# U-LLREF: An Optimal Scheduling Algorithm for Uniform Multiprocessors

Shelby H. Funk (Speaker) * Archana Meka †

## 1 Introduction

In real-time systems, correctness depends on both the logical result of computation and the time at which results are produced. Real-time systems may be divided into two categories – hard and soft. In soft real-time systems, jobs may miss some deadlines. In hard real-time systems, any missed deadline is considered a system failure. We consider the problem of scheduling a hard real-time system comprised of periodic tasks [6] executing on a uniform multiprocessors, allowing preemption and migration.

A periodic task $T_i = (p_i, e_i)$ generates jobs $T_{i,k}$ at times $k \cdot p_i$ for $k = 0, 1, 2, \ldots$. Each job has an execution requirement of $e_i$ — i.e., it requires $e_i$ time units to complete on a speed-1 processor. Job $T_{i,k}$'s deadline is the arrival time of task $T_i$'s next job, namely $(k+1) \cdot p_i$. On a uniform multiprocessor, each processor has an allocated speed $s$ – if a job runs on a speed-$s$ processor for $t$ time units then $s \times t$ units of work are performed. We denote an $m$-processor uniform multiprocessor $\pi = [s_1, s_2, \ldots, s_m]$, where $s_1 \geq s_2 \geq \ldots s_m$. The total speed of all processors is denoted $S(\pi)$.

One important task parameter is its *utilization*, $u_i = e_i/p_i$, which measures the proportion of time a task must execute on a unit speed processor (i.e., a processor with speed $s = 1$). The total utilization and maximum utilization of task set $\tau$ are denoted $U_{sum}$ and $u_{max}$, respectively. A set of periodic tasks can be scheduled on $m$ identical multiprocessor if $U_{sum} \leq m$ and $u_{max} \leq 1$. For example, the Pfair [2, 1] and LLREF [3] algorithms both can schedule any such task sets. To date, no optimal online scheduling algorithm exists for uniform multiprocessors.

### 1.1 The LLREF Scheduling Algorithm

LLREF, is based on the fluid scheduling model, which executes all tasks at a constant rate. This algorithm divides the schedule into Time and Local execution time planes (TL-planes), which are determined by task deadlines. The algorithm schedules tasks by creating smaller "local" jobs within each TL-plane. The only parameters considered by the algorithm during a TL-plane are the parameters of the local jobs. When a TL-plane completes, the next TL-plane is started. The duration of each TL-plane is the amount of time between consecutive deadlines.

*shelby@cs.uga.edu. Computer Science Department, University of Georgia, Athens, GA 30606 USA.
†archana@uga.edu. Computer Science Department, University of Georgia, Athens, GA 30606 USA.

For example, given the task set comprised of three tasks $(8, 3), (11, 4), (15, 9)$, the intervals of first four TL-planes are $[0, 8], [8, 11], [11, 15]$, and $[15, 16]$.

Each task is assigned an execution requirement for each TL-plane. If a TL-plane starts at time $t_{f_0}$ and ends at time $t_{f_1}$, then the initial value each task's local remaining execution requirement is proportional to its utilization. Specifically, $T_i$'s local execution is initialized to $\ell_{i,0} = u_i(t_{f_1} - t_{f_0})$. If $T_i$ is executing at time $t_x$, the value of $\ell_{i,x}$ decreases. At each scheduling event, LLREF always assigns the $m$ tasks with the highest remaining execution requirement to execute on some processor. These tasks will execute until one of two events occur.

- A bottom (or $B$) event occurs at time $t_x$ when a task $T_i$ has completed its required execution (i.e., when $\ell_{i,x} = 0$).
- A critical (or $C$) event occurs when the task will miss its deadline if it does not execute for the remainder of the TL-plane (i.e., $\ell_{i,x} = t_{f_1} - t_x$). If a task causes a $C$ event, we say the task has zero laxity.

Whenever either of these events occur at a time $t_x$, the $m$ tasks that have the highest remaining execution requirement are assigned to execute. This process is repeated until all tasks have executed for $\ell_{i,0}$ units of time within the TL-plane.

## 2 U-LLREF

We introduce U-LLREF, an extension of LLREF for uniform multiprocessors and prove the algorithm is optimal. Extending LLREF to apply to uniform multiprocessors requires a more complex $C$ event because there are more utilization bounds. On uniform multiprocessors, a periodic task set $\tau = \{T_1, T_2, \ldots, T_n\}$ with $u_1 \geq u_2 \geq \ldots \geq u_n$ is schedulable on $\pi = [s_1, s_2, \ldots, s_m]$ if the following bounds hold [4, 5].

$$\sum_{i=1}^{k} u_i \leq \sum_{i=1}^{k} s_i \text{ for } k = 1, 2, \ldots, m-1, \text{and} \quad (1)$$

$$\sum_{i=1}^{n} u_i \leq \sum_{i=1}^{m} s_i \text{ (i.e., } U_{sum} \leq S(\pi)). \quad (2)$$

By constraint 2, we have $(m-1)$ different types of zero laxity events on uniform multiprocessors instead of the single type of zero laxity event for identical multiprocessors described above. We say a $C_k$ event occurs at time $t_x$ when some set of $k$ tasks have zero laxity on the $k$ fastest processors – i.e., when their total remaining execution is equal to $\sum_{i=1}^{k} s_i(t_{f_1} - t_x)$. Unfortunately, determining which $k$ tasks will create such an event requires exponential running time. The following theorem allows us to use a simpler condition for determining $C$ events.

**Theorem 1** *Let $T_1, \ldots, T_n$ be scheduled on uniform multiprocessor $\pi$ during the interval $[t_{f_0}, t_{f_1}]$. Assume $\ell_{1,x} \geq \ell_{2,x} \geq \ldots \geq \ell_{n,x}$ at time $t_x$ and tasks $T_i$ is scheduled to execute on processor $s_i$ for $i = 1, \ldots, m$. If a $C_k$ event will occur in $\Delta_k$ time units, then there exists some task $T_i$ and some time $t_y$ such that $t_x \leq t_y \leq t_x + \Delta_k$ and $\ell_{i,y} = s_k(t_{f_1} - t_y)$.*

Using this theorem, we can invoke a different type of $C_k$ event – namely, a task $T_i$ has a $C_k$ event when it has zero laxity on processor $s_k$. When such an event occurs, U-LLREF schedules the task on the given processor for the remainder of the TL-plane. If we schedule in this manner, no set of $k$ tasks can have negative laxity on the $k$ fastest processors. Therefore, this method will ensure the first $(m-1)$ bounds given in constraint 2 will always hold.

While the above theorem ensures most of the conditions given above, it does not ensure the total utilization bound is never violated. Below, we see that total local utilization never increases within a TL-plane.

**Theorem 2** *Assume tasks $T_1, \ldots, T_n$ are schedule on uniform multiprocessor $\pi$, where $U_{sum} \leq S(\pi)$. Assume the tasks are scheduled using TL-planes and all $m$ processors execute some task whenever there are at least $m$ tasks with positive local remaining execution requirement. Then the total local utilization will never exceed $U_{sum}$. In fact, if $U_{sum} < S(\pi)$ then the total local utilization decreases over time.*

With these theorems in mind, we define U-LLREF as follows. At the beginning of each TL-plane $[t_{f_0}, t_{f_1}]$, let $\ell_{i,0} = u_i(t_{f_1} - t_{f_0})$ for each $i = 1, 2, \ldots, n$. At each scheduling event, assign the $m$ tasks with largest remaining execution to execute so that tasks with larger remaining execution execute on faster processors. Reschedule whenever some task has a $B$ event ($\ell_{i,x} = 0$) or a $C_k$ event ($\ell_{i,x} = s_k$).

The above theorems demonstrate that scheduling using U-LLREF prevents the utilization constraints 2 and 2 from being violated – i.e., U-LLREF is optimal for scheduling periodic tasks on uniform multiprocessors when preemption and migration are allowed.

## References


[1] Sanjoy K. Baruah. Strong P-fairness: A scheduling strategy for real-time applications. In *Proceedings of the IEEE Workshop on Real-Time Applications*, 1994.

[2] Sanjoy K. Baruah, Neil Cohen, C. Greg Plaxton, and Don Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, June 1996.

[3] Hyeonjoong Cho, Binoy Ravindran, and E. Douglas Jensen. An optimal real-time scheduling algorithm for multiprocessors. In *Proceedings of RTSS 2006: The $27^th$ IEEE Real-Time System Symposium*, Los Alamitos, CA, USA, 2006.

[4] Shelby Funk, Joël Goossens, and Sanjoy K. Baruah. On-line scheduling on uniform multiprocessors. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 183–192. IEEE Computer Society Press, December 2001.

[5] Edward C. Horvath, Shui Lam, and Ravi Sethi. A level algorithm for preemptive scheduling. *Journal of the ACM*, 24(1):32–43, 1977.

[6] Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.