technische universität
dortmund

fakultät für informatik
informatik 12

Mapping of Applications to Platforms

Peter Marwedel
TU Dortmund, Informatik 12
Germany

(2010年 12 月 14 日)
Subset of slides selected for EECS 222C.

Scope of mapping algorithms

Useful terms from hardware synthesis:

- **Resource Allocation**
  Decision concerning type and number of available resources
- **Resource Assignment**
  Mapping: Task → (Hardware) Resource
- **xx to yy binding:**
  Describes a mapping from behavioral to structural domain, e.g. task to processor binding, variable to memory binding
- ➢ **Scheduling**
  Mapping: Tasks → Task start times
  Sometimes, resource assignment is considered being included in scheduling.

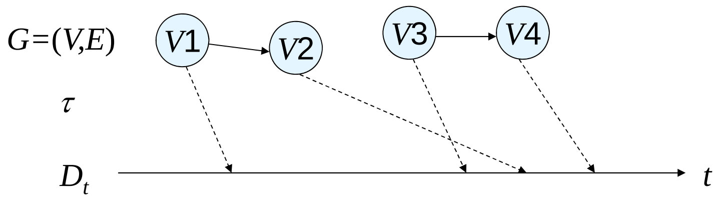
Classes of mapping algorithms considered in this course

- **Classical scheduling algorithms**
  Mostly for independent tasks & ignoring communication, mostly for mono- and homogeneous multiprocessors
- **Dependent tasks as considered in architectural synthesis**
  Initially designed in different context, but applicable
- **Hardware/software partitioning**
  Dependent tasks, heterogeneous systems, focus on resource assignment
- **Design space exploration using evolutionary algorithms;** Heterogeneous systems, incl. communication modeling

Real-time scheduling

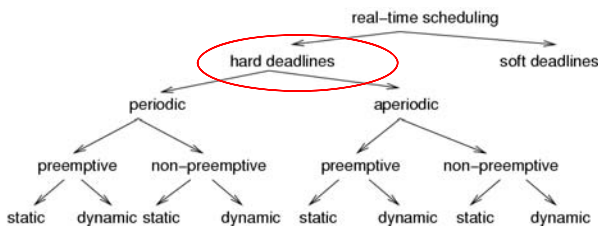
Assume that we are given a task graph $G=(V,E)$.

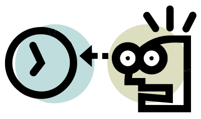
Def.: A **schedule** τ of G is a mapping
$$V \rightarrow D_t$$
of a set of tasks V to start times from domain D_t.



Typically, schedules have to respect a number of constraints, incl. resource constraints, dependency constraints, deadlines.
Scheduling = finding such a mapping.

Hard and soft deadlines



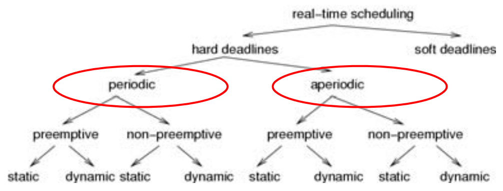


Def.: A time-constraint (deadline) is called **hard** if not meeting that constraint could result in a catastrophe [Kopetz, 1997].

All other time constraints are called **soft**.

We will focus on hard deadlines.

Periodic and aperiodic tasks

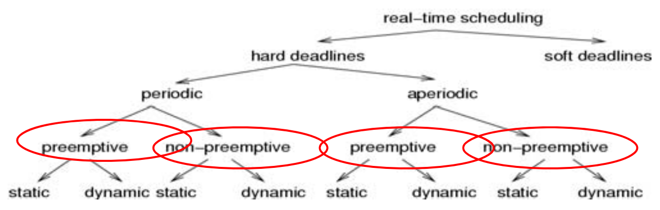


Def.: Tasks which must be executed once every *p* units of time are called **periodic** tasks. *p* is called their period. Each execution of a periodic task is called a **job**.

All other tasks are called **aperiodic**.

Def.: Tasks requesting the processor at unpredictable times are called **sporadic**, if there is a minimum separation between the times at which they request the processor.

Preemptive and non-preemptive scheduling

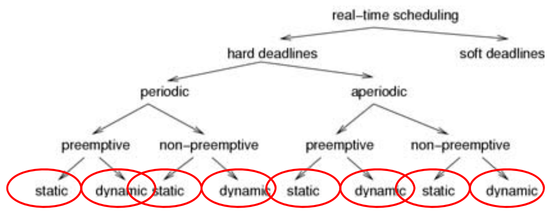


- **Non-preemptive schedulers:**
  Tasks are executed until they are done.
  Response time for external events may be quite long.
- **Preemptive schedulers:** To be used if
  - some tasks have long execution times or
  - if the response time for external events to be short.

Dynamic/online scheduling

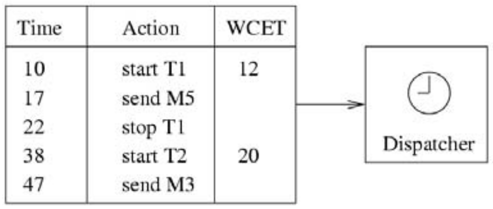
- **Dynamic/online scheduling:**
  Processor allocation decisions (scheduling) at run-time; based on the information about the tasks arrived so far.

HISTORY



Static/offline scheduling

- **Static/offline scheduling:**
  Scheduling taking a priori knowledge about arrival times, execution times, and deadlines into account. Dispatcher allocates processor when interrupted by timer. Timer controlled by a table generated at design time.

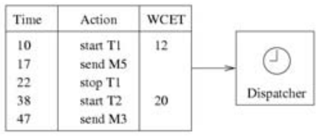


Time	Action	WCET
10	start T1	12
17	send M5	
22	stop T1	
38	start T2	20
47	send M3	

Time-triggered systems (1)

In an entirely time-triggered system, the temporal control structure of all tasks is established ***a priori*** *by off-line support-tools. This temporal control structure is encoded in a* ***Task-Descriptor List (TDL)*** *that contains the cyclic schedule for all activities of the node. This schedule considers the required precedence and mutual exclusion relationships among the tasks such that an explicit coordination of the tasks by the operating system at run time is not necessary. ..*

The dispatcher is activated by the synchronized clock tick. It looks at the TDL, and then performs the action that has been planned for this instant [Kopetz].



Time	Action	WCET
10	start T1	12
17	send M5	
22	stop T1	
38	start T2	20
47	send M3	

Time-triggered systems (2)

… pre-run-time scheduling is often the only practical means of providing predictability in a complex system. [Xu, Parnas].

It can be easily checked if timing constraints are met.
The disadvantage is that the response to sporadic events may be poor.

Centralized and distributed scheduling

- **Mono- and multi-processor scheduling:**
  - Simple scheduling algorithms handle single processors,
  - more complex algorithms handle multiple processors.
    - algorithms for homogeneous multi-processor systems
    - algorithms for heterogeneous multi-processor systems (includes HW accelerators as special case).
- **Centralized and distributed scheduling:**
  Multiprocessor scheduling either locally on 1 or on several processors.

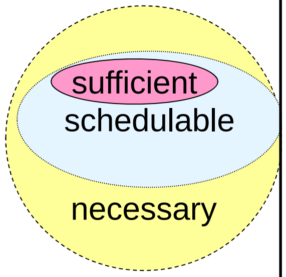
Schedulability

Set of tasks is **schedulable** under a set of constraints, if a schedule exists for that set of tasks & constraints.

Exact tests are NP-hard in many situations.

Sufficient tests: sufficient conditions for schedule checked. (Hopefully) small probability of not guaranteeing a schedule even though one exists.

Necessary tests: checking necessary conditions. Used to show no schedule exists. There may be cases in which no schedule exists & we cannot prove it.

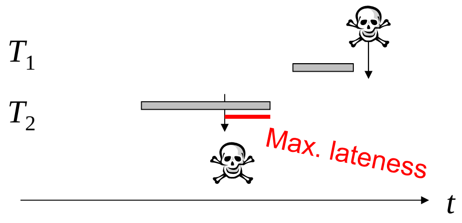


Cost functions

Cost function: Different algorithms aim at minimizing different functions.

Def.: Maximum lateness =

max$_{\textbf{all tasks}}$ (completion time – deadline)
Is <0 if all tasks complete before deadline.


T_1
T_2
Max. lateness
t