Written Assignment #4

CPEN 432: Real-time System Design

Notation: $\mathbb{N} = \{1, 2, \dots\}$.

1. **[10 points]** (-∿∿-)

   (a) **[5 points]** Three soft real-time tasks have to be scheduled on the same processor. Two tasks have fixed execution times but their periods/frequencies are flexible. The third task has a fixed periodicity but variable execution times because of supporting imprecise execution. The parameters of the tasks are as in the following table (Table 1).

Task	$e_{\min}$	$e_{\max}$	e_0	$P_{\min}$	$P_{\max}$	P_0	k
T_1	2	2	2	8	12	10	1
T_2	4	4	4	5	10	6	2
T_3	4	10	5	12	12	12	1.5

Table 1: Task set for question 1a

   k is the elasticity of each task and e_0 and P_0 are the *nominal* execution times and nominal periods for the tasks. Assume that elastic scheduling is used. What then are the execution parameters of these three tasks? Recall that the compression formula for elastic scheduling is as follows:

$$\forall T_i \in V : U_i = U_{i,0} - (U_0 - U_b + U_M)\frac{k_i}{\sum_{T_j \in V} k_j}.$$

   Solve the problem for EDF and RM scheduling with the LL bounds.

   (b) **[5 points]** Show that, under the assumption that task execution times are integers (or can be represented by integers), the greedy procedure we used to solve the LP of the imprecise computation model is optimal. That is, the greedy procedure, upon termination, produces execution times $e_1, \dots, e_n$ that satisfy the constraints of the LP and minimize the total QoS loss $\sum_{i=1}^{n}(e_{i,\max} - e_i)$.

   (c) (**Bonus: +3 points**) Does the same greedy procedure yield optimal solutions to the imprecise computation LP if we wish to minimize the mean square loss in QoS instead? What about the stronger maximum loss $\max_i(e_{i,\max} - e_i)$? What is the most general class of error functions $F \equiv F(e_i, e_{i,\min}, e_{i,\max})$ for which the greedy procedure yields optimal solutions?

2. **[10 points]** Packet processors are used in most commercial networking equipment to support a variety of traffic management and monitoring functionality. Examples of such functionality include packet classification, spam detection, virus and malware blockers and encryption support.

Consider a simple packet processing system with three stages: a packet classification stage, a spam detector, a virus signature analyzer, and a cryptography module. All traffic enters the packet classifier and packets are identified as email, general application and VoIP traffic (in 2 time units). Email traffic is routed through the remaining three stages and each email packet requires 4 time units, 5 time units and 3 time units respectively at the three stages. General application traffic does not require spam detection but does require virus scanning and encryption; these packets require 6 time units and 7 time units respectively at these two stages. VoIP traffic requires encryption only: 3 time units.

Suppose that email traffic arrives at the rate of 1 packet every 20 time units (traffic shapers can ensure this regular arrival rate), general application traffic arrives at the rate of 1 packet every 12 time units, and VoIP traffic arrives at the rate of 1 packet every 15 time units. The priority levels for the different classes of traffic are such that VoIP > general application > email.

(a) **[5 points]** What is the maximum delay experienced by a packet of email traffic? (You may make reasonable assumptions to determine this delay.)

(b) **[5 points]** What is the [approximate] maximum email traffic rate that this packet processing system can support assuming that all non-email traffic arrives at the fixed rate specified?

3. **[10 points]** Consider n non-preemptable jobs $J_1, \ldots, J_n$ that are all available at time 0. Job J_i has WCET $e_i \in \mathbb{N}$ and worst case (peak) memory requirement $M_i \in \mathbb{N}$ (in MB). Once assigned to processors, jobs cannot migrate to other processors. Moreover, the order of the jobs is fixed by the job dispatching system and so cannot be rearranged: In any allocation of processors to jobs, the jobs must appear in the same order as they are given in the input. Assume that the jobs are ordered according to their indices such that J_1 is the first job to be assigned, and if J_i is assigned processor k, then job J_{i+1} must either execute on processor k and start after J_i or execute on the $(k+1)$st processor. The processors onto which the jobs are to partitioned are identical and each has maximum available time $L \in \mathbb{N}$; that is, the sum of execution times of the jobs assigned to any processor cannot exceed L. You may assume that $e_i \leqslant L$ for every $i \in \{1, \ldots, n\}$.

   Given an allocation of processors to the input jobs, the peak memory demanded by processor π_j under the given allocation is defined as

$$\max\{M_i : J_i \text{ is assigned to processor } \pi_j\}.$$

   Our goal is to assign the jobs to as many processors as needed, respecting the given processor available time and the no rearrangement constraint, so that the overall peak memory usage is minimized; that is, the sum of the peak memory used on all processors is kept at a minimum. Develop an algorithm to solve this problem optimally and prove its correctness. Derive the asymptotic running time of your algorithm in terms of $n, \{e_i\}, \{M_i\}$, and L (or any subset thereof). *You will receive full credit only if your algorithm runs in time that is* ***polynomial*** *in the size of the input.*

   HINT: Use Dynamic Programming. Convince yourself, but not the grader, that the greedy algorithm that stuffs each processor as full as possible does not always give the minimum overall peak memory usage. An example exists.

4. **[10 points]** We need to execute three periodic tasks on a partitioned multiprocessor system using EDF at each processor. The utilizations of the three tasks are 0.4, 0.3 and

0.5, respectively. We will consider reliability over a fixed interval, and thus we will drop time from the reliability requirements. Assume that a processor has a reliability of 0.95 over the reliability interval under consideration. Failure of a processor implies failure of all tasks running on the processor (for a suitable notion of task failure), and the processors are not repairable. We would like the reliability of each task to be at least 0.999. How many processors do we need to ensure this level of reliability? How should tasks be partitioned to obtain this reliability level? Here we are allowed to replicate tasks, but jobs belonging to one replica execute on the same processor.

5. **[10 points]** Consider a device whose failure is governed by the exponential law with rate $\lambda > 0$ (hours/failure). The device is repairable; it is replaced with a new device (i.e., starts anew) as soon as it fails (i.e., replacement starts as soon as failure happens with zero delay). The time to replace the device is exponential with rate $\mu > 0$ (hours/replacement). Now, assume that the system is requested to operate with an availability greater than 0.999 but at the same time with a reliability greater than 0.9 over an interval of 1000 hours. Are the reliability and availability requirements both achievable simultaneously? If so, for what values of μ and λ?