International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 9, Issue 3 (December 2013), PP. 10-16

Optimal Assembly Plan and Control of Networks

Vaya Dinopoulou[1], Konstantinos Kakoulis[2], Panagiotis Kyratsis[3*]
TEI of Western Macedonia, Dept. Mechanical & Industrial Design Engineering, Kila Kozani GR50100 Greece

Abstract:- The routing problem of jobs/customers arriving in a network of parallel queues, constitutes a problem of queuing theory with important applications. Consider N parallel queues each with finite buffer size. The buffer size of queue i is denoted by Ni and it is equal with the number of processors in the queue. Jobs arrive according to a Poisson process with rate λ. A controller assigns new arrivals to queues. Arriving jobs, that find all buffers occupied, are lost according to the system. A job assigned to a processor may not change in the future. The time it takes for a job to be processed is exponentially distributed with rate μ. Processing times and arrival times are all assumed to be independent. The problem is the determination of the assembly plan of such systems and their control. The criterion that is used is the minimisation of sum of the services quality cost and server holding cost.

Keywords:- Optimal server allocation; optimal control of parallel queues; service level agreement.

I. INTRODUCTION

The routing problem of jobs/customers arriving in a network of parallel queues, constitutes a problem of queuing theory with particular interest and important applications (e.g. networks of computers-parallel processors, telecommunications-telephone networks etc). As an example consider the telecommunications-telephone networks. All calls (jobs/customers) arriving at the call centre, must be assigned to the operators for servicing. Also computer networks (i.e. web sites, internet providers) offer to their customers, the use of hardware and software, in order several services to be executed (information searching, e-commerce etc). In these cases, a serious problem of managing high quality services with the minimum cost arises. The total cost of such system comes up as a combination of both, the services quality cost (Quality of Service - QoS) and the cost from keeping the resources (equipments) idle (Server Holding Cost). Service level agreements are agreed between application providers and customers, in order to secure high level services. If violated it results in QoS cost which comprises of penalties for poor performance and customer dissatisfaction when they need to use the service and the system is unable to provide it due to overload. Figure 1 depicts an example of cost estimation for QoS ([1], [2]). This cost approximates an exponential function of the time that a customer needs to be served from the system and this cannot be accomplished because processors are not available. In order to avoid this situation, more resources (processors) must be added. As a result the service to the customer will be of a better quality, while at the same time, penalties due to low service quality will be avoided. Unfortunately, this leads to cost increase, because more processors remain idle when no customer seeks for their services in a specific period of time. In the example of the telecommunications-telephone network, this cost comes from the labor cost when these people remain without work, because no calls are entered into the centre. In the case of computer networks, this cost includes the running cost (i.e. electricity) and the fixed assets depreciation (a percentage of the initial investment).

Consider a system of N groups of processors, every group with Ni processors (i.e. Ni computers or alternatively the system comprises of N computers while each of them is able to execute Ni jobs simultaneously). An appropriate example of such system is an internet provider that offers a group of processors (computer network) in every major city and receives the customer requests for execution.

The present paper proposes an assembly plan of such systems and a routing policy of jobs in multiprocessor queues, such that minimized the total cost of the system. The remainder of this paper is organized as follows. Section 2 develops the cost model. Section 3 describes the formulation of the problem (3.1), the optimised control policy of the system (3.2) and the optimal assembly plan (3.3). Finally, Section 4 contains the conclusions.

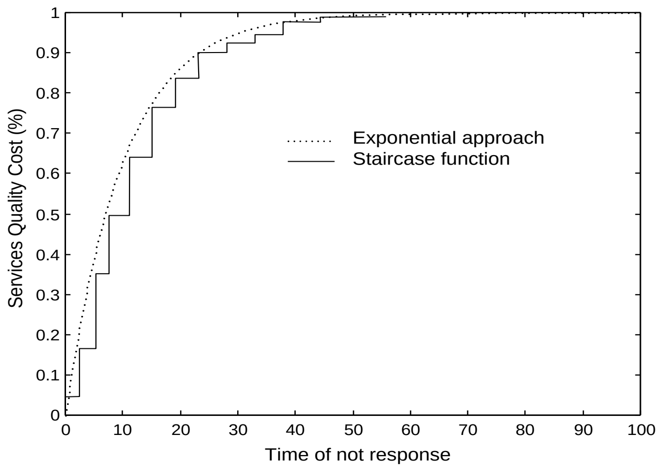


Fig. 1. Staircase function of QoS cost.

II. COST MODEL

This section describes the system's cost model. This model quantifies the cost of QoS and the server holding cost. A similar model was proposed by [1] and it approximates the real cost occurring within application providers.

Consider the function of the total cost of the system $c_t(y_1, y_2, \ldots, y_N)$, where y_i is the number of processors which are available, at the ith queue, in every specific time period t. Assuming that the system remains in the same condition $((y_1, y_2, \ldots, y_N)$, i.e. the number of the idle processors remain constant) for a period of T, the subscript t is omitted and the total cost function of the system for a period T is redefined to $c(y_1, y_2, \ldots, y_N)$. Lets assume that $c_Q(y_1, y_2, \ldots, y_N)$ is the cost of the services quality (QoS) and $c_H(y_1, y_2, \ldots, y_N)$ the server holding cost. Then

$$c(y_1, y_2, \ldots, y_N) = c_Q(y_1, y_2, \ldots, y_N) + c_H(y_1, y_2, \ldots, y_N) \qquad (1)$$

A staircase function (Fig.1) is used in order to simulate the services quality cost (QoS) $c_Q(y_1, y_2, \ldots, y_N)$. Lets $f(y_1, y_2, \ldots, y_N)$ be a function which describes the metrics of the service quality. This function could be the time response of the system to the customer request and is considered nonincreasing in each y_i. That results:

$$c_Q(y_1, y_2, \ldots, y_N) = b(1 - e^{-af(y_1, y_2, \ldots y_N)}) \qquad (2)$$

Parameter b is a constant that determines the maximum value of the services quality cost $c_Q(y_1, y_2, \ldots, y_N)$. Parameter α is considered to be the rate of the services quality cost increase.

The cost of the processors available is determined by summing a constant cost (investment capital) and several expenditures (running cost). The service providers require to cover that cost using an appropriate depreciation rate depending on the lifetime of every processor (this rate should be high enough in order to cover both constant and variable costs).

Let h_i the cost of one processor when it remains idle for one second. Then the total cost of all processors remaining idle for a period of time T is:

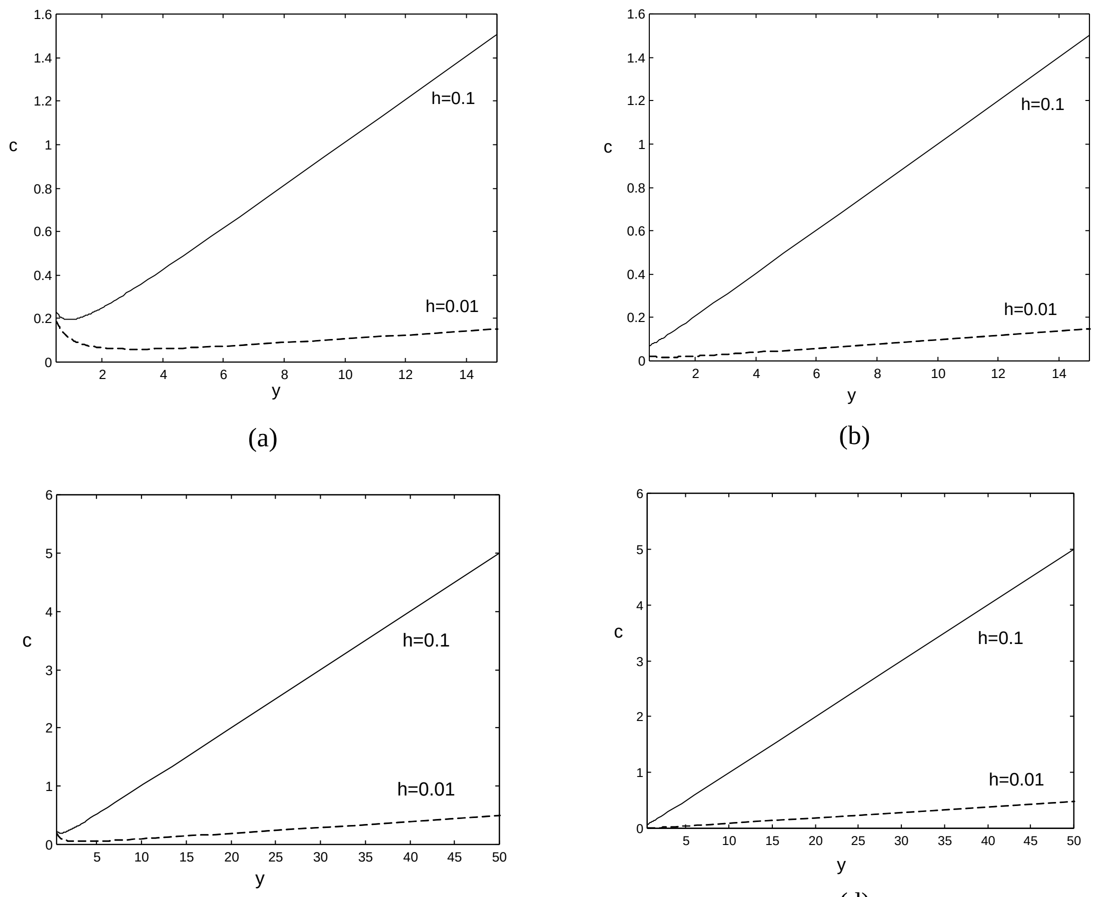
$$c_H(y_1, y_2, \ldots, y_N) = \sum_{i=1}^{N} h_i T y_i \qquad (3)$$

Substituting both equations (2) and (3) in equation (1) and dividing by b (for normalizing h_i) result:

$$c(y_1, y_2, \ldots, y_N) = 1 - e^{-af(y_1, y_2, \ldots y_N)} + \sum_{i}^{N} h_i T y_i \qquad (4)$$

The value of h_i in equation (4) is normalized (divided by b). The next step is to determine the values of α and h_i. Parameter α represents how fast the cost of the services quality increases. For example the dot line in fig 1 corresponds to α=0.1. Use of α=0.01 leads in a less steep cost of services quality increase. The normalised value of h_i represents the ratio of the investment capital plus the running cost for a period of time in which the depreciation will take place (i.e. a year) divided by the maximum cost of the services quality (QoS). The maximum cost of the service quality (QoS) encompasses both the cost of penalties agreed with the customers and the lost profit because of loosing all current or future customers. A quite realistic value for this ratio is 10% which in return means h_i=0.1. In the case that the capital depreciation needs to be smaller, the ratio can take

values down to 1%. As a result we assume that the cost function receives values of h_i between 0.01 and 0.1.
Let us assume that there is only one queue of processors (N=1) and the time period is 1 (T=1). In this simplified case, equation (4) becomes $c(y) = 1 - e^{-\alpha f(y)} + hy$. Fig 2 depicts the cost curve for a number of different values of α and h. When observing the cost curves it becomes clear that the significant cost comes from keeping the processors idle. When h is increased, the cost of keeping the processors idle increases substantially and becomes significantly larger than the cost of services quality (QoS). When the number of processors in the queue increases (that means that more processors may remain idle) the cost increases rapidly. This is true for every value of h. In the case that the number of processors in the queue equals 50 (fig.2 (c) and (d)) the cost of services quality (QoS) becomes negligible, compared to the cost of the idle processors. In networks of service providers, considering the ownership of at least 50 processors is quite realistic.



(a) (b)

(c) α=0.1 (d) α=0.01

Fig. 2. Cost Curves for N=1 and T=1.

As a result, it becomes clear that it is crucial to minimize the cost of having idle processors. The fewer the idle processors for a period of time, the less is their cost. The exact number of idle processors in every period of time, depends on the number of processors available and the number of customer requesting services for the same period of time which is a random variable. At this stage, it is true to say that, the minimization of total cost takes place when maximizing the possibility more processors to be occupied every period of time.

III. PROBLEM DESCRIPTION

Consider N parallel queues each with finite buffer size (capacity). The buffer size of queue i is denoted by N_i and it is equal with the number of processors in the queue (Fig. 3). Jobs arrive according to a Poisson process with rate λ. A controller assigns new arrivals to queues. Arriving jobs, that find all processors occupied, are lost according to the system. A job assigned to a processor may not change in the future. The time it takes for a job to be processed is exponentially distributed with rate μ. Processing times and arrival times are all assumed to be independent. The problem is the determination of the assembly plan of such systems, in other words, the determination of the number of processors of each queue, and then, the control of the system, that is to say the routing policy in order to optimise some criterion. The criterion that is used is the minimisation of sum of the services quality cost and server holding cost. In particular, for mean processing time sufficiently long (i.e. for time consuming jobs) we describe the optimal allocation of processors and the optimal job assignment policy.

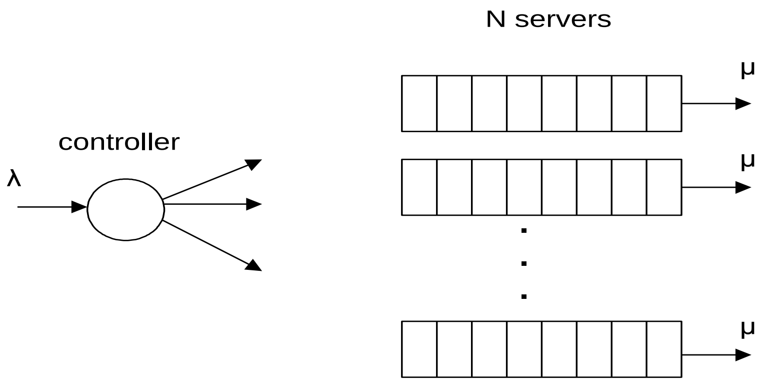


Fig. 3. Architecture of control system.

Reference [3] examined this problem under two assumptions: (a) N_i=1, i=1, …, N and (b) infinite buffer size for all queues. His main result was that the “join the shortest queue” (JSQ) policy stochastically maximizes the discounted number of jobs that complete service by a certain time. Reference [4] showed that if assumption (a) is dropped the obvious generalization of the JSQ policy (the policy that routes jobs to the queue with “the shortest expected delay” or SED policy) does not minimize the long run average delay per customer. Reference [5] extended Winston’s result ([10]) for finite buffer size (i.e. they dropped assumption (b) but kept (a)). Reference [6] pointed out the usefulness of the multiserver model for telephone networks and without assuming (a), it showed that for N=2 the SED policy performs reasonably well. Reference [1] is adding the assumption that servers break down and repair with rates α and β respectively, it considers one queue and determines the number of servers that minimizes the total cost.

In the present paper, we describe the optimal policy without the assumption (a) and replace (b) with the assumption that the capacity of each queue is equal with the number of processors in the queue.

A. Problem formulation

The status of the system is given by a state vector $\mathbf{x}(t)$ with $x_i(t)$ denoting the number of jobs in queue i at the time instant t, $0 \le x_i(t) \le N_i$ for i=1, …, N. Change in the state $\mathbf{x}$ means that either a job is completed and leaves the system $(-1_j\ \mathbf{x})=(x_1, x_2, \ldots, x_j-1, \ldots, x_N)$ or a new job arrives in the systems $(1_i\ \mathbf{x})=(x_1, x_2, \ldots, x_i+1, \ldots, x_N)$. If we assume that the new job is assigned to the queue $\alpha(\mathbf{x})$, $x_{\alpha(\mathbf{x})}<N_{\alpha(\mathbf{x})}$, the possible transitions from state $\mathbf{x}$ are either to a state $(-1_j\ \mathbf{x})$ for j such that $x_j>0$ or to $(1_{\alpha(\mathbf{x})}\ \mathbf{x})$. We will assume that a control level K_i is associated with each queue i, such that $0<K_i \le N_i$, i=1, …, N, and we examine the problem of the maximization of probability that all queues have K_i or more occupied processors (jobs) at any time instant t.

Let $c(\mathbf{x}(t))$ be the random failure process taking values 1 or 0 according to whether the following event A: “all queues have K_i or more jobs” is true or not. The distribution of this process depends on the job allocation policy π. To show that a policy π^* maximizes the probability that (event A is true) all queues have K_i or more jobs at any time instant t, we have to show that

$$P_{\pi^*}(c(\mathbf{x}(t))=1) \ge P_{\pi}(c(\mathbf{x}(t))=1) \text{ for all policies } \pi.$$

The approach used to deal with this problem takes the following steps:

- The continuous-time problem is discretized via uniformization and consideration of embedded Markov decision process ([7], [8]).
- Used Dynamic Programming techniques to compare different policies ([8]).

Let $I(\mathbf{x})$ denote the minimum number of jobs that have to be served when the system is in state $\mathbf{x}$ such that the event A is not true. Consider the family $\mathrm{P_M}$, M=0,1,2,… of discrete finite horizon Dynamic Programming problems with optimality equations given by

$$m = 1,\ldots,M$$

$$u(\mathbf{x};m) = \min_{\pi(x)} \left[(1-\lambda-\mu\sum_j x_j)u(\mathbf{x};m-1) + \lambda u(1_{\pi(\mathbf{x})}\mathbf{x};m-1) + \mu\sum_j x_j u(-1_j\mathbf{x};m-1) \right]$$

$$u(\mathbf{x};0) = \begin{cases} 0 & if \quad I(\mathbf{x})>0 \\ 1 & if \quad I(\mathbf{x})=0 \end{cases}$$

(5)

Let $u_\pi(\mathbf{x};\, m)$ denote the value function of a policy π. Using uniformization and a time rescaling so that the uniform rate is 1, one gets the following equation

$$P_\pi(c(\mathbf{x}(t)) = 0) = \sum_{M=0}^{\infty} u_\pi(\mathbf{x}; M) e^{-t} \frac{t^M}{M!} \qquad (6)$$

Hence, to show that a policy is optimal in the original problem, it is sufficient (but not necessary) for its discrete version to be optimal in the family P_M, M=0,1,2,…. A necessary and sufficient condition for a stationary policy $\pi(\mathbf{x})$ to be strongly optimal is the following

$$u_\pi(1_{\pi(\mathbf{x})}\mathbf{x}; m) \le u_\pi(1_i \mathbf{x}; m) \;\; \forall i \in A(\mathbf{x}) - \{\pi(\mathbf{x})\} \; \forall m \ge 0 \qquad (7)$$

The value function $u_\pi(\mathbf{x};\, m)$ is a polynomial in μ.

$$u_\pi(\mathbf{x}; m) = \sum_{\nu=0}^{\infty} u_\pi^{(\nu)}(\mathbf{x}; m)\mu^\nu \qquad (8)$$

The coefficients $u_\pi^{(\nu)}(\mathbf{x};m)$ vanish for $\nu>\nu_0(\mathbf{x};m)$ for some $\nu_0(\mathbf{x};m)$. Substituting this relation in the optimality equations we get the following recursive equations for ν=0,1,2,… and m=0,1,2,…

$$u_\pi^{(\nu+1)}(\mathbf{x}; m+1) = (1-\lambda)u_\pi^{(\nu+1)}(\mathbf{x}; m) + \lambda u_\pi^{(\nu+1)}(1_{\pi(\mathbf{x})}\mathbf{x}; m) + \sum_{\ell} x_\ell (u_\pi^{(\nu)}(-1_\ell \mathbf{x}; m) - u_\pi^{(\nu)}(\mathbf{x}; m))$$

$$u_\pi^{(0)}(\mathbf{x}; m+1) = (1-\lambda)u_\pi^{(0)}(\mathbf{x}; m) + \lambda u_\pi^{(0)}(1_{\pi(\mathbf{x})}\mathbf{x}; m)$$

$$u_\pi^{(0)}(\mathbf{x};0) = \begin{cases} 0 & if \quad I(\mathbf{x}) \ge 1 \\ 1 & if \quad I(\mathbf{x}) = 0 \;\; and \;\; \nu = 0 \\ 0 & if \quad I(\mathbf{x}) = 0 \;\; and \;\; \nu > 0 \end{cases}$$

(9)

For μ sufficiently small the optimal policy will be determined by the leading (i.e. the first non-zero) coefficients of this power series expansion.

B. Optimal control

The close interconnection between reliability and queuing models is well known and has been extensively discussed in the pertinent literature. The problem of assigned jobs to parallel queues corresponds to the problem of maintenance for reliability of systems in series. For this type of problems, the optimal control policy corresponds to the optimal maintenance policy. Reference [9] examined the problem of reliability of systems operating in series and described the repair allocation policy which maximizes the reliability of the system at any time instant t. This result is extended to the problem of controlling the corresponding network of parallel queues as follow:

Let $P(\mathbf{x})$:={i: $K_i \le x_i \le N_i - 1$} (the set of all queues that have at least K_i jobs and at least one idle server), $Q(\mathbf{x})$:={i: $x_i \le K_i - 1$} (the set of queues with less that K_i jobs),

$$\delta_i(\mathbf{x}) := \begin{cases} 0 & if\; i \in Q(\mathbf{x}) \\ 1 & if\; i \in P(\mathbf{x}) \end{cases},$$

$$\ell_i(\mathbf{x}) := \begin{cases} 0 & if\; i \in Q(\mathbf{x}) \\ x_i - K_i + 1 & if\; i \in P(\mathbf{x}) \end{cases},$$

$$\zeta_i(\mathbf{x}) := \begin{cases} 0 & if\; i \in Q(\mathbf{x}) \\ -K_i & if\; i \in P(\mathbf{x}) \end{cases}, \text{ and}$$

$L_i(\mathbf{x})$:=($\delta_i(\mathbf{x})$, $\ell_i(\mathbf{x})$, $\zeta_i(\mathbf{x})$).

The description of the optimal policy π^* may be obtained through the lexicographic ordering of the state index $L_i(\mathbf{x})$. We say that a vector $\mathbf{u}$:=(u_1, u_2, …, u_n) is lexicographically less than a vector $\mathbf{v}$:=(v_1, v_2, …, v_n) if and only if the following is true: $u_1 \le v_1$ and if $u_1 = v_1$ then $u_2 \le v_2$ and if $u_2 = v_2$ then $u_3 \le v_3$ and so on.

In other words, the optimal policy can be described as follow: for the routing problem and for mean processing times sufficiently long (i.e. for time consuming jobs) the policy that maximizes the probability that all queues have K_i or more occupied processors, in any time instant t, is the policy π^* and is described by the following steps:

<u>*Step 1*</u>: If there exist queues with $x_i \leq K_i - 1$ then assign the job to any one among them. Otherwise
<u>*Step 2*</u>: Assign the job to the queue with the minimum difference $x_i - K_i$. If there are two or more of those queues then
<u>*Step 3*</u>: Assign the job to the queue with the maximum K_i. If there are two or more of those queues then assign the job to any one among them.

C. Optimal assembly plan

The problem studied in this paragraph is the determination of the optimal plan among all feasible assembly plans. In other words, the assembler has to determine the capacities N_i ($N_i \geq K_i$) for all queues i and the amount of initial jobs allocated to each queue.

Let as assume that the following parameters are given: (i) $N>0$, the number of queues that will compose the system, (ii) $K_i>0$ $\forall i$, the threshold control level of each queue, (iii) the total number of processors T ($T = \sum_{i=1}^{N} N_i$), and (iv) the number of initial jobs G. Reference [10] examined the repair allocation functioning system with N subsystem connected in series and proved the optimal assembly plan of such system when the component failure rate is sufficiently small. The solution interpreted in the context of problem of routing time consuming jobs to parallel multiprocessor queues may be described as follows.

By renumbering the subsystems if necessary, the subsystems we may assume $K_1 \leq K_2 \leq \ldots \leq K_N$ without loss of generality. Let $S := T - \sum_{i=1}^{N} K_i$ be the surplus, $f := S$ *div* N and $h := S$ *mod* N, i.e. f is the integer part of S/N and h is the remainder from dividing S by N. Then, for service rate μ sufficiently small, the optimal assembly plan ($\mathbf{x}^*$, $\mathbf{N}^*$) is given by

$$\mathbf{N}^* = (K_1 + f, \ldots, K_{N-h} + f, K_{N-h+1} + f + 1, \ldots, K_N + f + 1)$$

and

(a) if the number of initial jobs $G < \sum_{i=1}^{N} K_i$ then $\mathbf{x}^*$ is any $\mathbf{x}$ such that $x_i \leq K_i$, $i=1, \ldots, N$, and ($\mathbf{x}$, $\mathbf{N}^*$) is feasible.

(b) if $G \geq \sum_{i=1}^{N} K_i$, then let $s := G - \sum_{i=1}^{N} K_i$, $g := s$ *div* N, and $e := s$ *mod* N. Then,

$$\mathbf{x}^* = (K_1 + g, \ldots, K_{N-e} + g, K_{N-e+1} + g + 1, \ldots, K_N + g + 1).$$

If we assume that there are no initial jobs in the systems, then the problem is to determine the capacities of queues only and the optimal system assembly plan tends to distribute the surplus component S equally among all queues and, when this is not possible, queues with larger control threshold K are favored.

Hence, for mean processing time sufficiently long (consuming jobs), if we start with the assembly plan $\mathbf{N}^*$ and follow the policy π^* to allocate the jobs in queues, we will obtain the maximum probability all queues have K_i or more occupied processors at any time instant t.

IV. CONCLUSIONS

There are two important problems in queuing networks of parallel queues (e.g. networks of computers-parallel processors, telecommunications-telephone networks etc). The first problem is the determination of the assembly plan of such systems, in other words, the determination of the number of processors of each queue, and the second, the control of the system, that is to say the routing policy in order to optimise some criterion.

In this paper the criterion that is used is the minimisation of total cost of such a system which is the sum of the services quality cost and server holding cost. The minimisation of total cost accomplished the maximisation of the probability all queues have more occupied processors than a control threshold, at any time instant t. In particular, for mean processing time sufficiently long (i.e. for time consuming jobs) the optimal allocation of processors and the optimal job assignment policy is described. Our future work will address optimizations for systems that the servers break down and repair.

ACKNOWLEDGMENT

This research is implemented through the Operational Program “Education and Lifelong Learning” and is co-financed by the European Union (European Social Fund) and Greek national funds. The contents of the paper reflect the views of the authors, who are responsible for the accuracy of the data presented herein.

REFERENCES

[1]. J.L. Hellerstein, K. Katircioglu, and M. Surendra, “An on-Line, business-oriented optimization of performance and availability for utility computing”, IEEE Journal on Selected Areas in Communications, vol 23(10), pp. 2013-2021, 2005.

[2]. C. Ward, M.J. Buco, R.N. Chang, and L.Z. Luan, “A generic SLA semantic model for the execution management of e-business”, Proc. 3rd Int. Conf. E-Commerce Web Technol., 2002, pp. 363-376.

[3]. W. Winston, “Optimality of the shortest line discipline”, J. Appl. Prob., vol. 14, pp. 181-189, 1977.

[4]. W. Whitt, “Deciding which queue to join: some counterexamples”, Operat. Res., vol. 34, pp. 55-62, 1986.

[5]. A. Hordijk, and G. Koole, “On the optimality of the generalized shortest queue policy”, Prob, Eng. Inf. Sci., vol. 4, pp. 477-487, 1990.

[6]. D.J. Houck, “Comparison of policies for routing customers to parallel queuing systems”, Operat. Res., vol. 35, pp. 306-310, 1987.

[7]. S.A. Lippman, “Applying a new device in the optimization of exponential queuing systems”, Operat. Res., vol.23, pp. 687-710, 1975.

[8]. S.M. Ross, Stochastic Processes. Wiley, New York, 1983.

[9]. V. Dinopoulou, and C. Melolidakis, “Asymptotically optimal maintenance of K-out-of-N systems connected in series”, Proc. 11th Conf Greek Statistical Inst., 1998, pp.67-76.

[10]. V. Dinopoulou, and C. Melolidakis, “Asymptotically optimal component assembly plans in repairable systems and server allocation in parallel multiserver queues”, Nav. Res. Logis, vol. 48(8), pp. 732-746, 2001.