A Scheduling Framework Applied to Digital Publishing Workflows*

Wilson Lozano† and Wilson Rivera‡
Parallel and Distributed Computing Laboratory
Electrical and Computer Engineering Department
University of Puerto Rico at Mayaguez
Mayaguez, Puerto Rico, USA 00680

ABSTRACT

This paper presents the advances in developing a dynamic scheduling technique suitable for automating digital publishing workflows. Traditionally scheduling in digital publishing has been limited to timing criteria. The proposed scheduling strategy takes into account contingency and priority fluctuations. The new scheduling algorithm, referred to as QB-MUF, gives high priority to jobs with low probability of failing according to artifact recognition and workflow modeling critera. The experimental results show the suitability and efficiency of the scheduling strategy.

Keywords: Digital publishing, dynamic scheduling, quality of service, scheduling algorithms.

1. INTRODUCTION

Digital publishing permits the linking of printing presses to computers, thereby bypassing the need for film and/or plate. As a result digital publishing offers the potential to raise the quality level for short-run printing, and enables the printing of documents that are highly variable in data content and layout. However, the realization of this potential has, to date, been seriously hampered by a number of difficulties. These include the problem of getting the document to print correctly without artifacts on the press and the difficulty of managing the increasingly complex workflow that results from shorter run jobs that must be completed in less time. Thus, digital publishing not only opens up new business but also requires new business models which lead to new workflow designs. The fact that information remains digital from the design stage all the way to printing leads to potential automation of processes that in traditional workshops are still manually executed. The typical stages in a digital publishing workflow are summarized in Table 1.

In this paper we discuss the advances in developing a dynamic scheduling technique suitable for automating digital publishing workflows. Traditionally scheduling in digital publishing has been limited to timing criteria. From our point of view, the scheduling process in digital publishing needs a new approach based on metrics targeting quality of service. In order to achieve such a scheduling methodology we have concentrated our research efforts in developing a scheduling algorithm that takes into account contingency (unexpected events) and priority fluctuations (changes in job priorities). Such a scheduling algorithm is a modification of the Maximum Urgent First[2] (MUF) algorithm. The new algorithm, referred to as QB-MUF, gives high priority to jobs with low probability of failing according to criteria defined in artifact recognition and workflow modeling and allows diverse scheduling policies.

The structure of this paper is as follows. Section 2 discusses the formulation of the scheduling problem in Digital Publishing and defines the scheduling strategy based on a new urgency criteria. Section 3 presents the experimental results. Section 4 discusses the related work. Finally, section 5 draws conclusions and future work.

†Graduate Assistant, wilson.lozano@ece.uprm.edu
‡Associate Professor, wilson.rivera@ece.uprm.edu
*This work has been supported by the Imaging and Printing Group (IPG) unit of Hewlett Packard Puerto Rico

Table 1. Digital Publishing Workflow Stages

Stage	Description
Pre-flight	Check if the digital document has all the elements requited to perform well in the production workflow. These elements include page file format, image resolution, font types, safety margins, mismatched colors.
Trapping	Overlap colors to compensate press registration. Register is the accurate positioning of two or more colors of ink in a printed sheet.
Imposition	Arrangement of individual pages on a press sheet, so that when it is folded and trimmed, the pages are in the correct orientation and order.
Proofing	Check an output before printed. Conventional: film-based; Soft proof: calibrated monitor; and digital proof (digital proofing printer).
Ripping	It decodes PostScript, creates an intermediate list of objects and instructions, and finally converts graphic elements into bitmaps for rendering on an output device.

2. A DYNAMIC SCHEDULING FRAMEWORK

Scheduling deals with the allocation of resources to tasks over time.[3] Tasks and resources can take different forms depending on the specific problem domain. From the Digital Publishing perspective, it is possible to identify printing PDF jobs as tasks, and the possible processes to execute over a job as resources.

The scheduling problem can be formulated as an optimization problem:

$$Minimize \sum_{j=1}^{n} w_j C_j \tag{1}$$

for a given set of tasks $\{\alpha_j\}_{j=1}^{n}$, where w_j is a weighted factor and C_j is a metric for each task $\alpha_j, 1 \leq j \leq n$.

The meaning of the objective function can be changed according to the specific problem domain. For example, the optimization problem can be the minimization of the aggregated completion time of a number of jobs, each pondered with a priority factor.

There exists a variety of scheduling models which adopt both deterministic and non-deterministic formulations. These models include single machine,[4,5] parallel machine,[6,7] and shop scheduling models.[8–12] Single machine model is the simplest type of scheduling models and a special case of all other environments. It is often found in practice when there is only one service point or a single stage. Algorithms developed for single machine models provide a basis for design of exact algorithms and heuristics for more complicated machine environments. Basic single machine models with regular objective functions are relatively simple and solvable via simple priority rules. More advanced single machine models deal with non-regular and/or multiple objective functions. These models are either solvable in polynomial time through dynamic programming or using polynomial time approximation schemes. A generalization of the class of single machine models is the type of parallel machine models. In parallel machine models a job requires a simple operation and may be processed on any of the m machines or on any one that belongs to a given subset of machines. The class of shop scheduling models comprises the open shop, flow shop and job shop models. In an open shop model, the operations of a job can be performed in any order. In a job shop, they must be processed in a specific job-dependent order. A flow shop is a special case of job shop in which each job has exactly m operations, one per machine, and the order in which they must be processed is same for all the jobs. Shop models are strongly NP-hard in their most general form. For the flow shop model, the case when there are more than two machines is strongly NP-hard, although the two machine version is polynomial solvable.

Production environments, such as print shops, are subject to many sources of uncertainty. Stochastic models[13,14] have been proposed as an alternative to model random features, such as job processing times, by specifying their probability distributions. Stochastic scheduling models, especially with exponential processing time, often

contain more structure than their deterministic counterparts. Consequently, models that are NP-hard in a deterministic setting often allow a simple priority policy to be optimal in a stochastic setting. In digital publishing, scheduling is not merely an activity that ensures on-time delivery, but a scientific tool that ultimately impacts print shop's profitability, customer satisfaction and competitiveness. Consequently, a major driving force for this research is the need of incorporating dynamicity into the scheduling process for digital publishing environments.

2.1. A Digital Publishing Scheduling Model

When a Job arrives to the Digital Publishing workflow, the first step is to extract static meta-data from the job and customer information to map it into a characterized job that can be managed by the scheduler system. These meta-data include an ID, the deadline of the job, levels of required quality, importance of the client and required resources from the system. The second step consists in determining other parameters such as the state of the job and latency according to the spare capabilities of the required resources. After a job path is defined based on the relevance of the job and the spare capabilities of the system, the workflow engine can move it from one resource to another to execute the required processes. Each resource/stage of the path releases meta-data that help calculate the spare capability of the system and a measure of probability of error for the next stages. Each resource/stage of the system has a local scheduler managing its own schedule into each stage according to expected and unexpected events presented at the system. Defining a model implies describing each component identified as part of the digital publishing workflow. The following sub-sections describe Job, Process and Resource in Digital Publishing scheduling.

2.1.1. Job

For our Digital Publishing model, a job is defined as an incoming PDF[§] file that requires the execution of a sequence of processes residents in the stages of the Digital Publishing workflow. Jobs are considered aperiodic, meaning job arrivals are not known a priori. Each Job has particular static parameters that describe it. These parameters are job arrival time (Ja_i), job deadline (Jd_i), and job relevance (Jr_i). Also there exist dynamic parameters which are calculated in real time according to the behavior of the system. These dynamic parameters are job latency ($Jlat_i$) and job laxity ($Jlax_i$). Job latency is the time required to complete a job. Job laxity is the difference between the job deadline and job latency at a specific time.

2.1.2. Process

A process can be defined as an action performed over a job. A job with a defined path requires, commonly in a specific order the realization of specific processes. Examples of processes in Digital Publishing are those described in Table 1. Processes are related to resources in the system, each resource can performs a specific process over a job. Furthermore, the system may have a heterogeneous sets of resources serving an unique process but on different levels. An example of this is a set of two different preflight tools, offering different levels of artifact recognition services.

2.1.3. Resource

A resource is an entity capable of executing a process over a job. A resource may be a software tool, an expert or a machine performing a process over a job. There exist different kinds of entities that can execute similar processes but with different specifications. An example in Digital Publishing is different preflight tools that may execute the preflight process but with different quality of outcomes.

Resources and processes can be formally defined as:

$$R = \{r_{jk_j}\} \; \forall \; j = 1, ..., m \; \wedge \; k_j = 1, ..., n_j,$$

where m is the number of processes in the system and n_j corresponds to the number of available resources to perform the process P_j. In this way, r_{jk_j} is an identifier of each autonomous resource that can execute the process P_j. The different number of elements in each heterogeneous set of resources that can execute the process P_j may be represented by the variation of k_j between 1 and n_j where n_j is different for each process P_j. On the

[§]Adobe Portable Document Format (PDF) is an open file format specification accepted as standard today by most of print-shops.

other hand, the resource property that affects the behavior of the system and the outcomes of the scheduling process is the load of the resource (Rl_{jk_j}). This is a measure that shows the load of the resource at specific time. The load of each resource can be managed by the scheduler in the process of generating a schedule for each job that require the resource.

2.2. A New Urgency Criteria

Our scheduling strategy focuses on providing high priority to jobs with low probability of failing. To achieve this an urgency criteria equation is introduced to account for relevance, laxities and probability of failures of incoming jobs. The proposed urgency criteria is based on one static parameter and two dynamic parameters (See Figure 1). These parameters are defined as follows.

1. Criticity (Relevance). This static factor is initially established by the user according to experience and/or customer importance. Criticity values range between 0 and 100.

2. QoS (Quality of Service). Scheduling involves matching of job needs with resource availability and capability and addressing the concern of the quality of the match. Different QoS metrics can be defined. In our Digital Publishing scheduling strategy QoS is defined by the probability of job success.

3. Laxity. This dynamic factor is defined as

$$Laxity = Jd - (t + Jlat),$$

where Jd is the Job deadline, t is the actual time of calculation, and Jl is the expected latency of the Job.

Laxity as defined above is not bounded and may conduct to unrealistic urgency criteria values. One way to compensate this is to define a modulator factor K (units of time). Using this modulator value, the Laxity factor is defined as follows.

$$LaxityF = \begin{cases} (K/Laxity) * C & if \quad Laxity > 0 \ and \ Laxity \geq K \\ C + (1 - (Laxity/K)) * (100 - C) & if \quad Laxity > 0 \ Laxity < K \end{cases} \tag{2}$$

where C is a constant that defines the maximum value (between 0 and 100) of LaxityF when $Laxity \geq K$ and the minimum value of LaxityF when $Laxity < K$. In the case that $Laxity < 0$, a degradation of quality of service on delivery time for a job will be permitted. Thereby a new deadline Jd must be generated but taking into account the probability of success of the job. The new Jd is calculated as $Jd = time_{now} + (f(Jlat) * MF)$. $f(Jlat)$ is a function of job latency. The multiply factor MF is calculated according to the QoS of the job as follows.

$$MF = \begin{cases} 0.5 & if \quad QoS \geq 80 \\ 0.7 & if \quad 60 \geq QoS < 80 \\ 0.8 & if \quad QoS < 60 \end{cases} \tag{3}$$

The resultant urgency criteria is defined as

$$Urgency = Criticity * W_1 + QoS * W_2 + LaxityF * W_3$$

$$\text{Where} \quad \sum_{i=1}^{3} W_i = 1 \ and \ , \ W_i > 0 \ \forall \ i$$

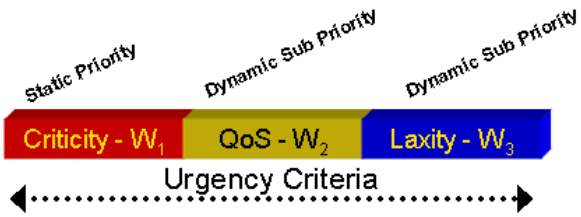

Figure 1. Urgency Criteria

2.3. The QB-MUF Algorithm

The QB-MUF algorithm iteratively assigns jobs to resources by considering resource availability and job urgency factors. If there are not resources available to process a job, the job is sent to a queue. The urgency criteria of the jobs in the queue are updated with certain periodicity to assure the flow of jobs with high probability of success. The general algorithm can be viewed as follows

```
While (There are jobs to schedule)
if (There are available resources)
    for each job i to schedule
        calculate job urgency;
        allocate job;
    end for
else
    if (Queue.length >0 )
        update urgency factor
    end if
    insert ordered job to the queue
end if
end while
```

2.4. Framework Architecture

The proposed urgency criteria and scheduling algorithm are embedded into a dynamic scheduling framework. The main goal of this framework is to provide a reusable infrastructure to design and evaluate scheduling strategies. An important feature is that the framework has the flexibility to be used in production environments other than digital publishing. The architecture of the framework is shown in Figure 2.

The Resource Manager gets the resource load information from a profiler placed inside of each stage. The profiler also maintains a forecasted load and calculates a tuning load. Such a tuning load is calculated from the real load and the forecasted load. The Global Scheduler uses the information gathered by the Resource Manager and static job information to generate scheduling events. A Job Monitor interacts with the Exchange Dispatcher and the Resource Manager to verify that the estimated conditions for scheduling are appropriate. If everything goes fine, the Dispatcher sends the job to its next step through the Workflow Engine. On the other hand, in the case of a failure, a contingency or a change in the schedule conditions occurs, the Dispatcher has to redirect the job according to predefined policies to manage exceptions. To reduce system overload, Local Schedulers are implemented at each stage to deal with local allocation of jobs over available resources.

3. EXPERIMENTAL RESULTS

We use GridSim[15] as simulation tool to implement and evaluate the QB-MUF algorithm. GridSim is a Java-based discrete-event grid simulation package, which allows modeling and simulation of entities in parallel and distributed computing systems. We took advantage of the GridSim capability for implementing new scheduling policies to deploy our QB-MUF algorithm and evaluate its performance.

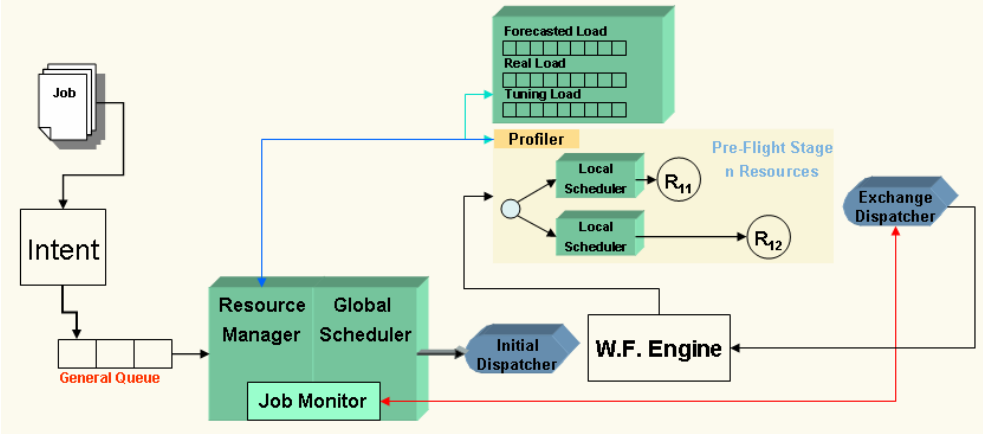

Figure 2. Framework Architecture

We assume that stages in a DP workflow provide useful information that can be used to forecast occurrence of faults and probability of failure of jobs at future stages. In this paper we focus our experiments on the preflight stage. A study over a variety of PDF documents analyzed by different preflighting tools was conducted to identify common faults in printing documents. According to this study, the mean probability of faults related to fonts not embedded is 67% and the mean probability of faults related to a wrong color base is 38%. This information is used to generate a flow of jobs with probability of faults around this values. Since the events are independent there may be a probability of having two faults in the same document.

Throughout the experimentation we compare the behavior of our QB-MUF algorithm with respect to two other scheduling approaches: The Minimum Laxity First, denoted as Laxity, and the well known First In First Out (FIFO) scheduling algorithm. Two metrics were observed throughout the experimentation. First, the number of successful jobs delivered while times were running. Second, the mean waiting time of successful delivered jobs for each scheduling algorithm. The whole set of experiments ran under the same conditions, except in those explicitly mentioned cases. The processing rate of preflighting resources is set to $10 bytes/sec$ and job sizes are set to $80Mb \pm 70\%$ according the job success probability.

The first set of graphics (Figures 3, 4 and 5) shows the behavior of the three scheduling algorithms, measuring the number of successfully jobs delivered each 250 units of time. Figure 3 shows the results for a total of 100 jobs. Figure 4 shows the results the same number of jobs but with an increased arrival rate of jobs. Figure 5 shows the results for a total of 200 jobs. We have conducted more experiments and similar results are obtained by increasing the number of jobs or the arrival rate. The QB-MUF algorithm outperforms both the Laxity and FIFO algorithms. We point out that for the experiments illustrated here the advantage of the QB-MUF algorithm is more evident around 9,000 units of time. This is because QB-MUF gives more importance to those jobs that have god expectations of finishing successfully.

The second set of graphics (Figures 6, 7 and 8) shows the mean of the waiting time for successful jobs using the three scheduling algorithms. The mean processing time is defined as the average of the time that a job has to wait since it was received until its start processing. Results show a reduction of waiting processing time of the QB-MUF over laxity and FIFO approaches.

4. RELATED WORK

There exists a number of commercial digital printing products that provide job tracking and scheduling capabilities including Production Manager from Hewlett-Packard, Lean Document Production (LDP) from Xerox, Print Shop Pro Manager from EDU, Pinnacle from Parsec, and Electronic Planning Board (EPB) from Pace Systems Group, among many others. To the best of our knowledge the only product that provides truly dynamic scheduling capabilities is PrintFlow from EFI. PrintFlow was developed around the Theory of Constraints

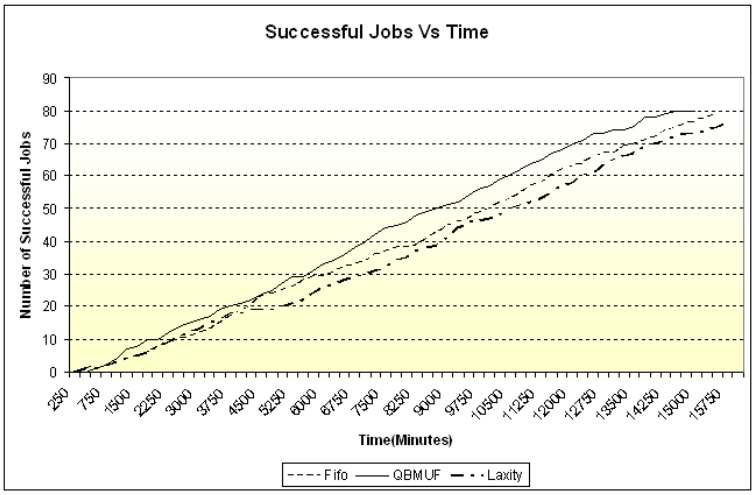

Figure 3. Number of successful jobs delivered; jobs=100; arrival rate=0.35

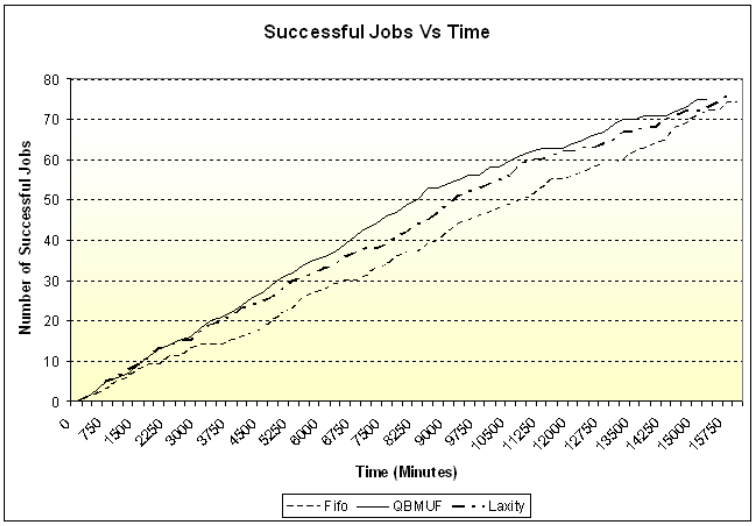

Figure 4. Number of successful jobs delivered; jobs=100; arrival rate=0.75

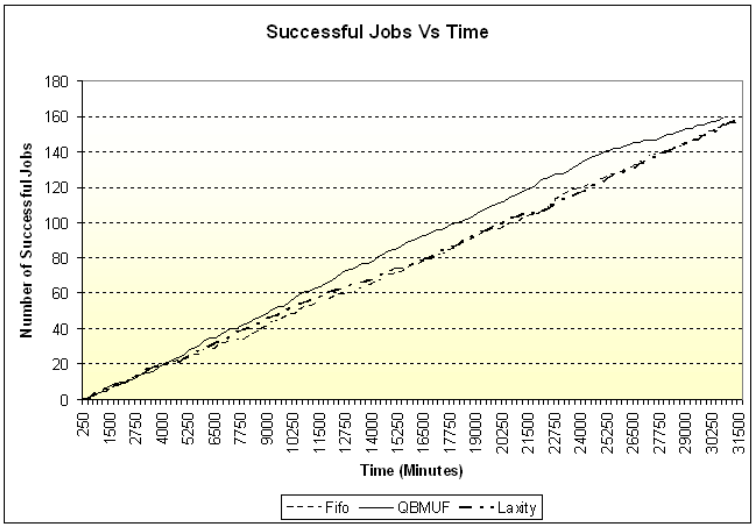

Figure 5. Number of successful jobs delivered; jobs=200; arrival rate=0.35

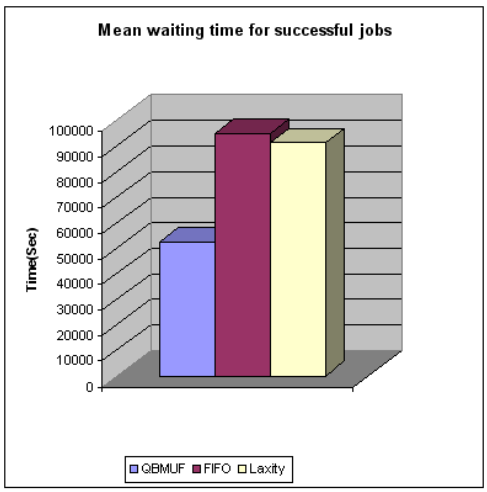

Figure 6. Mean waiting time; jobs=100; arrival rate=0.35

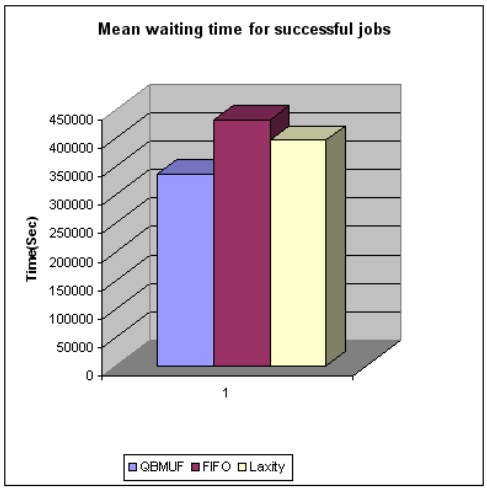

Figure 7. Mean waiting time; jobs=100; arrival rate=0.75

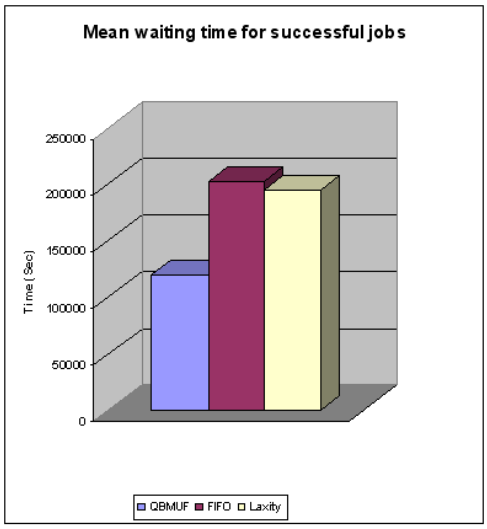

Figure 8. Mean waiting time; jobs=200; arrival rate=0.35

(ToC)[16] and was adapted to fit the printing industry. It defines printing as a manufacturing operation comprising interdependent links where only a few constraints control the throughput, on-time delivery and cost of the entire printing operation. We believe our approach, which diverges from the ToC approach, provides a more realistic scenario since it considers workflow priority fluctuations and contingency is a simplified formulation.

Stewart et. al.[2, 17, 18] proposed the Maximum Urgency First (MUF) algorithm as a flexible scheduler to support changing behaviors in sensor-based control systems. MUF gives to each job an urgency factor defined as a combination of two fixed priorities (criticality and user priority) and a dynamic priority (laxity). MUF combines the advantages of the Earliest Deadline First (EDF) and Minimum Laxity First (MLF) algorithms. EDF uses the deadline of a job as its priority. The job with the earliest deadline has the highest priority to be executed. MLF assigns a laxity to each job and selects the job with the minimum laxity to execute next. The difference between EDF and MLF is that MLF considers the execution time of a job, while EDF does not do. Kalogeraki et. al.[19] proposed a dynamic scheduling algorithm that monitors the computation times and resource requirements of a job to determine a feasible schedule of method invocations on processors. The schedule is driven by the laxities and the priority of the job. Ligang et. al.[20] proposed a dynamic framework with local and global schedulers based on the EDF criteria. In this approach jobs are rejected if their deadlines cannot be met under the condition of still guaranteeing the requirements of existing jobs. Zolfaghari et. al.[21] proposed an improvement for the MLF algorithm. Our approach is a departure from the above work providing a new formulation of the urgency criteria that includes information related to the relevance and laxities of the jobs as well as the probability of failure of jobs. The later factor is defined as a QoS metric and represents the major difference of our approach.

Hartmann et. al.[22] presented framework for data scheduling in packet-based wireless systems. This approach is based on the assumption that each user is characterized by a set of QoS requirements as his/her flow is accepted into the system. The authors define the residual time as a generalized measure for the urgency of the next packets over the flow. This residual time may be interpreted as a kind of laxity measure. Although the authors deal with QoS, the urgency of each packet is calculated base on the packet's residual time, leaving the QoS merely as a guide for the assignment of packets to the adequate cell. Our proposed QB-MUF algorithm deals with QoS parameters generated from the dynamic characteristics of each job, generating a QoS Factor that is included together with a Laxity Factor into the urgency of a Job.

Yuan et. al.[23] conveyed an analysis of the QoS properties in Manufacturing Grids (MG). A MG workflow can be defined as the composition of manufacturing activities executed on heterogeneous and distributed manufacturing resources. The authors presented a scheduling algorithm based on Qos. The main difference with our approach is that ours takes into account the dynamic internal characteristics of a job to calculate its probability of success.

An approach that includes the concept of QoS degradable jobs was presented by Mittal et. al.[24] They propose dynamic scheduling algorithms for integrated scheduling of hard and QoS degradable jobs in real-time multiprocessor systems. The real-time jobs are represented by two workload models, imprecise computation and (m,k)-firm guarantee, which quantify the trade-off between schedulability and result quality. This trade-off analysis will be introduced in our dynamic scheduling framework as an additional feature.

In summary, the main contribution of our approach compared to existing work is the introduction of a dynamic QoS factor representing the probability of success to drive the urgency criteria of jobs. The urgency factor is adapted to fit Digital Publishing workflows and allow the consideration of priority fluctuations and contingency.

5. CONCLUSIONS

A new formulation of the urgency criteria that includes information related to the relevance and laxities of the jobs as well as the probability of failure of jobs has been introduced. The new urgency criteria is used to design a scheduling algorithm that takes into account unexpected events and priority fluctuations. The new algorithm, referred to as QB-MUF, gives high priority to jobs with low probability of failing. Experimental results show that the QB-MUF algorithm outperforms both the Minimum Laxity First (MLF) and the First In First Out (FIFO) algorithms. A more complete analysis of the QB-MUF algorithm is in process on more complex scenarios.

ACKNOWLEDGMENTS

This work has been supported by a grant from the Image and Printing Group (IPG) of Hewlett-Packard (HP), Aguadilla, Puerto Rico.

REFERENCES

1. M. Kepler, *The Handbook of Digital Publishing*, Prentice Hall.
2. D. Stewart and P. Khosla, "Real-time scheduling of sensor-based control systems," *Proceedings 8th IEEE Workshop on Real-Time Operating Systems* , pp. 144–150, 1991.
3. M. Pinedo, *Scheduling - Theory, Algorithms and Systems*, Prentice Hall, 2002.
4. U. Al-Turki, C. Fedjki, and A. Andijani, "Tabu search for a class of single-machine scheduling problems, computers & operations research," pp. 1223–1230, 2001.
5. A. Agnetis, A. Alfieri, and G. Nicosia, "A heuristic approach to batching and scheduling a single machine to minimize setup costs, computers & industrial engineering," pp. 793–802, 2004.
6. H. Tamaki, T. Komori, and S. Abe, "A heuristic approach to parallel machine scheduling with earliness and tardiness penalties," *IEEE* , pp. 1367–1370, 1999.
7. Z. Chen and W. Powell, "Solving parallel machine scheduling problems by column generation," *Informs Journal of computing* , pp. 78–94, 1999.
8. D. Shmoys, C. Stein, and J. Wein, "Improved approximation algorithms for shop scheduling problems," *SIAM Journal of Computing* , pp. 617–632, 1994.
9. L. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk, "Better approximation guarantees for job shop scheduling," *8th ACM-SIAM Symposium on Discrete Algorithms(SODA)* , pp. 599–608, 1997.
10. S. Sevast'janov and G. Woeginger, "Makespan minimization in open shops : A polynomial time approximation scheme," *Mathematical Programming* , pp. 82(1–2), 1998.
11. M. Sviridenko, K. Jansen, and R. Solis-Oba, "Makespan minimization in job shops: A polynomial time approximation scheme," *31st Annual ACM Symposium on Theory of Computing* , pp. 394–399, 1999.
12. S. Yang and D. Wang, "Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling," *IEEE Transactions on Neural Networks* , pp. 474–486, 2000.
13. S. Elmaghraby, "On the optimal release time of jobs with random processing times, with extensions to other criteria," *International Journal of Production Economics* , pp. 103–113, 2001.
14. D. Golenko-Ginzburg and A. Gonik, "Optimal job-shop scheduling with random operations and cost objectives," *International Journal of Production Economics* , pp. 147–157, 2002.
15. A. Sulistio, G. Poduvaly, R. Buyya, and C. Tham, "Constructing a grid simulation with differentiated network service using gridsim," *Proceedings of The 6th International Conference on Internet Computing (ICOMP'05)* , 2005.
16. E. Goldratt, *The Goal*, The North River Press, MA 1984.
17. D. Stewart, D. Schmitz, and P. Khosla, "Implementing real-time robotic systems using CHIMERA II," *Proceedings of The IEEE International Conference on Robotics and Automation* , pp. 598–603, 1990.
18. D. Stewart and P. Khosla, "Real-time scheduling of dynamically reconfigurable systems," *IEEE International Conference on Systems Engineering* , pp. 139–142, 1991.
19. V. Kalogeraki, P. Melliar-Smith, and L. Moser, "Dynamic scheduling of distributed method invocations," *Proceedings of The 21st IEEE Real-Time Systems Symposium* , pp. 57–66, 2000.
20. L. He, S. Jarvis, D. Spooner, and G. Nudd, "Dynamic scheduling of parallel real-time jobs by modeling spare capabilities in heterogeneous clusters," *Proceedings of The IEEE International Conference on Cluster Computing* , 2003.
21. B. Zolfaghari, "A dynamic scheduling algorithm with minimum context switches for spacecraft avionics systems," *Proceedings of IEEE Aerospace Conference* , pp. 2618 – 2624, 2004.
22. C. Hartmann and R. Vilzmann, "Urgency based scheduling for user-individual qos in cellular mimo-systems," *ITG Workshop on Smart Antennas* , pp. 257– 260, 2004.
23. Y. Yuan., T. Yu., F. Xiong., and M. Fang, "Qos-based dynamic scheduling for manufacturing grid workflow," *Ninth International Conference on Computer Supported Cooperative Work in Design* , pp. 1123– 1128, 2005.

24. A. Mittal, G. Manimaran, and C. Murthy, "Integrated dynamic scheduling of hard and qos degradable real-time tasks in multiprocessor systems," *Fifth International Conference on Real-Time Computing Systems and Applications* , pp. 127–136, 1998.