Storage Requirements for Fair Scheduling*

Michael J. Fischer
Yale University
New Haven, Connecticut

and

Michael S. Paterson
University of Warwick
Coventry, England

Manuscript Date
August 1982

Report Date
October 1982

Keywords: Fair scheduling, analysis of algorithms, storage bounds, parallel computation

*This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, and by the National Science Foundation under Grant MCS-8116678.

1. Introduction

In [3], Park discusses notions of strong and weak fairness in the execution of guarded iterations. These concerns are also considered in [1] and [2]. We show that any "strongly fair" scheduling algorithm for n processes requires at least n! storage states (i.e. space proportional to n log n). Similarly, any "weakly fair" scheduling algorithm requires at least n storage states. Both bounds are optimal.

For our present purposes we may define a *scheduler* as a transducer A with an input alphabet of symbols corresponding to the non-empty subsets of $\{1, ..., n\}$ and output alphabet $\{1, ..., n\}$. It has the property that for each symbol input the generated output symbol is an element of the corresponding subset. We may regard each input symbol as requests for service from some subset of n processes and the output given by A as the scheduler's choice of which one of these to *serve*. We consider infinite runs of such a scheduler.

A scheduler is

1. *strongly fair* if each process which requests service infinitely often is served infinitely often, and
2. *weakly fair* if each process which requests service all but finitely often is served infinitely often.

Thus at any time in a strongly (weakly) fair schedule any process will eventually be served if it requests service infinitely (continuously) from that time. Park's example of a strong scheduler in [P] keeps the processes in a queue. At each step it serves that requesting process which is earliest in the queue and then sends this process to the back of the queue. That this provides strongly fair scheduling is easy to see since when any process is unsuccessful in its request it advances one position in the queue. Park expresses disquiet at the implementation overheads for such a scheduler.

By contrast, he shows a simple economical weakly fair scheduler. A counter with values in $\{1, ..., n\}$ is maintained. At each step the counter is incremented modulo n until it reaches the number of a process requesting service. This process is then served.

We shall show that both of the schedulers given by Park are optimal in their use of storage space.

Proof. Consider the (constant) input sequence in which each process requests service at every step. If the scheduler has fewer than n states, its resulting ultimately periodic behaviour has period less than n and so fails to serve some processor. □

Acknowledgement

We are grateful to David Park for introducing us to this problem.

References

[1] K. R. Apt and E.-R. Olderog. Proof rules dealing with fairness. Bericht Nr. 8104, Institut für Informatik u. Praktische Mathematik, Kiel University (1981).

[2] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: the ethics of concurrent termination. In *Automata, Languages and Programming*, S. Even and O. Kariv, eds., Lecture Notes in Computer Science Vol. 115, Springer-Verlag, 1981, 264-277.

[3] D. Park. A predicate transformer for weak fair iteration. *Proc. Sixth IBM Symp. on Mathematical Foundactions of Computer Science*, IBM Japan (1981), 257-275.