.

The overall average job delay is:

$$J(\phi_1^\alpha, \phi_2^\alpha, \phi_1, \phi_2^\beta) = \frac{\lambda^\alpha}{\lambda^\alpha + \lambda^\beta} * J^\alpha(\phi_1^\alpha, \phi_2^\alpha) + \frac{\lambda^\beta}{\lambda^\alpha + \lambda^\beta} * J^\beta(\phi_1^\alpha, \phi_2^\alpha, \phi_1^\beta, \phi_2^\beta)$$

Theorem 1 : There exists a Stackelberg equilibrium.

The high preemptive resume priority class α solves the following problem:

$$minimize \qquad J^{\alpha}(\phi_1^{\alpha}, \phi_2^{\alpha}) = \sum_{i=1}^{2} \frac{\phi_i^{\alpha}}{\mu^{\alpha} C_i - \lambda^{\alpha} \phi_i^{\alpha}}$$

$$\begin{array}{ll} with\ respect\ to & \phi_1^{\alpha},\ \phi_2^{\alpha} \\ such\ that & \phi_1^{\alpha} + \phi_2^{\alpha} = 1, \quad \phi_1^{\alpha},\ \phi_2^{\alpha} \geq 0. \end{array}$$

On the other hand, the low preemptive resume priority class β solves the following problem:

$$minimize \qquad J^{\beta}(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^{\beta}, \phi_2^{\beta}) = \sum_{i=1}^{2} \frac{\phi_i^{\beta} * [\frac{C_i}{\mu^{\beta}} - \frac{\lambda^{\alpha}\phi_i^{\alpha*}}{\mu^{\alpha}\mu^{\beta}} + \frac{\lambda^{\alpha}\phi_i^{\alpha*}}{(\mu^{\alpha})^2}]}{(C_i - \frac{\lambda^{\alpha}\phi_i^{\alpha*}}{\mu^{\alpha}}) * (C_i - \frac{\lambda^{\alpha}\phi_i^{\alpha*}}{\mu^{\alpha}} - \frac{\lambda^{\beta}\phi_i^{\beta}}{\mu^{\beta}})}$$

$$\begin{array}{ll} with\ respect\ to & \phi_1^{\beta},\ \phi_2^{\beta} \\ such\ that & \phi_1^{\beta} + \phi_2^{\beta} = 1, \quad \phi_1^{\beta},\ \phi_2^{\beta} \geq 0. \end{array}$$

Then, the following policy allocates the arriving jobs to the two servers such that a Stackelberg equilibrium is achieved:

If $\quad \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_1 + C_2,$

then

If $\ C_1 - \sqrt{C_1 C_2} \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}}$ *and* $C_2 - \sqrt{C_1 C_2} \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}}$

$$then \ \phi_1^{\alpha*} = \frac{C_1}{\frac{\lambda^{\alpha}}{\mu^{\alpha}}} - \frac{C_1 + C_2 - \frac{\lambda^{\alpha}}{\mu^{\alpha}}}{\frac{\lambda^{\alpha}}{\mu^{\alpha}}} * \frac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$$

If $\ 0 \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_1 - \sqrt{C_1 C_2}$

then $\ \phi_1^{\alpha*} = 1$

If $\ 0 \leq \frac{\lambda^{\alpha}}{\mu^{\alpha}} \leq C_2 - \sqrt{C_1 C_2}$

then $\ \phi_1^{\alpha*} = 0$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_1^\alpha + C_2^\alpha$$

then

$$If \quad C_1^\alpha - C_2^\alpha * \sqrt{\frac{C_1^\beta}{C_2^\beta}} \le \frac{\lambda^\beta}{\mu^\beta} \quad and \quad C_2^\alpha - C_1^\alpha * \sqrt{\frac{C_2^\beta}{C_1^\beta}} \le \frac{\lambda^\beta}{\mu^\beta}$$

$$then \quad \phi_1^{\beta *} = \frac{C_1^\alpha}{\frac{\lambda^\beta}{\mu^\beta}} - \frac{C_1^\alpha + C_2^\alpha - \frac{\lambda^\beta}{\mu^\beta}}{\frac{\lambda^\beta}{\mu^\beta}} * \frac{\sqrt{C_1^\beta}}{\sqrt{C_1^\beta} + \sqrt{C_2^\beta}}$$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_1^\alpha - C_2^\alpha * \sqrt{\frac{C_1^\beta}{C_2^\beta}}$$

$$then \quad \phi_1^{\beta *} = 1$$

$$If \quad \frac{\lambda^\beta}{\mu^\beta} \le C_2^\alpha - C_2^\alpha * \sqrt{\frac{C_2^\beta}{C_1^\beta}}$$

$$then \quad \phi_1^{\beta *} = 0$$

where

$$C_1^\alpha = C_1 - \frac{\lambda^\alpha \phi_1^{\alpha *}}{\mu^\alpha} \qquad C_2^\alpha = C_2 - \frac{\lambda^\alpha \phi_2^{\alpha *}}{\mu^\alpha}$$

$$C_1^\beta = \frac{C_1}{\mu^\beta} - \frac{\lambda^\alpha \phi_1^{\alpha *}}{\mu^\alpha \mu^\beta} + \frac{\lambda^\alpha \phi_1^{\alpha *}}{(\mu^\alpha)^2} \quad C_2^\beta = \frac{C_2}{\mu^\beta} - \frac{\lambda^\alpha \phi_2^{\alpha *}}{\mu^\alpha \mu^\beta} + \frac{\lambda^\alpha \phi_2^{\alpha *}}{(\mu^\alpha)^2}$$

Of course, the equilibrium load sharing probabilities to the other server are $\phi_2^{\alpha *} = 1 - \phi_1^{\alpha *}$ and $\phi_2^{\beta *} = 1 - \phi_1^{\beta *}$. Substituting these equilibrium probabilities into the average delay functions, we have the equilibrium outcome of the game [5, 4].

4.2 Interesting Results

From the above equilibrium solution and outcome of the game, we have some interesting results:

Proposition 1: *For a given system $C_1 \ge C_2$,*
if $\mu^\alpha = \mu^\beta = \mu$, $\lambda^\alpha + \lambda^\beta \le \mu(C_1 + C_2)$, and $\lambda^\alpha + \lambda^\beta = \lambda = constant$,
then $J(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{\beta *}, \phi_2^{\beta *}) = constant$

The above proposition says that for equal mean service requirements for both priority classes and constant total arrival rate, the overall average job delay is constant, i.e. it does not depend on the mix of high and low priority jobs.

In Fig. 2, we show the equilibrium average delay of the high priority class α, $J^\alpha(\phi_1^{\alpha *}, \phi_2^{\alpha *})$, of the low priority class β, $J^\beta(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{\beta *}, \phi_2^{\beta *})$, and of the system $J(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^{\beta *}, \phi_2^{\beta *})$

versus different mixes of the high and low priority arrival rates $\frac{\lambda^\alpha}{\lambda^\beta}$, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed total arrival rate $\lambda^\alpha + \lambda^\beta = 2.5$, and equal mean service requirement of the high and the low priority jobs $1/\mu^\alpha = 1/\mu^\beta = 1$. We note that the overall average job delay is constant and independent from the mix of the high and low priority jobs.

Proposition 2: *For a given system with $C_1 \geq C_2$ and $\mu^\alpha = \mu^\beta = \mu$, if $\mu(C_1 + C_2) \leq \lambda^\alpha + \lambda^\beta$ and $C_1 - \sqrt{C_1 C_2} \leq \lambda^\alpha/\mu$, then* $\phi_1^{\beta *} = \frac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$

What the above proposition says is that for equal mean service requirements for both priority classes, when the high priority class α uses both servers, then the equilibrium decisions of the low priority class β are constant and independent of the arrival rates $(\lambda^\alpha, \lambda^\beta)$. This result is not intuitive, because we might expect that the load sharing decisions for the low priority class β should also depend on the arrival rates (as it is the case for the high priority class α). It is also very important, because even when the arrival rates vary over time, the load sharing policy for the low priority jobs remains the same (Fig. 3).

Proposition 3: *For a given system $C_1 \geq C_2$: if* $\frac{\lambda^\alpha}{\mu^\alpha} + \frac{\lambda^\beta}{\mu^\beta} \rightarrow C_1 + C_2$, *then* $\phi_1^{\beta *} \rightarrow \frac{\sqrt{C_1}}{\sqrt{C_1} + \sqrt{C_2}}$.

The above proposition says that even for different service requirements for the two priority classes, when the total arriving service requirement $\frac{\lambda^\alpha}{\mu^\alpha} + \frac{\lambda^\beta}{\mu^\beta}$ approaches the total service capacity $C_1 + C_2$, the load sharing decisions for the low priority class β become constant and independent from the arrival rates.

In Fig. 3, we show the equilibrium load sharing probabilities of both the high priority class α, $\phi_1^{\alpha *}$, and of the low priority class β, $\phi_1^{\beta *}$, versus the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$, for fixed server capacities $C_1 = 2, C_2 = 1$, equal arrival rates $\lambda^\alpha = \lambda^\beta$ and different ratio of the mean service requirement of the high and the low priority jobs $1/\mu^\alpha = 2/\mu^\beta = 1$, $1/\mu^\alpha = 1/\mu^\beta = 1$, $1/\mu^\beta = 2/\mu^\alpha = 1$.

For equal mean service requirements of the high and low priority jobs, we note that when the high priority class α uses both processors ($0 < \phi_1^{\alpha *} < 1$), then the load sharing decisions $\phi_1^{\beta *}$ for the low priority class β are constant and independent of the arrival rates $\lambda^\alpha, \lambda^\beta$. That means that even if the arrival rates change during operation, our load sharing algorithm will still perform "optimally" for the low priority class. For different mean service requirements of the high and low priority jobs, we note that when the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$ approaches 1, then the load sharing decisions $\phi_1^{\beta *}$ for the low priority class β approach the same constant value as for the case of equal mean service requirement.

5. NUMERICAL RESULTS & DISCUSSION

In this section, we discuss some other results that can be derived from the solution of the two processor load sharing problem among jobs from two preemptive resume priority classes.

5.1 Constant λ^α

Consider a two processor system $C_1 \leq C_2$ with fixed arrival rate of interactive (high priority) jobs $\lambda^\alpha = constant$. If this multiprocessor is also to be used by batch (low priority) jobs and we want to secure an upper bound on the average delay of batch jobs $J^\beta \leq J_0^\beta$, then we should restrict the arrival rate of the batch jobs up to an upper limit. For example, if $\mu^\alpha = \mu^\beta = \mu$, Case 1, then

$$\frac{(\sqrt{C_1}+\sqrt{C_2})^2}{\mu * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu}) * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu} - \frac{\lambda^\beta}{\mu})} \leq J_0^\beta \quad \Rightarrow$$

$$\lambda^\beta \leq \mu * (C_1 + C_2) - \lambda^\alpha - \frac{(\sqrt{C_1}+\sqrt{C_2})^2}{J_0^\beta * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu})}$$

In Fig. 4, we show the equilibrium average delay of both the higher priority class α, $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, of the lower priority class β, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^*, \phi_2^{\beta*})$, and of the system $J(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^*, \phi_2^{\beta*})$ versus the low priority class β arrival rate, λ^β, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed mean service requirements $1/\mu^\alpha = 1/\mu^\beta = 1$ and fixed high priority class α arrival rate $\lambda^\alpha = 1.0$. So, for example, if the average delay of batch jobs J^β should be less than 10, then the arrival rate of batch jobs should be $\lambda^\beta < 1.71$.

5.2 Constant λ^β

Next, consider a two processor system $C_1 \leq C_2$ with fixed arrival rate of batch jobs (low priority) $\lambda^\beta = constant$. If this multiprocessor is also to be used by interactive jobs (high priority) and we want to secure an upper bound on the average delay of batch jobs $J^\beta \leq J_0^\beta$, then we should restrict the arrival rate of the interactive jobs up to an upper limit. For example, if $\mu^\alpha = \mu^\beta = \mu$, Case 1, then

$$\frac{(\sqrt{C_1}+\sqrt{C_2})^2}{\mu * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu}) * (C_1 + C_2 - \frac{\lambda^\alpha}{\mu} - \frac{\lambda^\beta}{\mu})} \leq J_0^\beta \quad \Rightarrow$$

$$\lambda^\alpha \leq \mu(C_1 + C_2) - \frac{\lambda^\beta}{2} - \sqrt{(\frac{\lambda^\beta}{2})^2 + \frac{\mu(\sqrt{C_1}+\sqrt{C_2})^2}{J_0^\beta}}$$

In Fig. 5, we show the equilibrium average delay of both the higher priority class α, $J^\alpha(\phi_1^{\alpha*}, \phi_2^{\alpha*})$, of the lower priority class β, $J^\beta(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^*, \phi_2^{\beta*})$, and of the system $J(\phi_1^{\alpha*}, \phi_2^{\alpha*}, \phi_1^*, \phi_2^{\beta*})$ versus the high priority class α arrival rate, λ^α, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed mean service requirements $1/\mu^\alpha = 1/\mu^\beta = 1.$ and fixed low priority class β arrival rate $\lambda^\beta = 1.0$.

So, for example, if the average delay of batch jobs J^β should be less than 10, then the arrival rate of interactive jobs should be $\lambda^\beta < 1.59$.

5.3 Constant λ

Thirdly, consider a two processor system $C_1 \leq C_2$ with fixed total arrival rate of both interactive and batch jobs $\lambda^\alpha + \lambda^\beta = \lambda = constant$. We want to determine what mix of interactive and batch jobs will secure an upper bound on the average delay of interactive jobs $J^{\leq} J_0^\alpha$ as well as on batch jobs $J^\beta \leq J_0^\beta$. Let $k = \frac{\lambda^\alpha}{\lambda^\beta}$ be the mix of interactive over batch jobs. Then we can write the arrival rate of interactive jobs as $\lambda^\alpha = \frac{k}{k+1}\lambda$ and the arrival rate of batch jobs as $\lambda^\beta = \frac{1}{k+1}\lambda$. For example, if $\mu^\alpha = \mu^\beta = 1$, Case 2.2, then

$$\frac{1}{\mu C_1 - \frac{k}{k+1}\lambda} \leq J_0^\alpha \qquad \Rightarrow k \leq \frac{J_0^\alpha * \mu C_1}{1 - J_0^\alpha * (\mu C_1 - \lambda)}$$

$$\frac{\mu C_1}{(\mu C_1 - \frac{k}{k+1}\lambda) * (\mu C_1 - \lambda)} \leq J_0^\beta \quad \Rightarrow k \leq \frac{\mu C_1 * [J_0^\beta * (\mu C_1 - \lambda) - 1]}{\mu C_1 - J_0^\beta * (\mu C_1 - \lambda)^2}$$

In Fig. 2, we show the equilibrium average delay of both the high priority class α, $J^\alpha(\phi_1^{\alpha *}, \phi_2^{\alpha *})$, of the low priority class β, $J^\beta(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^*, \phi_2^{\beta *})$, and of the system $J(\phi_1^{\alpha *}, \phi_2^{\alpha *}, \phi_1^*, \phi_2^{\beta *})$ versus different mix of the high and low priority arrival rates $k = \frac{\lambda^\alpha}{\lambda^\beta}$, for fixed server capacities $C_1 = 2, C_2 = 1$, fixed total arrival rate $\lambda = \lambda^\alpha + \lambda^\beta = 2.5$, and equal mean service requirement of the high and the low priority jobs $1/\mu^\alpha = 1/\mu^\beta = 1$.

So, for example, if the average delay of interactive jobs should be less than 2 and the average delay of batch jobs should be less than 10, then mix of interactive and batch jobs should be $\frac{\lambda^\alpha}{\lambda^\beta} < 2.8$.

5.4 Different Server Rate Ratios

Finally, in Fig. 6, we show the equilibrium probability to processor 1, of both the high and low priority classes versus the system load $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$ for equal arrival rates $\lambda^\alpha = \lambda^\beta$, equal mean sevice requirements $1/\mu^\alpha = 1/\mu^\beta = 1$ and different ration of the service rates of the two processors $\frac{C_1}{C_2} = 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5$.

When the service rate of server 1 is substantially larger than the service rate of server 2, ($C_1 = 5 * C_2$), then server 1 is used exclusively for almost all arrival rates. When the service rates are $C_1 = 4 * C_2$, then for low and medium load, server 1 is exclusively used, but for heavy system load, server 2 also is used. When the service rates are $C_1 = 3 * C_2$, then the slow server starts been used for lower system load. When the service rates are $C_1 = 2 * C_2$, then the slow server starts been used for even lower system load. When the service rates are equal $C_1 = C_2$, then both servers are used equally ($\phi_1^{\alpha *} = \phi_2^{\alpha *} = \phi_1^{\beta *} = \phi_2^{\beta *} = 0.5$). Now, when server 2 is faster, a similar scenario happens, i.e. the faster server 2 is, the more it is exclusively used.

6. CONCLUSIONS

In this paper, we formulate and solve a *priority load sharing* problem. Real distributed systems assign different priorities to different classes of jobs, in order to give preferential treatment to some classes of jobs. Therefore, it is not meaningful to optimize a single function over all different priority classes simultaneously. In this paper, we introduce an alternative methodology for dealing with multipriority optimization problems. We formulate a two-priority class load sharing problem as a Stackelberg game with leader the high priority class and follower the low priority class. Furthermore, we gave the explicit solution when two preemptive resume priority classes want to minimize their average job delay. We found that for equal mean service requirements of jobs from both classes, when both processors are used, then the decisions of the low priority class do not depend on the arrival rates, i.e. even if the arrival rates vary, the same routing probabilities can be used for the low priority jobs. Also, for equal mean service requirements of jobs from both classes, when the total arrival rate of jobs is constant but the mix of high and low priority jobs varies, then the overall average job delay remains constant.

Straightforward extensions are to consider multiple priority (> 2) classes, as well as more than two servers. This is the first study that formulates and solves a multi-priority resource allocation problem as a Stackelberg game. We have also formulated and solved the joint load sharing, routing and congestion control problem as a Nash and a Stackelberg game [4].

References

[1] T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, 1982.

[2] D.P. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, 1987.

[3] D.L. Eager, E.D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. on Software Eng., Vol SE-12, No 5*, pp. 662-675, May 1986.

[4] A.A. Economides. A unified game-theoretic methodology for the joint load sharing, routing and congestion control problem. *Ph.D. Dissertation, University of Southern California*, August 1990.

[5] A.A. Economides and J.A. Silvester. Priority load sharing: an approach using Stackelberg games. *USC Technical Report CENG 89-39*, 1989.

[6] A.A. Economides and J.A. Silvester. Routing games. *USC Technical Report CENG 89-38*, 1989.

[7] A.A. Economides and J.A. Silvester. A game theory approach to cooperative and noncooperative routing problems. *Proc. IEEE International Telecommunications Symposium*, Brazil, Sept. 3-6, 1990.

[8] L. Kleinrock. *Queueing Systems, Vol. 2: Applications.* J. Wiley & Sons, 1976.

[9] Y.-T. Wang and R.T.J. Morris. Load sharing in distributed systems. *IEEE Trans. on Computers, Vol C-34, No 3*, pp. 204-217, March 1985.

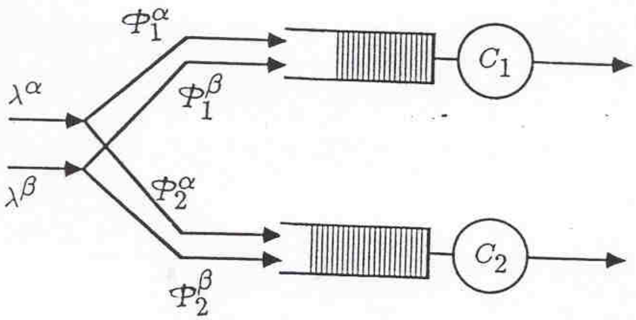

Fig. 1 A Two Priority Load Sharing Problem

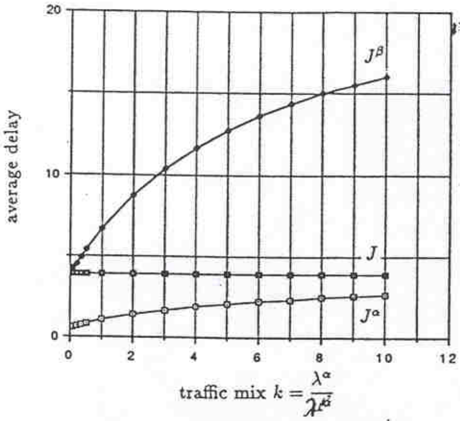

Fig. 2 Stackelberg equilibrium average delay J^α, J^β and J versus different mix of the high and low priority arrival rates $\frac{\lambda^\alpha}{\lambda^\beta}$, for $C_1 = 2, C_2 = 1$, $\lambda^\alpha + \lambda^\beta = 2.5$, and $\mu^\alpha = \mu^\beta = 1$.

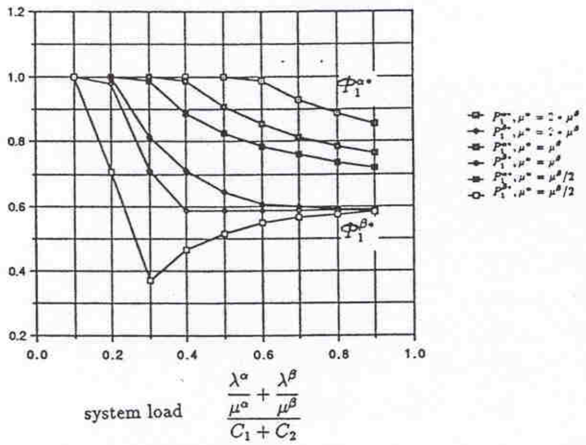

Fig. 3 Stackelberg equilibrium probabilities $\Phi_1^{\alpha*}$ and $\Phi_1^{\beta*}$ versus $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$, for $C_1 = 2, C_2 = 1$, $\lambda^\alpha = \lambda^\beta$ and $1/\mu^\alpha = 2/\mu^\beta = 1$, $1/\mu^\alpha = 1/\mu^\beta = 1$, $1/\mu^\beta = 2/\mu^\alpha = 1$.

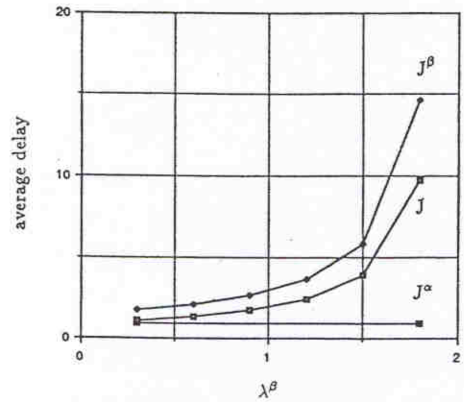

Fig. 4 Stackelberg equilibrium average delay J^α, J^β and J versus λ^β, for $C_1 = 2, C_2 = 1$, $1/\mu^\alpha = 1/\mu^\beta = 1$ and $\lambda^\alpha = 1.0$.

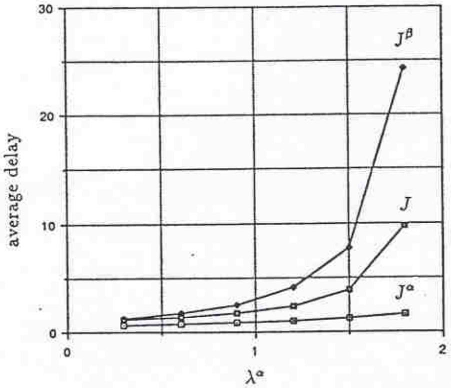

Fig. 5 Stackelberg equilibrium average delay J^α, J^β, and J versus λ^α, for $C_1 = 2, C_2 = 1$, $1/\mu^\alpha = 1/\mu^\beta = 1$ and $\lambda^\beta = 1.0$.

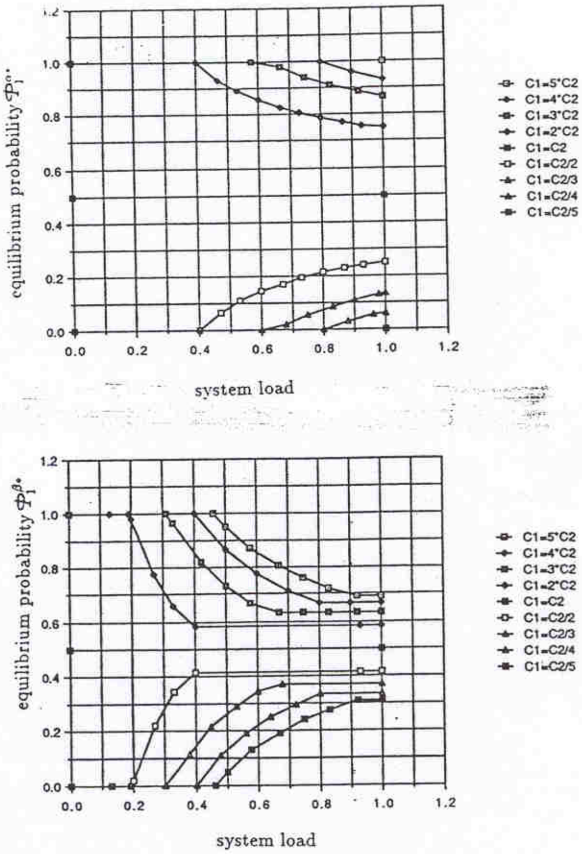

Fig. 6 Stackelberg equilibrium probabilities $\Phi_1^{\alpha*}$ and $\Phi_1^{\beta*}$ versus $\frac{\lambda^\alpha/\mu^\alpha + \lambda^\beta/\mu^\beta}{C_1 + C_2}$, for $\lambda^\alpha = \lambda^\beta$, $1/\mu^\alpha = 1/\mu^\beta = 1$ and $\frac{C_1}{C_2} = 5, 4, 3, 2, 1, 1/2, 1/3, 1/4, 1/5$.