Proofs

H.J. Rivera Verduzco 0977393

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Email: H.J.Rivera.Verduzco@student.tue.nl

May 3, 2017

1 Blocking tasks

Lemma 1. *Given a task-set $\mathcal{T}$ of n independent tasks scheduled using FPTS, and a level-n busy period $[t_s, t_f]$ of such a schedule. If for a new task-set $\mathcal{T} \cup \{\tau_{n+1}\}$, τ_{n+1} is scheduled in such a way that a job k of τ_{n+1} executes within an interval $[t_1, t_s)$ where there is exactly an amount of time C_{n+1} available for the execution of τ_{n+1}, then the schedule of all jobs in $[t_s, t_f)$ remains the same after introducing τ_{n+1}.*

Proof. Assume that task τ_{n+1} is scheduled in such a way that a job k of τ_{n+1} executes within an interval $[t_1, t_s)$ where there is exactly an amount of time C_{n+1} available for the execution of τ_{n+1}. After introducing τ_{n+1}, the schedule in $[t_s, t_f)$ can be guaranteed to remain unchanged if there is no pending load at time t_s from higher priority tasks of τ_{n+1}, i.e. if $P_n(t_s) = 0$. In addition, there should not be pending load at time t_s from a job of τ_{n+1} that has already started. We show that these two conditions are always met for the given assumptions.

$\{P_n(t_s) = 0\}$. First note that, since there is exactly an amount of time C_{n+1} available for the execution of τ_{n+1} in the interval $[t_1, t_s)$, job k of τ_{n+1} can fit exactly in such an interval. Therefore, the higher priority tasks executing in $[t_1, t_s)$ will still finalize before or at t_s, and consequently the pending load $P_n(t_s)$ will be zero.

{*Pending load of already started jobs of τ_{n+1} at time t_s is zero*}. Since we assumed that job k executes within $[t_1, t_s)$ and therefore it finalizes before or at t_s, we only have to prove that next jobs of τ_{n+1} cannot start before t_s. Note that job k of τ_{n+1} already occupies all the time that was idle in $[t_1, t_s)$. Therefore, the start time of any job of τ_{n+1} after job k activated before t_s will be postponed after t_f.

□

Lemma 2. *Given a task-set $\mathcal{T}$ of n independent tasks scheduled under FPTS, a level-n busy period $[t_s, t_f]$ of such a schedule, and a new task-set $\mathcal{T} \cup \{\tau_{n+1}\}$ where $U^{\mathcal{T} \cup \tau_{n+1}} < 1$ or $U^{\mathcal{T} \cup \tau_{n+1}} = 1$ and the lcm of the periods exists. It is always possible for task-set $\mathcal{T} \cup \{\tau_{n+1}\}$ to schedule τ_{n+1} in such a way that a job k of τ_{n+1} executes within an interval $[t_1, t_s)$ where there is exactly an amount of time C_{n+1} available for the execution of τ_{n+1}.*

Proof. Since $U^{\mathcal{T} \cup \tau_{n+1}} < 1$ or $U^{\mathcal{T} \cup \tau_{n+1}} = 1$ and the lcm of the periods exists, it is always possible to find a contingent interval $[t_1, t_s)$ before the level-n busy period $[t_s, t_f]$ where there is an amount of time C_{n+1} available for the execution of τ_{n+1}. The only reason for which a job k of τ_{n+1} may not be able to execute within $[t_1, t_s)$ is that it could experience some blocking from previous jobs delaying the start time of such a job k. However, it is possible to change the phase of ϕ_{n+1} in order to move the activations of the jobs of τ_{n+1} at an earlier moment in time, and therefore allow enough space before $[t_1, t_s)$ for the execution of the earlier jobs of τ_{n+1} before job k that may delay its start time. Since the number of jobs of τ_{n+1} that may delay the start time of job k is bounded by the number of jobs in the worst-case length of the level-$\{n+1\}$ active period, it is always possible to find enough space before $[t_1, t_s)$ for their execution. □

Lemma 3. *Let all tasks of a set $\mathcal{T}$ be strictly periodic and let $U^{\mathcal{T}} < 1$ or $U^{\mathcal{T}} = 1$ and the lcm of the periods exists. Under FPTS, the best-case response time BR_i of a task $\tau_i \in \mathcal{T}$ is not influenced by its lower priority tasks.*

Proof. Assume that we find a schedule for a subset of task-set $\mathcal{T}$ defined as $\mathcal{T}_i = \{\tau_a | a \leq i, \tau_a \in \mathcal{T}\}$ where the *best-case response time* BR_i of τ_i is assumed by job k^{bcrt} in the level-i busy period $[t_s, t_f]$. Note that task-set $\mathcal{T}_i$ contains the same tasks as $\mathcal{T}$ without the lower priority tasks of τ_i. Given lemmas 1 and 2, we can construct a schedule for a task-set $\mathcal{T}_{i+1}$ where task τ_{i+1} is scheduled in such a way that a job k executes in a contingent time interval $[t_1, t_s)$ before $[t_s, t_f)$. Hence, the schedule in time interval $[t_s, t_f)$ will remain unchanged and the *best-case response time* BR_i of τ_i will remain the same in $\mathcal{T}_{i+1}$. Furthermore, note that since job k of τ_{i+1} occurs in a contingent time interval $[t_1, t_s)$ before $[t_s, t_f)$, these two time intervals are contained in a level-$\{i+1\}$ busy period $[t\prime_s, t\prime_f]$; hence, job k^{bcrt} of τ_i occurs in $[t\prime_s, t\prime_f]$ as well. Based on this observation and using lemmas 1 and 2 again, we can find a schedule for a task-set $\mathcal{T}_{i+2}$ where the jobs in the level-$\{i+1\}$ busy period $[t\prime_s, t\prime_f]$ remain unchanged and, therefore, BR_i is preserved. Since we can continue constructing schedules preserving the *best-case response time* of τ_i until $\mathcal{T}_{i+l} = \mathcal{T}$, we conclude that the *best-case response time* BR_i of a task τ_i is not affected by its lower priority tasks. □

2 Higher priority tasks

Lemma 4. *Let $\mathcal{T}$ be a set of strictly periodic tasks with a task $\tau_i \in \mathcal{T}$. The threshold of higher priority tasks do not influence the response time of a job of τ_i.*

Proof. In order to prove Lemma 4, first note that the response time $R_{i,k}$ of a job $\iota_{i,k}$ of τ_i is equal to $S_{i,k} + H_{i,k}$. Therefore, it is sufficient to show that the thresholds of higher priority tasks do not have an influence on the start time $S_{i,k}$ and the hold time $H_{i,k}$.

{*Influence on $S_{i,k}$*}. Under FPTS, a necessary condition for a job $\iota_{i,k}$ to start at a time t after its activation is that the priority of τ_i is the highest among all tasks with pending load at time t. Since this condition only depends on the priority of the tasks, higher priority tasks will always affect the start time $S_{i,k}$ of $\iota_{i,k}$ independently of their thresholds.

{*Influence on $H_{i,k}$*}. Under FPTS, a job $\iota_{i,k}$ of a task τ_i can only be preempted by a higher priority task τ_h iff $\pi_h > \theta_i$. Since this condition is independent of the threshold θ_h of τ_h, we conclude that the thresholds of higher priority tasks of τ_i do not have an influence on the hold time $H_{i,k}$ of $\iota_{i,k}$. □

3 BCRT for non-preemptive tasks

Lemma 5. *Let τ_i be a non-preemptive task. The best-case response time of τ_i is always equal to its best-case computation time, i.e. $BR_i = BC_i$.*

Proof. Given a task-set $\mathcal{T}$ and a non-preemptive task $\tau_i \in \mathcal{T}$, assume that all jobs of τ_i have a computation time equal to BC_i. Since τ_i is non-preemptive, the hold time of all its jobs is simply equal to BC_i. Recall that the response time of a job k of τ_i is given by $R_{i,k} = S_{i,k} + H_{i,k}$; therefore, in order to prove the lemma, it is sufficient to prove that it is always possible to schedule the tasks in such a way that the relative start time of a job k of τ_i is zero, i.e. $S_{i,k} = 0$. We divide this proof in two cases:

{*Case $U^{\mathcal{T}} < 1$*}. Given an arbitrary schedule for $\mathcal{T}$, let $[t_s, t_e)$ be an idle interval in such a schedule. Note that it is always possible to find an idle interval because $U^{\mathcal{T}} < 1$. Furthermore, let $\iota_{i,k}$ be the first job of τ_i activated at or after time t_e, i.e. $t_e \leq a_{i,k}$ and there are no jobs of τ_i activated in $[t_e, a_{i,k})$. We now show that by pushing the activation of $\iota_{i,k}$ to occur in the interval $[t_s, t_e)$, the relative start time $S_{i,k}$ of job $\iota_{i,k}$ becomes zero.

First observe that there is no other job of τ_i between the idle interval $[t_s, t_e)$ and $\iota_{i,k}$; therefore, after pushing the activations of τ_i till $a_{i,k}$ occurs in $[t_s, t_e)$, job $\iota_{i,k}$ cannot experience blocking by a previous job. Furthermore, since $[t_s, t_e)$ was originally idle, job $\iota_{i,k}$ can start upon activation, leading to $S_{i,k} = 0$.

{*Case* $U^{\mathcal{T}} = 1$ *and the lcm of the periods exists*}. Given an arbitrary schedule for $\mathcal{T}$, we first observe that the schedule will be repeated every hyper-period because the lcm of the period exists. Assume that there are n jobs of τ_i in every hyper-period and no job of τ_i starts upon activation. Furthermore, let $\iota_{i,1}$ and $\iota_{i,n}$ be the first and last job of τ_i respectively in an hyper-period. Hence, for each interval $[a_{i,k\prime}, s_{i,k\prime})$ with $1 \leq k\prime \leq n$, there are only higher priority tasks of τ_i and/or at most one blocking lower priority task executing at that interval. Let $\iota_{i,k}$ be the job of τ_i with the shortest relative start time among all jobs of τ_i in an hyper-period. Since no job of τ_i starts upon activation and $\iota_{i,k}$ is the job with the shortest start time, we can push the activation of all jobs of τ_i to a later moment in time, without changing the schedule, till job $\iota_{i,k}$ starts upon activation. After pushing the activations of τ_i, it holds that $S_{i,k} = 0$ therefore concluding the proof.

□

References