How Different are Offline and Online Scheduling?

Gerhard Fohler, University of Kaiserslautern, Germany

The heated debate on whether offline or online scheduling (e.g., [2], [1], and [3]) is preferable for real-time systems has taken attention away from the question how the two are actually defined and what their essential differences are. Results and arguments in the debate are typically based on a conglomerate of assumptions, system design issues, and particular views of scheduling rather than the actual cores of the ways to schedule themselves. We do not wish to get involved into the "TT' vs. ET" debate, rather have a look the related scheduling paradigms to determine what their differences are. We conclude they are much smaller than perpertuated clichees.

One of the central choicesfor scheduler design is that of the *activation paradigm*, i.e., when are events recognized, who initiates activities, when are these decisions taken? In conventional systems, the *event triggered* approach is prevalent, in which occurrences of events initiate activities in the system immediately. In *time triggered* systems, activities are iniated at predefined points in time [2].

Offline Scheduling We start with the time triggered approach, contrasting some properties already here to those of event triggered. Initiating activities in the system with the progression of time requires thorough, complete understanding of the system and the environment it will operate in. Scheduling for TT is usually carried out via a scheduling table, which lists tasks and their activation times. An offline algorithm takes complete information about the system activities, which reflect the knowledge about anticipated environmental situations and requirements, and creates a single table, representing a feasible solution to the given requirements. As the algorithm is performed offline, fairly complex task sets can be handled, e.g., precedence constraints, distribution and communication over networks, task allocation, mutual exclusion, separation of tasks, etc. Should a feasible solution not be found, retries are possible, e.g., by changing the parameterization of the algorithm or the properties of the task set. At runtime, a very simple runtime dispatcher executes the decisions represented in the table, i.e., which (portion of a) task to execute next. Typically, a minimum granularity of time is assumed for the invocations of the runtime scheduler, so called *slots* .

Online Scheduling In event triggered systems, events invoke an online scheduler, which takes a decision based on a set of pre defined rules, e.g., represented as priorities. An offline schedulability test can be used to show that, if a set of rules is applied to a given task set at runtime, all tasks will meet their deadlines. Major representative lines of such algorithms are based on fixed priorities, e.g, rate monotonic or dynamic priorities.

Fundamentally Different? Given the different assumptions, approaches, and properties, offline and online scheduling could be perceived as very different, or even opposing paradigms. We will take a closer look.

The ET *real-time scheduling process* takes sets of tasks with timing constraints and performs a test if these constraints can be met if a given algorithm is used at

runtime. The algorithm may take properties of tasks, notably priority or deadline, as input, or determine them as directives, artifacts for the online scheduling algorithm Then, the task properties, "priority" are separated from the importance of a task, "deadline" from the timing constraint. Rather, they both serve only to direct the online scheduling algorithm to execute the proper rules for schedulability.

The scheduling table of offline scheduling provides a "proof by construction" that all timing constraints will be met. In contrast to a proof that no timing constraints could be violated in any situation, an offline approach only needs to show that one situation exists, i.e., the scheduling table, in which the timing constraints are met: instead of a "for all" proof, considering even situations which may never occur during the runtime of a system, a "there exists one" suffices.

Thus, the process follows the steps: task set with timing constraints – schedulability test and determination of rules (e.g., via directives priority or deadline) – execution of rules by runtime scheduler – timing constraints met.

Looking closer, we can see that TT real-time scheduling work in the same way. Instead of an definition of rules, e.g., "earliest deadline first", the decisions on which task to execute are represented in the scheduling table, "schedule next task as given table".

While TT scheduling has to assume a *periodic world* and ET provides *flexibility* for tasks with not fully known parameters, e.g., aperiodic, the difference concerns mostly runtime execution without guarantees. When offline guarantees are required, task parameters have to be known offline: without worst case execution, period or maximum arrival frequency, offline guarantees cannot be given, independent of the scheduling paradigm used.

Hence, we can conclude that the terms "offline" and "online" scheduling cannot be seen as disjoint in general. Real-time scheduling requires offline guarantees, which require assumptions about online behavior at design time. At runtime, both offline and online execute according to some (explicitly or implicitly) defined rules, which guarantee feasibility. The question "offline" vs. "online" is thus less black and white, but more about how much of the decision process is Thus, both offline and online are based on a substantial offline part. The question is then where to set the tradeoff between determinism - all decisions offline - and flexibility - some decisions online.

Conclusion Mixing system design issues and actual scheduling of TT and ET, offline and online scheduling appear very different. Separating activation paradigm, i.e., when is the scheduler invoked, from the actual scheduling reveals a different situation: the essential steps of (i) offline analysis and feasibility test, either via proof that no infeasible situation can occur or by constructing a complete schedule, (ii) determination of scheduling rules, whether in explicit forms, e.g., EDF, or implicit as scheduling table, and (iii) runtime execution according to these rules, are the same, albeit in different forms.

References

[1] Alan Burns. Programming real-time systems. In *ECRTS 04 - 16th EUROMICRO Conference on Real-Time Systems*, Catania, Sicily, July 2004.

[2] Hermann Kopetz. *Real-Time Systems - Design Principles for Distributed Embedded Applications.* Springer, 2011.

[3] Paulo Verissimo. Fundamental questions in the et vs. tt debate? please look elsewhere. In *Next-TTA Workshop on ET-TT Integration*, Grenoble, France, October 2002.