ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2013 Prof. R. Wattenhofer / K.-T. Foerster, T. Langner, J. Seidel

Discrete Event Systems

Exercise Sheet 14

1 Power-Down Mechanisms

Since Alice has not been very happy with her internship at the processor manufacturer *Tintel*, she has applied for another internship at the company *Hewlate-Packard*, which is well-known for its energy-saving laptop models. The company's intelligence division has recently acquired a laptop from their strongest competitor *Samson* which seems to outperform their own models. Alice's first assignment is to investigate further into this matter.

The state-of-the-art laptops of both Hewlate-Packard and Samson have two power modes:

- *Active state*: Energy consumption is 1 energy unit per time unit
- *Sleep state*: Energy consumption is 0 energy unit per time unit

The transition from the active state to the sleep state and back to the active state costs D energy units altogether. Over time, there are busy periods where the user works with the device and idle periods where the laptop is not used. During busy periods, the laptop must be in the active state.

The **power-down strategy** controls the idle time after which the laptop goes to sleep. The competitiveness of a strategy S is the ratio of the energy units spent by S and the energy units spent by the optimal algorithm for a given busy sequence.

Hint: To show that a strategy is c-competitive, it suffices to show that it is c-competitive for an arbitrary idle period $I = [t_1, t_2]$. At time $t_1 - \varepsilon$, both algorithms have to be in the active state because the busy period has not ended yet. At time $t_2 + \varepsilon$, both algorithms have to be in the active state again because a busy period has just begun. Hence, it suffices to compare the energy consumption of ALG and OPT within the time span $[t_1, t_2]$ (energy costs for going to sleep state and back to active state are part of the costs for I).

a) To get familiar with power-down strategies, Alice has to find a deterministic 2-competitive power-down strategy.

b) Alice's supervisor Brian tells Alice that nobody in the company has yet found a strategy that is better than 2-competitive. Prove that no deterministic strategy can be better than 2-competitive.

c) Brian is unhappy about Alice's findings from part **b)** because he knows that the power-down strategy in Samson's laptops is better than 2-competitive. Alice has to find a strategy with a competitive ratio $c < 2$ otherwise she will be fired. Help Alice!

Hint: Alice recently heard her colleagues talking about probabilistic strategies...

2 PhD-Scheduling [Exam]

In this assignment we wish to study the PhD-Scheduling-Problem, in which Professor Arno Nym wants to assign some of his more cumbersome tasks to his m PhD students. When Prof. Nym enters his office in the morning there is a pile of tasks on his desk, and he is only able to see the topmost task at a time. He wants do decide *online* to which of his m PhD students he assigns the topmost task before taking a look at the next task. Prof. Nym does not know in advance when the last task is reached, because there might be more tasks hidden beneath the topmost one. Every task needs to be assigned to exactly one PhD student, and every PhD student is able to solve any task. Of course, Prof. Nym would like all tasks to be finished as fast as possible.

A task is defined by its *effort* $e_i \in \mathbb{R}^+$, independent of the PhD student who processes the task, and different tasks may require different effort. The *load* L_j of a PhD student j is defined by the sum of the efforts of all tasks assigned to j. He thus needs time L_j to finish all assigned tasks. Prof. Nym's goal is that all tasks are finished in as little time as possible, which means that the maximum load of any PhD student is to be minimized. For a sequence of tasks $\sigma = (e_1, e_2, \ldots)$, the cost of an assignment $\mathcal{A}$ of tasks to students is thus defined by

$$\text{cost}(\mathcal{A}(\sigma)) := \max_{j=1,\ldots,m} L_j(\mathcal{A}(\sigma)).$$

In the following we analyze the algorithm SmallLoad that assigns the topmost task to a student whose load is minimal with respect to the previous assignments.

a) Assume that Prof. Nym has only two PhD students, i.e., $m = 2$.

(i) Describe the execution of SmallLoad for the following sequence of tasks σ.

$$\sigma = (2, 5, 4, 3, 7)$$

What would be an optimal solution for this sequence? How large is the competitive ratio of SmallLoad with respect to σ?

(ii) The cost of a solution found by SmallLoad can be worse than that of an optimal solution. Construct a sequence of tasks σ' for which the costs of SmallLoad are as high as possible compared to the costs of an optimal solution.

(iii) Is the algorithm SmallLoad c-competitive for some constant c? If so, give the smallest possible such c. Prove your claim!

(iv) Is there an efficient way to compute an optimal *offline* solution? Explain your answer.

b) Now, analyze the case in which the number of PhD students m is arbitrary. What is the smallest competitive ratio of SmallLoad now? Prove your claim!