jian-jia.chen [☺] tu-dortmund.de
georg.von-der-brueggen [☺] tu-dortmund.de
wen-hung.huang [☺] tu-dortmund.de
jan.kleinsorge [☺] tu-dortmund.de

Exercises for Lecture
Real-Time Systems and Applications
Summer Semster 15

Exercise Sheet 9

(11 Punkte)

Exercise Due at Wednesday, July 15, 2015, 12:00 Uhr

Hinweise: Gruppenarbeit von bis zu drei Personen aus der gleichen Übungsgruppe ist möglich. Bitte vergessen Sie nicht Ihre Namen und Ihre Matrikelnummern auf die Lösung zu schreiben. **Die Abgaben können in den beschrifteten Briefkasten vor dem Sekretariat des LS12 (OH16/E22) eingeworfen oder per Mail (PDF Format) an georg.von-der-brueggen [☺] tu-dortmund.de abgegeben werden.**

Note: It is allowed to work in a group of up to three persons, if these persons are from the same practice group. Please do not forget to write your name and your Matrikelnummer on the solutions. **The solutions can either be placed in the mailbox in front of the secretary's office of LS 12 (OH/E22) or sent by mail (PDF format) to georg.von-der-brueggen [☺] tu-dortmund.de**

Exercise Sessions:

Do, 10:15 - 11:45	OH16/E18
Do, 14:15 - 15:45	OH16/E18

9.1 Multiprocessor Scheduling (4 Punkte)

1. Explain and compare Partitioned, Semi-Partitioned, and Global multiprocessor scheduling for sporadic real-time tasks. What are the corresponding advantages and disadvantages?
2. Explain the Makespan problem and the Bin Packing problem. What are their connections to the multiprocessor partitioned scheduling problem for sporadic real-time tasks with implicit deadlines?
3. Mr. Smart wants to define the monotonicity of the largest utilization first (LUF) algorithm. He has two options in his mind:
   - If LUF derives a feasible task partition (with respect to schedulability under EDF) for an input task set, LUF also gives a feasible task partition if the execution time of certain task is reduced.
   - If LUF derives a feasible task partition for an input task set, the resulting task partition remains feasible (with respect to schedulabilty under EDF) if the execution time of certain task is reduced.

   Help him clarify the correctness of these statements.
4. Mr. Smart wants to decide the number of processors he needs to schedule a set of sporadic real-time tasks with implicit deadlines. He tries to use Global EDF and Global RM. The total utilization $\sum_{\tau_i} \frac{C_i}{T_i}$ of the given task set is a constant. But, no matter how many processors he adds (less than the number of tasks), he can always find an execution pattern that is not schedulable under Global EDF or Global RM. Give an example of such a set of tasks. (**Hint:** Dhall's effect.)

Hint: 1+1+1+1

9.2 Hyperbolic-Bound for Task Assignment (3 Punkte)

Implicit-Deadline Task Partitioning: Task partitioning on M identical multiprocessor systems for rate-monotonic scheduling (implicit-deadline sporadic tasks) can be done by using the hyperbolic bound $\Pi_{\tau_i \in \mathcal{T}_m}(U_i + 1) \leq 2$, where $\mathcal{T}_m$ is the set of tasks assigned on processor m.

- Describe a task partitioning algorithm by extending the LUF algorithm in the lecture and using the hyperbolic bound to decide whether a task can be assigned on a processor m.
- Explain the condition
$$(U_k+1)^M \Pi_{i<k}(U_i+1) > 2^M$$
of failure of the above algorithm when assigning task τ_k on M processors, in which the task assignment starts from a predefined order $\tau_1, \tau_2, \ldots, \tau_k$.
- Explain that the speed-up factor of the above algorithm is 2.668 when k is sufficiently large. (*Hint: the solution x for the following equation $(1+1/x)\cdot e^{1/x} - 2 = 0$ is about 2.668, where e is Euler's number. The factor 2.414 can be achieved if the tasks are sorted according to their utilizations (non-increasingly).*)

Hint: 1+1+1

9.3 Semi-Partitioned Scheduling (3 Punkte)

Mr. Smart designs a semi-partitioned scheduling algorithm for sporadic real-time tasks $\mathcal{T}$ on two processors as follows:

- The first processor handles at most $C_{i,1}$ amount of execution time for task τ_i.
- The second processor handles at most $C_{i,2}$ amount of execution time for task τ_i.
- $C_{i,1} + C_{i,2}$ is the worst-case execution time for task τ_i.
- The system has a static priority ordering for executing tasks on the first processor, and also a static priority ordering for executing tasks on the second processor.
- The scheduling algorithm migrates the tasks as follows: when the first processor executes τ_i for $C_{i,1}$ amount of time, the remaining computation is *immediately* passed to and handled by the second processor under the static-priority scheduling.
- *Mr. Smart claims that the worst-case response time for τ_i in the above algorithm is $R_{i,1} + R_{i,2}$ if $R_{i,1} + R_{i,2} \leq T_i$, where*
  - $R_{i,1}$: $\min 0 < t \leq T_i, \quad C_{i,1} + \sum_{j \in hp_1(i)} \left\lceil \frac{t}{T_j} \right\rceil C_{j,1}$
  - $R_{i,2}$: $\min 0 < t \leq T_i, \quad C_{i,2} + \sum_{j \in hp_2(i)} \left\lceil \frac{t}{T_j} \right\rceil C_{j,2}$
  - $hp_1(i)$ ($hp_2(i)$, respectively) is the set of tasks that have higher priority than τ_i on the first (second, respectively) processor.
- Is his claim correct? Why or why not? If not, how can we fix it (either the scheduling algorithm or the worst-case response time analysis)?

9.4 Challenge on Partitioned Multiprocessor Scheduling (1 Punkt)

Mr. Smart claims the following statement when considering task partitioning in identical multiprocessor systems: "If there exists a polynomial-time algorithm A which guarantees an augmentation factor for using at most ρ times processors, this algorithm can be converted to a polynomial-time algorithm with a speed-up factor $\lceil \rho \rceil$." Is he correct? Explain your support or disprove the argument.

General Hints: