Cent. Eur. J. Comp. Sci. • 1(3) • 2011 • 253-265
DOI: 10.2478/s13537-011-0023-z

VERSITA

Central European Journal of **Computer Science**

Genealogy of Hard Real-Time Preemptive Scheduling Algorithms for Identical Multiprocessors

Research Article

Dirk Müller[1*], Matthias Werner[1‡]

1 *Chemnitz University of Technology,*
Str. der Nationen 62, D-09111 Chemnitz, Germany

Received 2 February 2011; accepted 7 September 2011

Abstract: With the advent of multi-core computer systems, real-time (RT) scheduling on multiprocessors (MPs) is gaining more and more practical relevance. Basic algorithms developed in the 1970s or earlier are strongly influential on state-of-the-art preemptive hard real-time scheduling algorithms. These relationships will be revealed and visualized by four genealogies. Here, the two classification dimensions prioritization dynamics and migration dynamics serve as a framework for the systematization. Finally, such genealogical reconstructions could even lead to the discovery of new algorithms. An extra look will be devoted to the group of fully dynamic scheduling algorithms with full migration in which algorithms can reach full utilization, neglecting scheduling and context switching overheads. The important question for a set of RT scheduling algorithms serving as a basis for more sophisticated ones will be dealt with.

Keywords: multiprocessor • scheduling • migration dynamics • prioritization

1. Introduction

1.1. History

Until recently, computing has been dominated by single-processor architectures. However, as the microprocessor industry has reached the limits on clock frequency imposed by thermal dissipation requirements, it has turned toward multiprocessor computers and multi-core processors. The latter are especially attractive because the tighter coupling of the computing units can reduce internal communication and synchronization delays. While the transition to multi-core computing is already well advanced for desktop and laptop computers, it is just beginning for embedded systems.
Regrettably, this change involves some severe side effects along with solving the problem of thermal dissipation limits. Among them, important ones are the creation of compilers and schedulers capable of taking advantage of the improved hardware and to manage the synchronization between jobs running on different cores.

* *E-mail: dirkm@cs.tu-chemnitz.de*
‡ *E-mail: mwerner@cs.tu-chemnitz.de*

Springer

The most fundamental crux is the impossibility of using several idle cores for running a serially programmed job in parallel.[1]
Some portions of code are always impossible to parallelize, having an immense influence already on the theoretically possible gain, cf. *Amdahl's law* [1]. The global scheduling approach is characterized by the allowance of migration between different processors, see Subsection 4.2. It suffers additionally from anomalies like *Dhall's effect* [31] which shows clearly the potential of huge improvements by applying more sophisticated algorithms.
In the beginning of real-time engineering in the 1970s only two major scheduling strategies, namely *Rate-Monotonic Scheduling* (RMS) and *Earliest Deadline First* (EDF) [52], were considered. RMS applies a static prioritization according to increasing periods. Note that for the important special case of implicit[2] deadlines, this is equivalent to increasing *relative* deadlines. EDF is based on a dynamic prioritization according to increasing *absolute* deadlines making it more powerful but also more complicated. There has been an intensive debate on the advantages of RMS and EDF. A well-written summary on this debate can be found in [20]. In fact, despite of the theoretical superiority of EDF over RMS, RMS is still more prominent due to its implementation simplicity.

1.2. Motivation

Today, there is a multitude of real-time scheduling strategies. This emerged to an elusive set of different approaches where relationships among them frequently are hidden. Hence, it is useful to uncover these dependencies and to rearrange RT scheduling algorithms with respect to them. One possible method is to reconstruct the *genealogy* of real-time scheduling strategies. It will turn out that this method is appropriate since more advanced strategies often can be regarded as crossings of traditional scheduling approaches. Even the discovery of new algorithms might be possible by a thorough evaluation of existing and potential crossings.

1.3. System Model

The principal challenge for MP scheduling is to schedule n given tasks on a minimum number of m processors. Recently, the problem of *sustainability* gained attention. The widespread use of worst-case parameters in system models like the periodic or the more general sporadic task model (where the concept of period is generalized to a minimum release time separation, making the release time pattern non-deterministic) raises the question whether an intuitive improvement on the system parameters always maintains the schedulability property, i.e., the question of sustainability. These intuitive improvements include decreased execution times, larger periods, larger relative deadlines, later arrival times, and smaller release-time jitter. Release-time jitter is the difference between maximum and minimum release time, i.e., the statistical measure *range* of the release time. Unfortunately, it turned out that the "[..] global EDF scheduling policy is not sustainable with respect to later deadlines for the [most general] arbitrary job sets model." [10] The question, whether this poor situation applies for the more specific sporadic task model as well, is an open one, see [10].
Due to the wide range of multiprocessor scheduling algorithms, we have to focus here on a subset. The article only covers *preemptive* algorithms for *hard* real-time scheduling on *identical* multiprocessors. Preemptive means that a job can be suspended when a more important job becomes ready for execution. The opposite of preemptive is non-preemptive where no preemption is allowed at all. Note that between these two extremes, there are as well intermediary solutions like *Fixed Preemption Points* (FPP), see [65]. A survey on several approaches with limited allowed preemption can be found in [64]. Hard real-time means that exceeding a deadline entails severe consequences and can not be tolerated at all. In contrast to that, *soft* RT systems are characterized by a smoother decline of the utility function at and beyond the deadline. Identical multiprocessors are symmetric in terms of computational capacities. The more general model of *uniform* MPs considers processors of different speeds but independent of the task running on them. In the most general

[1] *Providing such a mechanism would be the all-in-one for all parallelization attempts. Certainly, it is impossible to provide such a tool working well in the general case. Otherwise all MP scheduling problems could be reduced to uniprocessor ones.*
[2] *With respect to the relationship between relative deadlines and periods, usually three models of increasing generality are distinguished: implicit deadlines equal to periods, constrained deadlines lower than or equal to deadlines, and arbitrary deadlines with no constraints on them.*

model of *heterogeneous* MPs, the execution speed depends on the task-processor combination.
We assume both zero preemption and migration costs. Although the overheads could be included in worst-case execution times (WCETs), this makes the analysis more pessimistic in terms of remaining application payload. This selection coincides with the one in [28], a well-written, comprehensive survey on MP scheduling lacking the approach of development lines and genealogies taken in this article.
Additionally, an immediate application in embedded real-time systems is possible. Using the insight gained by the systematization, more reasonable decisions on the scheduling algorithms to be used can be taken in early project phases. Second, even *mode changes* in form of an online switching between different approaches are an option. This will become evident by looking at the behavior of the so-called EDZL (*Earliest Deadline until Zero Laxity*) algorithm in urgent situations with zero laxity, see Subsection 4.1.
Due to the complexity of the subject, we consider separate genealogies for the two dimensions *prioritization dynamics* and *migration dynamics*. For the latter one, a further subdivision into EDF- and RMS-based approaches is appropriate.
The rest of the article is structured in the following way. In Section 2, we explain a classification with respect to prioritization dynamics of RT scheduling algorithms. Migration dynamics is a second basis for a classification discussed in Section 3. Three genealogies, one for the first and two for the second dimension are then given in Section 4. This section includes a closer look at fully dynamic approaches with full migration. Finally, a summary and an outlook on future work are given in Section 5.

2. Prioritization Dynamics and Utilization Bounds

In [21], a two-dimensional classification scheme of real-time multiprocessor scheduling algorithms was suggested. The two dimensions extracted are the complexity of the priority scheme and the degree of migration allowed. Let us focus on the first one which distinguishes between *task-level static*, *job-level static*, and *dynamic* priorities. Most typical examples of these approaches are RMS, EDF, and *Least-Laxity First* (LLF), see [55]. LLF chooses priorities according to increasing laxities. Laxity is the difference between time to absolute deadline and remaining execution time. Thus, it is assumed that these three algorithms constitute a basis for more sophisticated ones. In this sequence, they are characterized by increasing performance in terms of schedulability utilization bounds. Unfortunately, implementation complexity and runtime overhead are increasing in parallel. RMS entails the well-known utilization bound[3] of $n(2^{1/n}-1)$ proven in [52] and slightly corrected in [30]. But, EDF and LLF both are capable of *full* processor utilization in the uniprocessor case and, thus, superior in terms of utilization bound on a uniprocessor system.
In the multiprocessor case, the pairing is different. With both RMS and EDF, problems arise. None of them is capable of coping with Dhall's effect, see [31]. The utilization[4] bound even drops to $1/m$, i.e., there is no advantage at all of using m instead of 1 processor when relying on the simple strategies RMS or EDF. The behavior of LLF is slightly better in terms of (average) schedulability as shown in [29].
Thus, it might be worth to generate pairwise crossings of these basic approaches. Later on, we will point out which of them have been found useful.

3. Migration Dynamics and Utilization Bounds

Ascending to the multiprocessor case, things become more complicated. The second categorization dimension, the degree of migration allowed, comes into play, see [21]. The main groups to be distinguished are *partitioned* approaches

[3] *Note that there is a lot of improvement possible by including task-specific data. For an overview on that, see, e.g., [56]. Another way is even to transcend the idea of a utilization bound which can be achieved by using* accelerated simply periodic task sets, *see [56].*
[4] *Note that there are two different common definitions of multiprocessor utilization. The first one keeps with summing up all task utilizations. We prefer here the second one with a subsequent normalization with respect to the number of processors m. It has the advantage of a (potential) independence of m and, thus, a simpler comparability with the uniprocessor case, both ranging in the interval* $[0, 1]$.

with fixed task-processor relationships, *restricted migration* with only jobs, but not tasks, pinned to a processor, and *full migration* allowing additionally intra-job migrations. Again, schedulability utilization bounds are increasing. This becomes evident in the case of dynamic priorities where full migration allows for almost a doubling (from $\frac{m+1}{2m}$ to 1) of the utilization bound, see [21]. Sadly, an increase in implementation complexity and runtime overhead can be observed, too. More details on RM utilization bounds using the partitioned approach can be found in [59] and [53]. It turned out that common algorithms relying an a single approach achieve even lower utilizations bounds. The (still not quite useful) maximum possible utilization bound of 50% could be achieved by an algorithm given in [3] in 2003.
Hence, it is again attractive to analyze pairwise crossings of these approaches in order to alleviate the trade-offs between the fundamental migration policies.
Note the analogy between the prioritization and the migration dynamics classification. Although the common wording does not directly suggest this, the first level is always characterized by invariants referring to *tasks*, the second group by invariants referring to *jobs*, and the last group by the lack of any restrictions. Summing up, there are nine different cases in real-time multiprocessor scheduling according to this classification given in [21] since there are two orthogonal three-valued dimensions: prioritization and migration dynamics.

4. Genealogies

Next, genealogies mapped to these classification dimensions will be elaborated in Subsections 4.1 and 4.2. There is only one genealogy for the prioritization dynamics, but there are two genealogies (for EDF and for RMS) presented for the migration dynamics dimension. Additionally, a closer discussion of fully dynamic approaches with full migration is given in Subsection 4.3. The most valuable improvements are usually obtained by crossings since this allows for combinations of precious properties. But it will be shown that this is not the only way of development. Single-parent descendants occur as well as generalizations and specializations.

4.1. Prioritization Dynamics

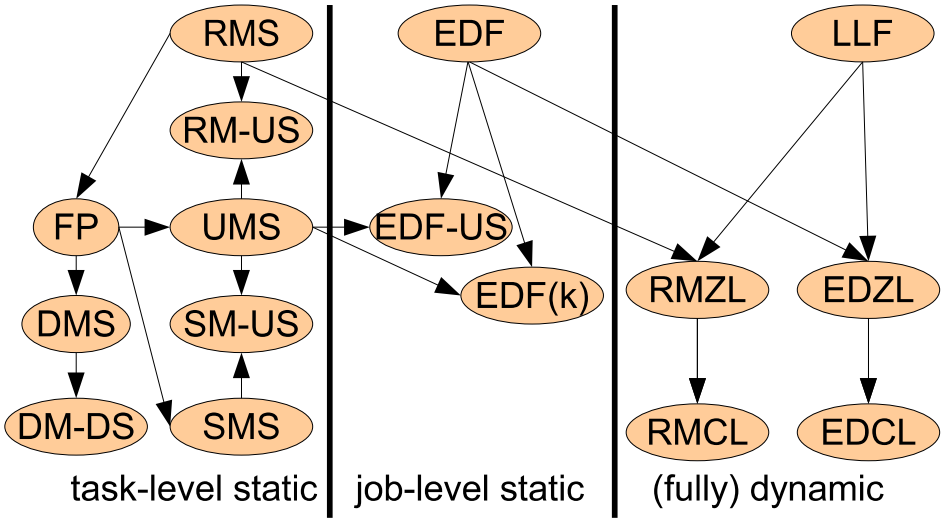

Figure 1. Genealogy of scheduling algorithms based on prioritization dynamics.

For a proposed genealogy based on prioritization dynamics, see Figure 1. RMS is optimal for implicit deadlines fully determined by periods. First, RMS has been generalized to the family of *Fixed Priority Scheduling* (FP Scheduling). Here, the tight coupling of priority to rate (reciprocal of period) is omitted. An excellent discussion of priority orderings can be found in [27]. In the special case of constrained deadlines, it was shown in [51] that *Deadline-Monotonic*

Scheduling (DMS) is optimal on a uniprocessor. Here, priorities are assigned statically according to increasing relative deadlines. Thus, for the implicit deadlines model, RMS and DMS are equivalent. Clearly, DMS is another important specialization of FP. A further specialization of FP scheduling is the *Deadline minus Jitter strategy*, see [66]. It is optimal for tasks with non-zero release jitter and constrained deadlines. The general case of arbitrary deadlines is covered by Audsley's greedy priority assignment algorithm, see [8]. Another conceptually important FP policy is *Utilization-Monotonic Scheduling* (UMS). Here, the higher the utilization the higher the priority[6]. Pure UMS qualifies as a not well-working priority order, see [22]. A further possible ordering is *Slack-Monotonic Scheduling* (SMS), proposed, e.g., in [61]. This algorithm is discussed for the uniprocessor case in [5] and for the multiprocessor case in [4].

An improvement of RMS can be achieved by separating the highest utilization tasks (beyond a threshold value), i.e., providing them with the highest priority, cf. [3]. It is called RM-US meaning *Rate-Monotonic Scheduling with Utilization Separation*. For this family of algorithms, the threshold value coincides with the utilization bound. Note that the resulting ties inside this group of tasks can be broken arbitrarily (but fixed). So, RM-US is a crossing between RMS and UMS. Clearly, the crucial point when applying RM-US is to find a good threshold value. The more obvious choice of $\frac{m}{3m-2}$ suggested in [6], resulting in a limit utilization bound of $1/3 \approx 0.333$ was improved in [54] to the value of the (unique) solution of the transcendent equation $\ln(1+y) = \frac{1-y}{1+y}$ with $y \in \mathbb{R}$. It turns out to be $\ln(1+y) \approx 0.375$.

An analogous combination of SMS and UMS yields *Slack-Monotonic Scheduling with Utilization Separation* (SM-US), see [4], incorporates a special treatment of *heavy* tasks. A good threshold value was found to be $\frac{2}{3+\sqrt{5}} \approx 0.382$ with the same utilization bound value. As a conjecture, B. Andersson gave a threshold and bound value of $\sqrt{2}-1 \approx 0.414$, see [4] which is the maximum possible bound for static-priority scheduling where the priority is only determined by a scale-invariant function of period and execution time, cf. [3]. More advanced crossings with UMS are discussed in [11].

Similar to that, the behavior of DMS can be improved by modifying it with the idea of UMS to DM-DS (*Deadline Monotonic with Density Separation*), cf. [17]. Note that the analogous concept applied here instead of utilization is density. Density is the ratio between execution time and the minimum of relative deadline and period while utilization is defined as the execution-time-to-period ratio. Since DMS is only appropriate for constrained deadlines, density specializes here to an execution-time-to-relative-deadline ratio.

Next, a combination between EDF and UMS, called EDF-US (utilization separation), is possible, see [62]. Here, the basic idea is to provide the highest priority for heavy tasks with a utilization beyond $\frac{m}{2m-1}$. This approach is already close to the optimum threshold of 1/2 which results in the maximum possible utilization bound of $\frac{m+1}{2m}$ yielding 1/2 in the limit $m \to \infty$ according to [9]. This optimal threshold choice is given in the fpEDF (fixed-priority EDF) algorithm [12]. A slight modification with the same parents assigns the highest priority to the $k-1$ heaviest tasks (with highest utilizations). This algorithm was coined EDF(k), see [37]. The difference to EDF-US is the fixed number of these privileged tasks compared to a utilization threshold criterion. There, an optimal value for k is derived. Note further that both approaches can be seen as generalizations of EDF in the forms EDF-US with threshold value 1 and EDF(1).

Furthermore, the strict rate-monotonic approach can be broken in urgent situations. Obviously, a job becomes most urgent when its laxity reaches zero. Then, its priority should be promoted to the highest level. Otherwise this job would not meet its deadline, which has to be avoided by all means[7]. The approach is called *Rate-Monotonic Zero Laxity* (RMZL), presented in [45], in [32] as NPRM (*A New Priority-driven Scheduler Based on RM*) and in [58] as Zero-Slack RM scheduling.

The useful idea of a special treatment of zero laxity situations has been applied to the EDF algorithm, too. The result is the *Earliest Deadline until Zero Laxity* (EDZL) algorithm, see, e.g., [11].

Implementation problems with RMZL and EDZL led to the advanced versions RMCL, see [63], and EDCL, see [42]. Here, the letter 'C' means 'critical'. Priority promotion takes place earlier which relieves the operating system from checking for zero laxity steadily. Especially the number of scheduler invocations can be reduced by applying RMCL instead of RMZL or EDCL instead of EDZL. In [26], the RMZL algorithm has been extended to cover the more general case of constrained deadlines. At the same time, [26] give a good survey on the family of fixed-priority zero-laxity (FPZL) scheduling algorithms.

[6] *Note that* higher *priorities are usually characterized by* lower *priority level values.*

[7] *This corresponds to* hard *real-time systems. The situation is different in the case of soft RT systems. Here, a less greedy strategy with compromises minimizing some mathematical lateness measure can be the better option.*

The *mode change* from task-level static (RMZL, RMCL) or job-level static (EDZL, EDCL) to fully dynamic in the urgent situation of zero laxity gives an idea for an implementation. The kernel only needs to support the basic algorithms RMS, EDF and LLF. A guard running in parallel can then detect an urgency situation and force the kernel to switch to the fully dynamic approach. Having completed the zero laxity job at its deadline, a switch back to the initial scheduler has to be triggered. This idea can be, e.g., realized on the *MAintainable Real-time System (MARS)*, cf. [48][25]. For a discussion on mode switches in MARS, see, e.g., [19], p. 330.
The leaves EDF-US, RMCL and EDCL in the genealogy all are advanced algorithms originating from crossings over the basic algorithms RMS, EDF and LLF. Among them, RMCL seems to be most promising for real-time systems since it combines predictability and ease of implementation, both inherited from classical RMS, with an increased schedulability compared to this prominent and successful algorithm.

4.2. Migration Dynamics

With respect to migration dynamics, see Figure 2, partitioned scheduling approaches are the simplest ones. A distribution of the given tasks to the processors is selected *statically*. On the other hand, finding an optimal matching between processors and tasks is a problem closely related to Bin Packing, and thus NP-Hard in the strong sense.
Then, all processors run completely independently under the simplifying assumption of independent tasks. Uniprocessor scheduling methods are applied individually. The more advanced technique of global scheduling allows for migrations. In its stricter variant, migration is allowed only between jobs.

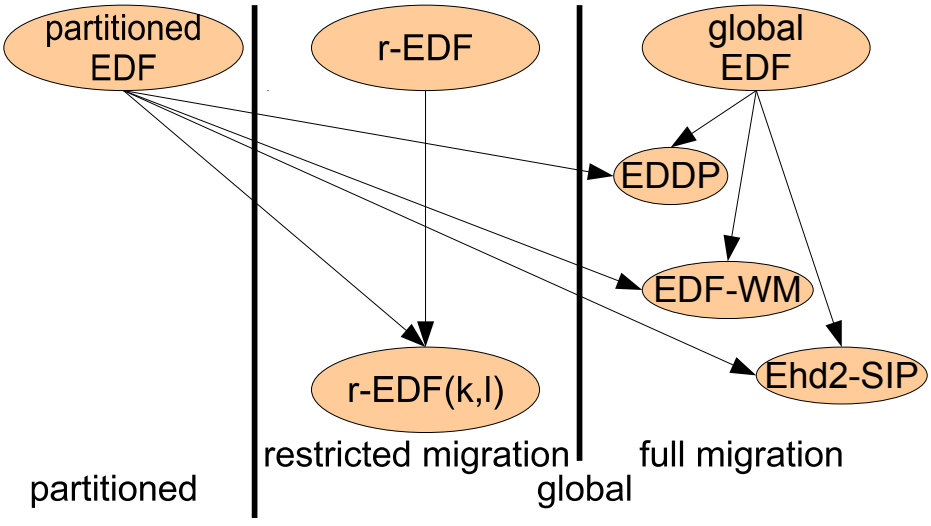

Figure 2. Genealogy of scheduling algorithms based on migration dynamics for EDF-based algorithms.

The most liberal policy allows for migrations at *any* time. One possible strategy is global EDF. A restriction of global EDF eliminating intra-job migrations is r-EDF, see [13] and [34], where 'r' means restricted migration.
A combination of the partitioned approach and full migration has led to a new strategy called *semi-partitioned* scheduling. As long as possible, the task set is partitioned onto processors. Then, all remaining tasks are split and the parts are distributed onto processors with sufficient rest capacities. Thus, the migration behavior is spread compared to global scheduling. For pinned tasks, no migration at all is allowed while for the split tasks, migration occurs in a regular pattern and becomes the rule. Semi-partitioned scheduling has its origin in an algorithm suggested for *soft* real-time, *EDF-fm* (fixed or migrating) [2]. With EDF as a basis, this specializes to *EDF with Window-constraint Migration* (EDF-WM) described in [47], *Earliest Deadline Deferrable Portion* (EDDP) given in [43] or to *Earliest Deadline First with the highest-priority deferrable portion-2 task - Sequential assignment in Increasing Period* (Ehd2-SIP), cf. [41].

While EDF-fm as a soft RT algorithm[9] provides no (useful) utilization bound, EDDP, cf. [43], reaches $4\sqrt{2}-5 \approx 0.657$, Ehd2-SIP, see [41], has a utilization bound of 1/2. These three approaches are all based on implicit deadlines and, thus, comparable. Opposed to that, EDF-WM works with the most general case of arbitrary deadlines.
Furthermore, a crossing of r-EDF with partitioned scheduling results in r-EDF(k,l), cf. [34], meaning a separation of k heavy tasks onto l processors. In that paper, it turned out that enabling for migrations on the entire set of processors is not useful. Instead, a restriction to smaller subsets, following the semi-partitioned approach is recommended. Note that referring to Figure 2, the discussion is restricted to EDF-based approaches. First practical results in [15] taking overheads into account show that partitioned EDF is superior to global EDF.

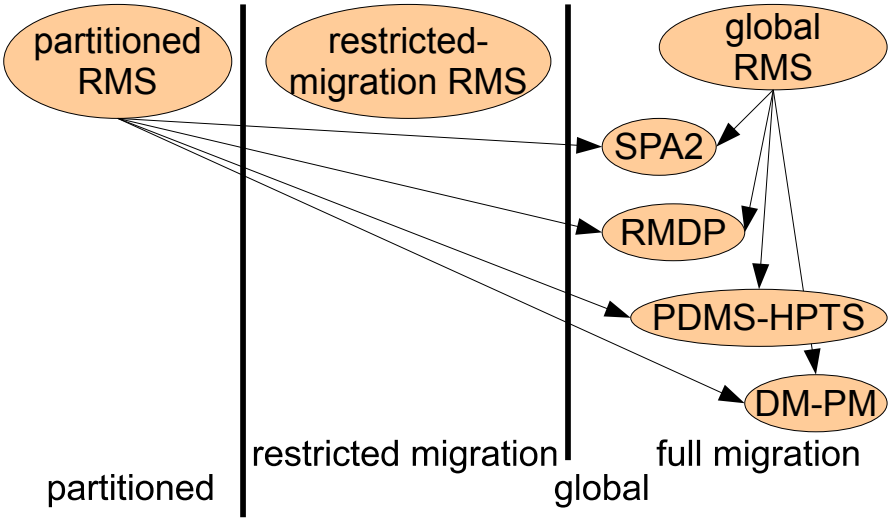

Figure 3. Genealogy of scheduling algorithms based on migration dynamics for RMS-based algorithms.

For RMS, see Figure 3, hitherto only the basic variants of partitioned and global RMS have played an important role. The restricted-migration variant of RMS is shortly discussed in [21]. Recently, combinations of partitioned and global RMS, coined *Rate Monotonic Deferrable Portion* (RMDP), see [44] and *Deadline Monotonic with Priority Migration* (DM-PM), cf. [46], were suggested. Note that DM-PM equals RM-PM when considering implicit deadlines where deadlines equal periods. Another algorithm belonging to this group is *Partitioned Deadline-Monotonic scheduling with Highest Priority Task on each processing core is allowed to be Split* (PDMS-HPTS) given in [50].
The utilization bounds of these fixed-priority semi-partitioned algorithms are 1/2 both for RMDP and DM-PM, which is the same as for partitioned algorithms. A first improvement to about 0.6547 was delivered with PDMS-HPTS, see [50]. PDMS-HPTS uses a threshold value of 0.25 to distinguish between heavy and light tasks, cf. [50]. It assigns the highest priority to split tasks and, thus, exploits the fact that highest priority tasks are executed completely undisturbed. This is similar to the approach taken with UMS-parent hybrid prioritization dynamics algorithms, cf. Subsection 4.1. The algorithm EKG-2, see [7], a variant of EKG (*EDF with task splitting and k processors in a group*), see Subsection 4.3 is a semi-partitioned algorithm with a utilization bound of $2/3 \approx 0.667$ and a maximum of 4 preemptions per job on the hyperperiod average, cf. [7]. The practical relevance of semi-partitioned algorithms was recently confirmed in [16]. There, real-world overheads are taken into account.
A breakthrough by Guan *et al.* was worked out in 2010. The famous Liu/Layland (limit) bound for uniprocessors ($\ln 2 \approx 0.693$) could be transferred to multiprocessors with their algorithm *Semi-partitioned algorithm 2* (SPA2) given in [38]. This seminal algorithm differs from the other ones in the allocation policy to the processor. While First Fit was

[9] *Due to its importance as first published algorithm in this group, we exceptionally relax the restriction only to consider hard real-time algorithms.*

the standard before, Guan *et al.* rely on the Worst Fit strategy. With it, the processors are filled in turn, improving on the chances for the split tasks to find an appropriate processor. Note that the core idea (and as well another application field) of the Worst Fit strategy is load balancing. Another clever move in SPA2 is the discrimination into heavy and light tasks known from the hybrid prioritization dynamics algorithms with UMS parent, see Subsection 4.1. The specialty here is that the threshold value is a function of n and not of m, see [38].

For the special case of *simply periodic* task sets explained, e.g., in [56], an optimal algorithm with a utilization bound of 1 is presented in [40]. This bound decreases when not all tasks can be split arbitrarily. Their approach is general enough to provide a solution even for uniform MPs.

Recently, this concept of semi-partitioned scheduling has been generalized to *cluster-based multiprocessor scheduling*, see [33]. Semi-partitioned scheduling partitions *tasks* into pinned and migrating ones, cf. [46]. Thus, some tasks are split into subtasks, making the migration more regular. In contrast to that, cluster-based scheduling tackles the problem from the resources side, the *processors*/*cores*. First, physical clusters with one-to-one mappings between their processors and the processors of the platform can be created. Inside each cluster, global scheduling is applied, whereas the cluster formation itself resembles the partitioned approach. The even more general concept of virtual clusters allows for cluster boundaries across processors according to [33]. This means that virtual clusters can share some processors of the platform. The well-known cases of partitioned and global scheduling are then only the two extreme cases of cluster-based scheduling with m clusters of size 1 and with just 1 cluster of size m, cf. [33]. A summary in form of an inheritance diagram is given in Figure 4. Note that the arrows in the diagram can be interpreted as development lines, too, since the direction of research followed the generalization approach in this particular case. It has to be stressed that an inheritance diagram differs from a genealogy in the general case, which becomes apparent by noting the time-independent and the historical nature of them, respectively.

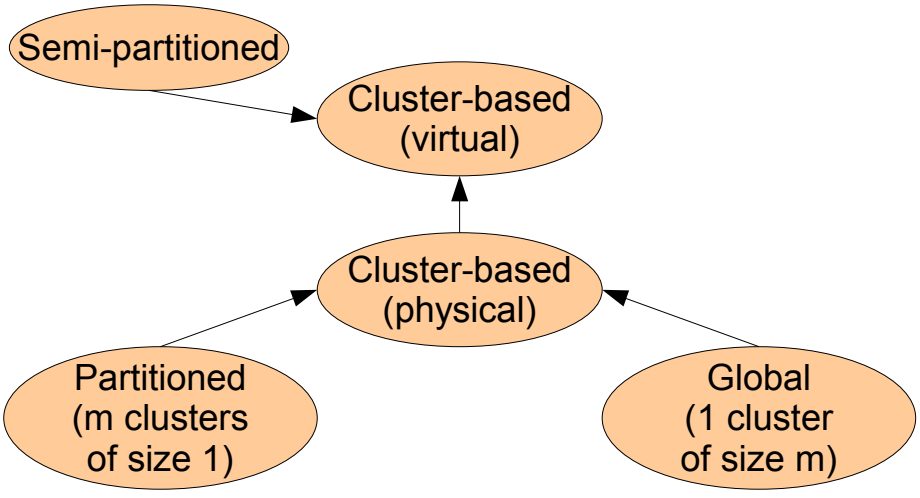

Figure 4. Cluster-based scheduling as a very general concept on migration dynamics: inheritance diagram.

Having a closer look at Figures 1 and 2, it can be observed that a crossing of two policies always belongs to the more liberate class. After a short reflection, this is not a surprising fact since the categorization is based on universal statements which are falsified by counter examples. Thus, a blending of two differently strong requirements results in a hybrid approach meeting only the weaker ones.

Among the leaves in Figure 2, EDF-WM has the highest relevance. It considerably reduces migrations while keeping a high schedulability (in terms of utilization). It strictly dominates classical partitioned approaches since it falls back to migrating tasks only if necessary, cf. [47]. The last point makes EDF-WM superior to EDDP and Ehd2-SIP. It seems to be one of the best known solutions where task splittings are semantically possible.

Evaluating the leaves in Figure 3, DM-PM is better than RMDP since it creates less preemptions and migrations. Algorithm PDMS-HPTS is even better due to its higher utilization bound, cf. [46]. But the best known fixed-priority

semi-partitioned algorithm is SPA2 since it allows for a transfer of the famous Liu/Layland (limit) bound of $\ln 2 \approx 0.693$ from uni- to multiprocessors, see [38].

4.3. Fully Dynamic Scheduling with Full Migration

Finally, the case of combining full dynamics with full migration is examined in more detail. According to the state of the art, *fairness* and the *fluid* model are the two basic requirements and, thus, cornerstones of all algorithms in this group. Note that, recently, this general agreement was broken by the presentation of a non-fair optimal MP scheduling algorithm in [57]. Important algorithms related to these cornerstones are Pfair (*Proportional fair*), see [14], EKG (*EDF with task splitting and k processors in a group*), cf. [7], LLREF (*Largest local remaining execution time first*), see [23], LRE-TL (*Local remaining execution-TL*[11]), cf. [36] and GPS (*Generalized processor sharing*), see [60], which has its origin in computer networks and combines the above mentioned principles of fairness and fluid traffic. For a further explanation on GPS, see, e.g., [24].
A genealogy of these approaches can be found in Figure 5. Note that EDF itself does not belong to this group but has influenced both EKG and PD^2 (*Pfair scheduling with deadlines and 2 tie-breaking rules*). In the diagram, the principle of fairness and the fluid scheduling approach, see [39], are included, too. Additionally, Round Robin (RR) serves as a pattern for Pfair scheduling which can be seen as a *weighted* version of RR. The grouping in EKG is closely related to cluster-based scheduling with physical clusters. Note that only EKG-M where a processor covers *all* processors belongs to this schema in a strict sense. EKG variants with a smaller k value belong to the group of semi-partitioned scheduling algorithms, cf. Subsection 4.2. LRE-TL is an improvement of LLREF as it eliminates unnecessary task switches and migrations, cf. [36]. A further algorithm is *Notional Processor Scheduling Fractional capacity* (NPS-F) [18]. The trade-off between maximizing the utilization bound and minimizing the number of preemptions can be controlled by a parameter, similar to EKG. Thus, it can be regarded as a successor of EKG.

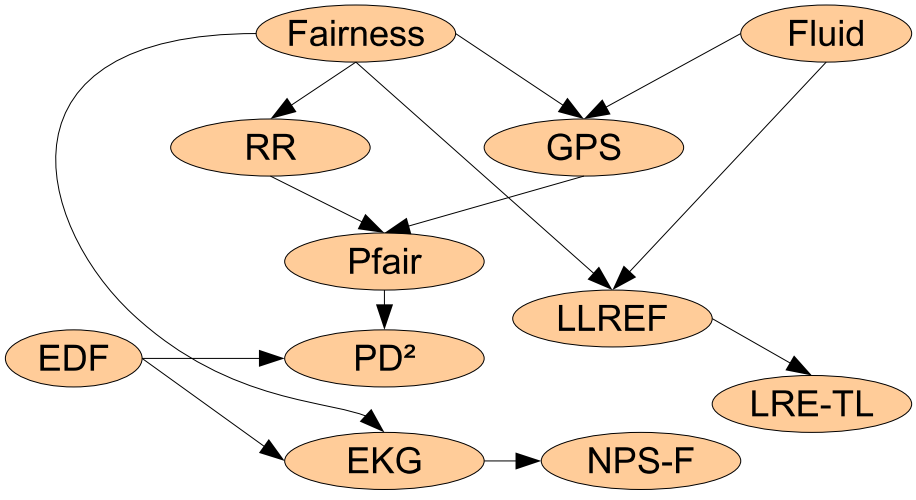

Figure 5. Genealogy of fully dynamic scheduling algorithms with full migration.

Among the four leaves in Figure 5, algorithms EKG, NPS-F and LRE-TL are considered the best ones. They allow for a substantial reduction of context switches compared to the fourth leaf PD^2. For LLREF, an adaptation with a switch from continuous to *discrete* time is necessary in practice. For an application of LLREF in automotive embedded systems, see,

[11] *TL stands for the two axes in the TL-plane: time (abscissa) and local execution time (ordinate).*

e.g., [49]. Recently, a new concept of generalization of fully dynamic algorithms with full migration was presented by Funk *et al.* in [35]. Despite these advances in concepts and generalizations, due to complexity and managing overheads, the practical benefit of such algorithms remains restricted in these days.

5. Summary

A careful analysis of some state-of-the-art real-time scheduling algorithms for multiprocessors resulted in a matching to the classification scheme given in [21]. Next, two genealogies with respect to the two dimensions prioritization dynamics and migration dynamics have been extracted. Additionally, a third genealogy for the case of fully dynamic algorithms with full migration has been elaborated.

With respect to the migration dynamics dimension, the concept of cluster-based scheduling turned out to be very general, integrating the well-known approaches of global and partitioned scheduling as special cases.

The conjecture that the small set of RMS, EDF and LLF is sufficient for the construction of a wide variety of real-time scheduling algorithms has been confirmed. This stresses their salient position and justifies their gained currency in university RT scheduling courses.

The best algorithms investigated in terms of a good compromise between high utilization bounds and a low number of migrations and context switches turned out to be RMCL, EDF-WM, SPA2, PDMS-HPTS and LRE-TL. Implementations of them are promising. Some crossings performed led to new algorithms which already incorporate the mode changes mentioned in the introduction, Section 1. A prominent example is EDZL which switches in zero laxity situations from the EDF to the LLF paradigm. Such switches are included in the related algorithms RMZL, EDCL and RMCL as well.

A subdivision of the given task set into heavy and light tasks in order to fight Dhall's effect turned out to be extremely useful. This basic approach has been applied in the family of US-algorithms, but also in EDDP, Ehd2-SIP, PDMS-HPTS and SPA2.

For the future, trends should be detected and applied to advanced scheduling algorithms to be developed. Such an analysis is alleviated by the genealogy approach presented here.

A further step of the analysis is to examine the intersection of the two dimensions leading to a nine square field. This direction has already been undertaken in [21]. The special case of semi-partitioned deadline-monotonic scheduling was analyzed in [46].

Since real-time multiprocessor scheduling is a very large subject, the authors had to restrict their study to a subset. The field of hard real-time preemptive scheduling on identical multiprocessors was chosen. Further systematization and genealogy research should cover as well soft real-time and non-preemptive scheduling algorithms also on uniform and heterogeneous multiprocessors. A further step is to consider the degree of allowed preemption as a third dimension. At least, there are the variants preemptive, non-preemptive and FPP scheduling, cf. [64]. This would result in $3 \times 3 \times 3 = 27$ cases out of them only 9 have been discussed here.

Acknowledgments

The authors would like to thank Ted Baker for useful comments on a preliminary version of this article.

References

[1] Amdahl G.M., Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS '67 (Spring), Proceedings of the April 18-20, 1967, spring joint computer conference, pp. 483–485. ACM, New York, NY, USA, 1967

[2] Anderson J.H., Bud V., Devi U.C., An EDF-based scheduling algorithm for multiprocessor soft real-time systems. In: ECRTS '05, Proceedings of the 17th Euromicro Conference on Real-Time Systems, pp. 199–208. IEEE Computer Society, Washington, DC, USA, 2005

[3] Andersson B., Static-priority scheduling on multiprocessors. Ph.D. thesis, Chalmers University of Technology, 2003

[4] Andersson B., Global static-priority preemptive multiprocessor scheduling with utilization bound 38%. In: OPODIS '08, Proceedings of the 12th International Conference on Principles of Distributed Systems, 73-88. Springer-Verlag, Berlin, Heidelberg, 2008

[5] Andersson B., The utilization bound of uniprocessor preemptive slack-monotonic scheduling is 50%. In: SAC '08, Proceedings of the 2008 ACM symposium on Applied computing, 281-283, ACM, New York, NY, USA, 2008

[6] Andersson B., Baruah S., Jonsson J., Static-priority scheduling on multiprocessors. In: RTSS '01, Proceedings of the 22nd IEEE Real-Time Systems Symposium, 93, IEEE Computer Society, Washington, DC, USA, 2001

[7] Andersson B., Tovar E., Multiprocessor scheduling with few preemptions. In: RTCSA '06, Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 322-334, IEEE Computer Society, Sydney, Australia, 2006

[8] Audsley N., Burns A., Richardson M., Tindell K., Wellings A., Applying new scheduling theory to static priority pre-emptive scheduling. Software Engineering Journal 8(5), 284–292 (1993)

[9] Baker T.P., An analysis of EDF schedulability on a multiprocessor. IEEE Transactions on Parallel and Distributed Systems, 2005, 16, 760-768

[10] Baker T.P., Baruah S.K., Sustainable multiprocessor scheduling of sporadic task systems. In: ECRTS, pp. 141–150. IEEE Computer Society, 2009

[11] Baker T.P., Cirinei M., Bertogna M., EDZL scheduling analysis. Real-Time Syst., 2008, 40(3), 264-289

[12] Baruah S., Optimal utilization bounds for the fixed-priority scheduling of periodic task systems on identical multiprocessors. Computers, IEEE Transactions on, 2004, 53(6), 781-784

[13] Baruah S.K., Carpenter J., Multiprocessor fixed-priority scheduling with restricted interprocessor migrations. In: ECRTS, 195-202, IEEE Computer Society, 2003

[14] Baruah S.K., Cohen N.K., Plaxton C.G., Varvel D.A., Proportionate progress, A notion of fairness in resource allocation. Algorithmica, 1996, 15(6), 600-625

[15] Bastoni A., Brandenburg B.B., Anderson J.H., An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers. In: RTSS, 14-24, IEEE Computer Society, 2010

[16] Bastoni A., Brandenburg B.B., Anderson J.H., Is semi-partitioned scheduling practical? Real-Time Systems, Euromicro Conference, 125-135, 2011

[17] Bertogna M., Cirinei M., Lipari G., New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors. In: J.H. Anderson, G. Prencipe, R. Wattenhofer (Eds.) OPODIS, Lecture Notes in Computer Science, Vol. 3974, 306-321. Springer, 2005

[18] Bletsas K., Andersson B., Preemption-light multiprocessor scheduling of sporadic tasks with high utilisation bound. In: Proceedings of the 2009 30th IEEE Real-Time Systems Symposium, RTSS '09, 447-456, IEEE Computer Society, Washington, DC, USA, 2009

[19] Buttazzo G.C., Hard real-time computing systems, predictable scheduling algorithms and applications. Kluwer Academic Publishers, 1997

[20] Buttazzo G.C., Rate monotonic vs. EDF, judgment day. Real-Time Syst., 2005, 29, 5-26, http://dl.acm.org/citation.cfm?id=1035387.1035388

[21] Carpenter J., Funk S., Holman P., Anderson J., Baruah S., A categorization of real-time multiprocessor scheduling problems and algorithms. In: Handbook on Scheduling Algorithms, Methods, and Models, pp. 30–1–30–19. Chapman Hall/CRC, Boca, 2004

[22] Chang Y., Davis R., Wellings A., Schedulability analysis for a real-time multiprocessor system based on service contracts and resource partitioning. Tech. rep., University of York, 2008, http://www.cs.york.ac.uk/ftpdir/reports/2008/YCS/432/YCS-2008-432.pdf

[23] Cho H., Ravindran B., Jensen E.D., An optimal real-time scheduling algorithm for multiprocessors. Real-Time Systems Symposium, IEEE International , 101-110, 2006

[24] Cirinei M., Exploiting the power of multiprocessors for real-time systems. Ph.D. thesis, Scuola Superiore S Anna, 2007

[25] Damm A., Reisinger J., Schwabl W., Kopetz H., The real-time operating system of MARS. Operating Systems Review, 1989, 23(3), 141-157

[26] Davis R., Burns A., FPZL schedulability analysis. techreport YCS-2010-452, University of York, Department of Computer Science, 2010

[27] Davis R.I., Burns A., Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time

systems. In: T.P. Baker (Ed.) IEEE Real-Time Systems Symposium, 398-409, IEEE Computer Society, Washington, DC, USA, 2009

[28] Davis R.I., Burns A., A survey of hard real-time scheduling for multiprocessor systems. ACM Computing Surveys, 2011, 43(4),

[29] Dertouzos M.L., Mok A.K., Multiprocessor online scheduling of hard-real-time tasks. IEEE Trans. Softw. Eng., 1989, 15(12), 1497-1506

[30] Devillers R., Goossens J., Liu and Layland's schedulability test revisited. Information Processing Letters, 2000, 73(5-6), 157-161

[31] Dhall S.K., Liu C., On a real-time scheduling problem. Operations Research, 1978, 26(1), 127-140

[32] Dong J., Zhang Y., A modified rate-monotonic algorithm for scheduling periodic tasks with different importance in embedded system. In: ICEMI 2009 - Proc. of 9th International Conference on Electronic Measurement and Instruments, pp. 4606–4609 (2009)

[33] Easwaran A., Shin I., Lee I., Optimal virtual cluster-based multiprocessor scheduling. Real-Time Syst., 2009, 43(1), 25-59

[34] Funk S., Baruah S., Restricting EDF migration on uniform multiprocessors. In: Proceedings of the 12th International Conference on Real-Time Systems, 2004

[35] Funk S., Levin G., Sadowski C., Pye I., Brandt S., DP-Fair, a unifying theory for optimal hard real-time multiprocessor scheduling. Real-Time Systems, 2011, 47(5), 389-429

[36] Funk S.H., Nadadur V., LRE-TL, An optimal multiprocessor scheduling algorithm for sporadic task sets. In: Proceedings of the 17th Real-Time and Network Systems (RTNS), 2009, 159-168

[37] Goossens J., Funk S., Baruah S., Priority-driven scheduling of periodic task systems on multiprocessors. Real-Time Syst., 2003, 25, 187-205

[38] Guan N., Stigge M., Yi W., Yu G., Fixed-priority multiprocessor scheduling with liu and layland's utilization bound. In: RTAS '10, Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium, 165-174, IEEE Computer Society, Washington, DC, USA, 2010

[39] Holman P., Anderson J.H., Adapting pfair scheduling for symmetric multiprocessors. J. Embedded Comput., 2005, 1(4), 543-564

[40] Jung M.J., Seong Y.R., Lee C.H., Optimal rm scheduling for simply periodic tasks on uniform multiprocessors. In: ICHIT '09, Proceedings of the 2009 International Conference on Hybrid Information Technology, 383-389, ACM, New York, NY, USA, 2009

[41] Kato S., Yamasaki N., Real-time scheduling with task splitting on multiprocessors. In: RTCSA '07, Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 441-450, IEEE Computer Society, Washington, DC, USA,2007

[42] Kato S., Yamasaki N., Global EDF-based scheduling with efficient priority promotion. In: RTCSA '08, Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 197-206, IEEE Computer Society, Kaohsiung, Taiwan, 2008

[43] Kato S., Yamasaki N., Portioned EDF-based scheduling on multiprocessors. In: EMSOFT '08, Proceedings of the 8th ACM international conference on Embedded software, 139-148, ACM, New York, NY, USA, 2008

[44] Kato S., Yamasaki N., Portioned static-priority scheduling on multiprocessors. In: IPDPS, 1-12, IEEE, 2008

[45] Kato S., Yamasaki N., Real-time scheduling module for linux kernel. In: IPSJ Transactions on Advanced Computing Systems, 2009, Vol. 2, 75-86

[46] Kato S., Yamasaki N., Semi-partitioned fixed-priority scheduling on multiprocessors. In: RTAS '09, Proceedings of the 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium, 23-32, IEEE Computer Society, Washington, DC, USA, 2009

[47] Kato S., Yamasaki N., Ishikawa Y., Semi-partitioned scheduling of sporadic task systems on multiprocessors. In: ECRTS '09, Proceedings of the 2009 21st Euromicro Conference on Real-Time Systems, 249-258, IEEE Computer Society, Dublin, Ireland, 2009

[48] Kopetz H., Damm A., Koza C., Mulazzani M., Schwabl W., Senft C., Zainlinger R.: Distributed fault-tolerant real-time systems, The Mars Approach. IEEE Micro, 1989, 9(1), 25-40

[49] Krämer S., Mottok J., Meier H., Anpassung des LLREF-Algorithmus: Multi-Core-Scheduling in Embedded Systemen (in German). Hanser automotive, 2010, 1/2; 3/4; 5/6, 18-22; 23-25; 14-16, http://www.hanser-automotive.de/fileadmin/heftarchiv/2004/31567.pdf

[50] Lakshmanan K., Rajkumar R., Lehoczky J., Partitioned fixed-priority preemptive scheduling for multi-core processors. In: ECRTS '09, Proceedings of the 2009 21st Euromicro Conference on Real-Time Systems, 239-248, IEEE Computer Society, Washington, DC, USA, 2009

[51] Leung J.Y.T., Whitehead J., On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation, 1982, 2(4), 237-250

[52] Liu C.L., Layland J.W., Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM, 1973, 20(1), 46-61

[53] López J.M., García M., Díaz J.L., García D.F., Utilization bounds for multiprocessor rate-monotonic scheduling. Real-Time Syst., 2003, 24, 5-28, http://portal.acm.org/citation.cfm?id=608141.608166

[54] Lundberg L., Analyzing fixed-priority global multiprocessor scheduling. In: RTAS '02, Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), p. 145. IEEE Computer Society, Washington, DC, USA, 2002

[55] Mok A.K., Fundamental design problems of distributed systems for the hard-real-time environment. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA, 1983

[56] Müller D., Accelerated simply periodic task sets for RM scheduling. In: Proc. of Embedded Real Time Software and Systems (ERTS^2), 46, Toulouse, 2010, http://www.erts2010.org/Site/0ANDGY78/Fichier/PAPIERS%20ERTS%202010%202/ERTS2010_0140_final.pdf

[57] Nelissen G., Berten V., Goossens J.G., Milojevic D., An optimal multiprocessor scheduling algorithm without fairness. In: In Proceedings of the 31th IEEE Real-Time Systems Symposium (Work in Progress session - RTSS10-WiP), 2010

[58] Niz D.d., Lakshmanan K., Rajkumar R., On the scheduling of mixed-criticality real-time task sets. In: RTSS '09, Proceedings of the 2009 30th IEEE Real-Time Systems Symposium, 291-300, IEEE Computer Society, Washington, DC, USA, 2009

[59] Oh D.I., Baker T.P., Utilization bounds for n-processor rate monotone scheduling with static processor assignment. Real-Time Systems, 1998, 15(2), 183-192

[60] Parekh A.K., Gallager R.G., A generalized processor sharing approach to flow control in integrated services networks, the single-node case. IEEE/ACM Trans. Netw., 1993, 1(3), 344-357

[61] Ramamritham K., Stankovic J.A., Shiah P.F., Efficient scheduling algorithms for real-time multiprocessor systems. IEEE Trans. Parallel Distrib. Syst., 1990, 1(2), 184-194

[62] Srinivasan A., Baruah S., Deadline-based scheduling of periodic task systems on multiprocessors. Inf. Process. Lett., 2002, 84(2), 93-98

[63] Takeda A., Kato S., Yamasaki N., Real-time scheduling based on rate monotonic for multiprocessors (in japanese). IPSJ Transactions on Advanced Computing Systems, 2009, 2(1), 64-74

[64] Yao G., Buttazzo G., Bertogna M., Comparative evaluation of limited preemptive methods. In: Proc. of the 15th IEEE Int'l Conf. on Emerging Technologies and Factory Automation (ETFA 2010). Bilbao, Spain, 2010, http://retis.sssup.it/~marko/papers/ETFA10.pdf

[65] Yao G., Buttazzo G., Bertogna M., Feasibility analysis under fixed priority scheduling with fixed preemption points. In: Proc. of the 16th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA 2010). Macau, China, 2010, http://retis.sssup.it/~marko/papers/RTCSA10.pdf

[66] Zuhily A., Burns A., Optimal (D-J)-monotonic priority assignment. Inf. Process. Lett., 2007, 103(6), 247-250