Technische Universität Dresden

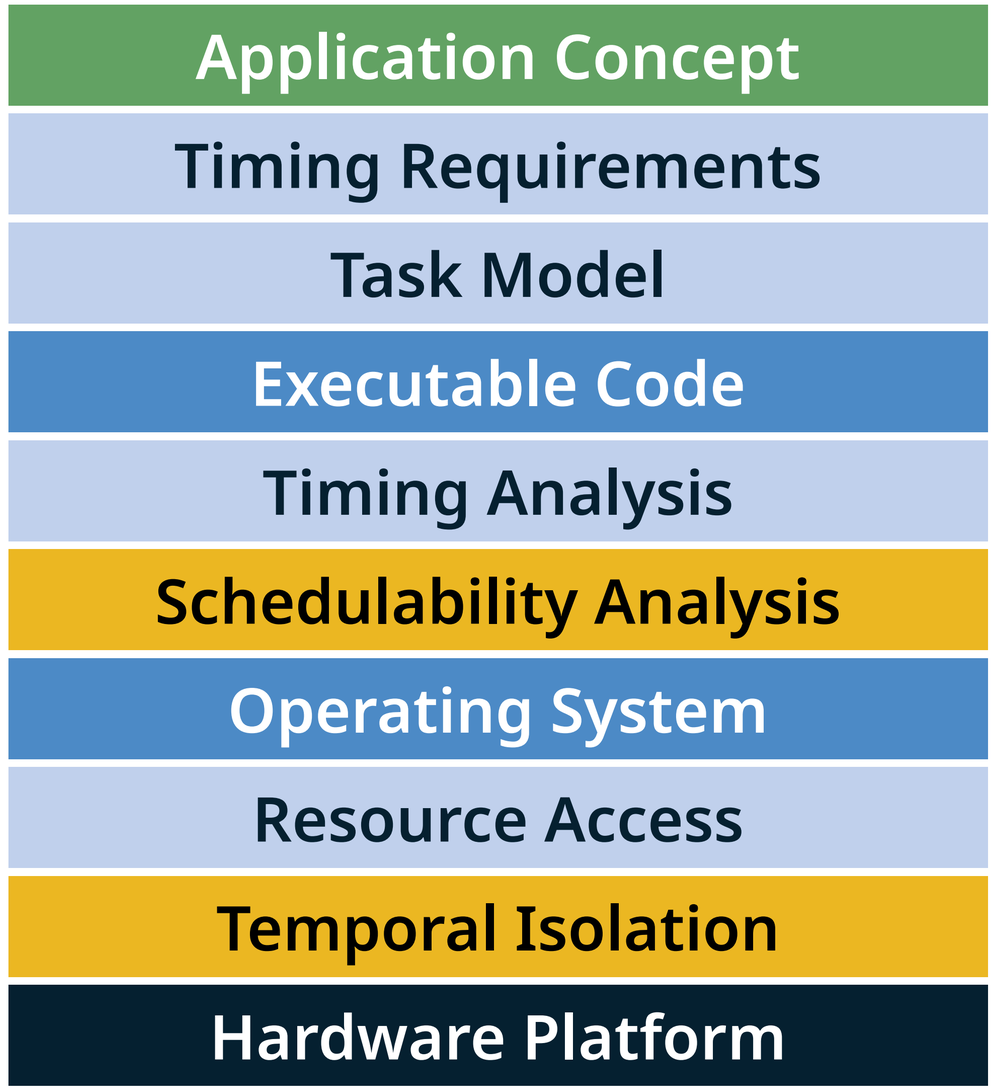
barkhausen institut

Time-Driven and Partitioned Systems

Real-Time Systems

Michael Roitzsch

Real-Time Systems Technology Stack



Time-Driven vs. Event-Driven Scheduling



Time-Driven System

- at design time, a feasible schedule is computed
- the schedule is stored in a table
- *at certain points in time,* the scheduler dispatches tasks

Event-Driven System

- at design time, the existence of a feasible schedule for a set of tasks is determined depending on the scheduling algorithm
- *at certain events,* the scheduler computes the next scheduling decision and dispatches tasks accordingly

Time-Driven Systems

Properties

- decisions, which job to execute next at specific time instants
- these are chosen a priori (before system begins execution)
- schedule is computed offline (possibly by searching over all schedules)

Typically Assumptions: closed, deterministic system

- fixed number of tasks in the system
- with a priori known parameters (fixed inter-release times)
- tasks must be ready at their release times

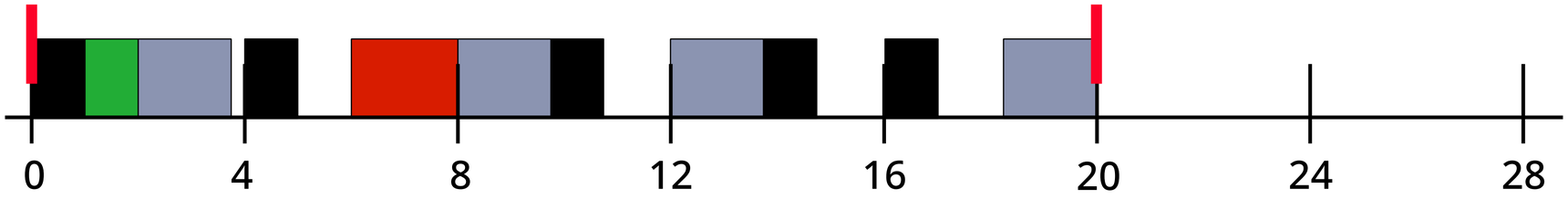
often used for safety-critical hard real-time systems

Deriving a Time-Driven Schedule

- it is sufficient to somehow (by search) find schedule for hyper-period

Example

- Tasks (p_i, e_i): (4,1) (5,1.8) (20,1) (20,2)
- hyperperiod: 20
- arbitrary possible schedule for one hyperperiod:



Unused parts can be used for aperiodic jobs

Executing a Time-Driven Schedule

store all scheduling points (t_i, $T(t_i)$) in a table

```
Do
   set timer to next decision point
   run current job in table
   wait for timer
Done
```

- scheduling actions at pre-determined instants in time
- repeats after one hyper-period
- contrast to: priority driven systems make scheduling decisions at events (job release, job completion)
- safety aspect: you may want to police for job overruns caused by defects

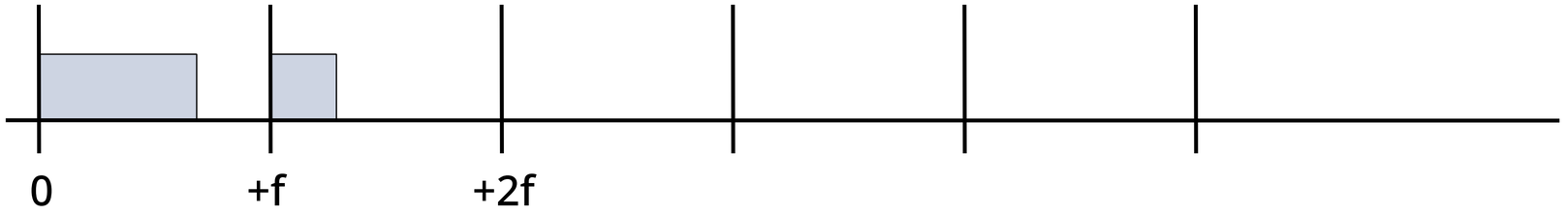
Technische Universität Dresden

barkhausen institut

Cyclic Schedules

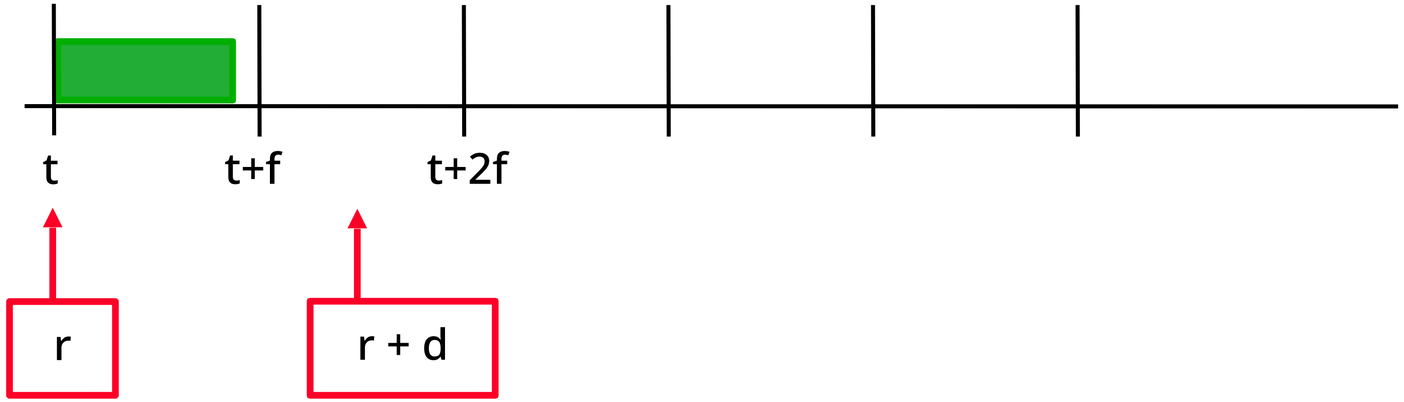
Cyclic Schedules

- also called synchronous systems, tick-driven systems
- scheduling actions only at **periodic instants** of time
- timeline divided into *minor frames*, hyper-period called *major frame*
- at minor frame borders:
  - check for execution time violations from previous frame
  - run the job from the next frame
- question: What frame size?



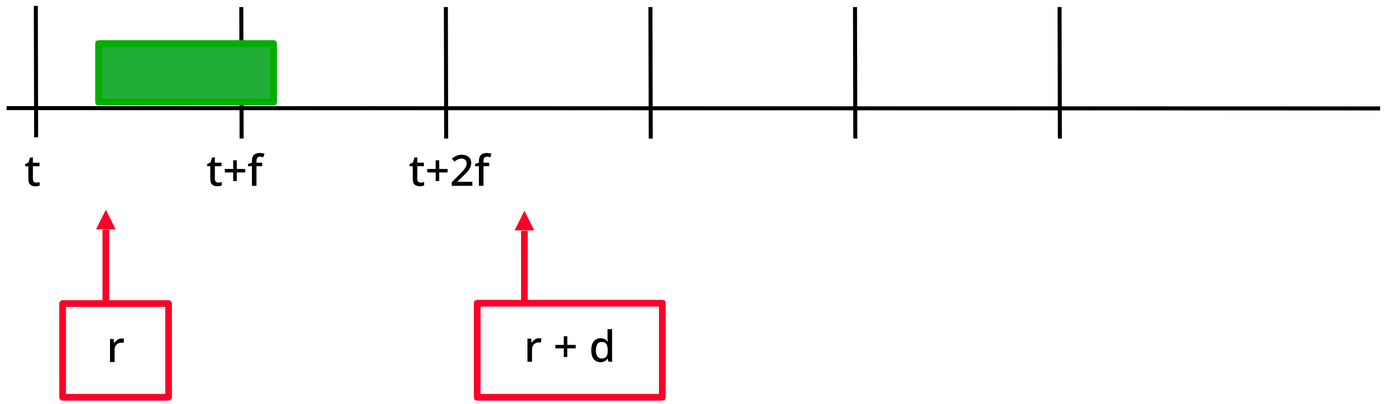
Frame Size Constraints

1. at least one period should be multiple of f
   ensures an integer number of frames per hyperperiod
2. $f \geq \max(e_i)$
   avoids preemption
3. one full frame between release time r and deadline d for each job
   to enable the scheduler to police overruns before the deadline



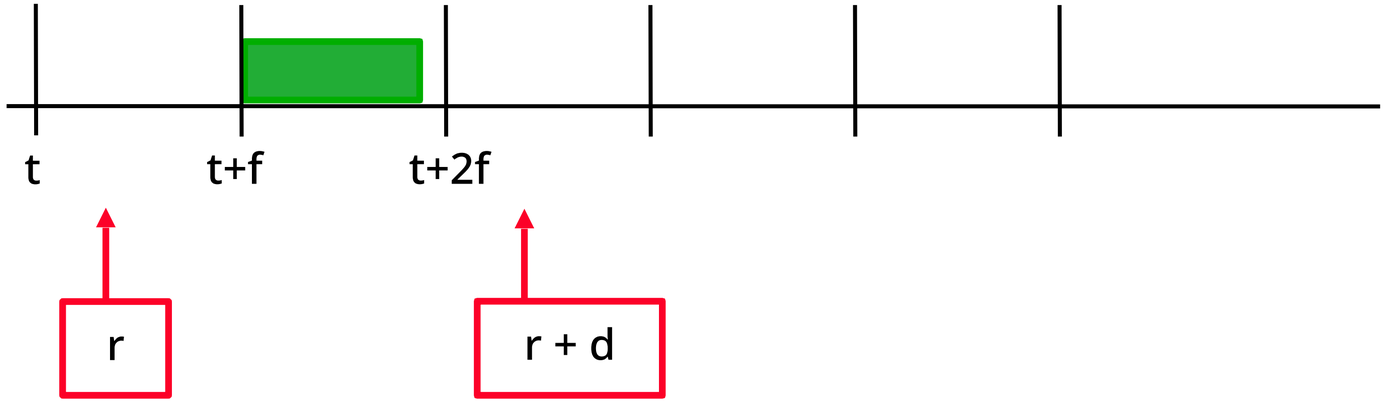
Frame Size Constraints

1. at least one period should be multiple of f
   ensures an integer number of frames per hyperperiod
2. $f \geq \max(e_i)$
   avoids preemption
3. one full frame between release time r and deadline d for each job
   to enable the scheduler to police overruns before the deadline



Frame Size Constraints

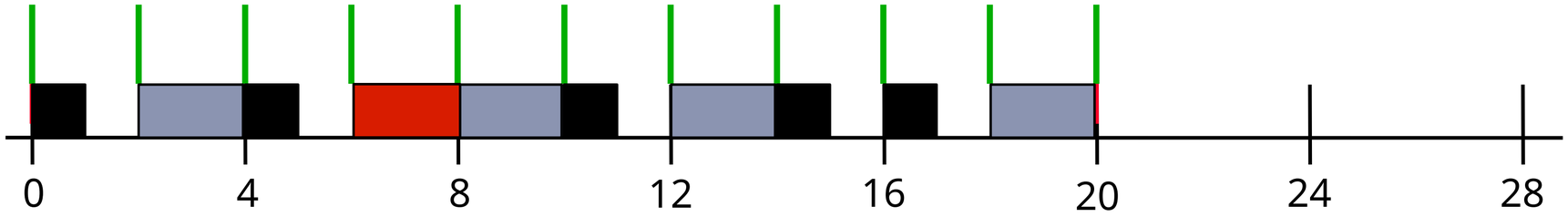
1. at least one period should be multiple of f
   ensures an integer number of frames per hyperperiod
2. $f \geq \max(e_i)$
   avoids preemption
3. one full frame between release time r and deadline d for each job
   to enable the scheduler to police overruns before the deadline



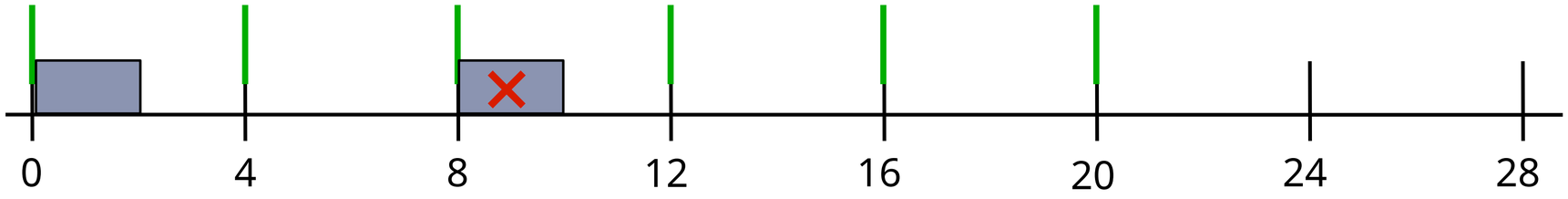
Example

(4,1) (5,2) (20,2)

Frame Size 2:



Frame Size 4:



No check for overrun at 10!

Example

(4,1) (5,2) (20,5)

Not schedulable, as (20,5) covers an entire job window of (4,1) and (5,2)

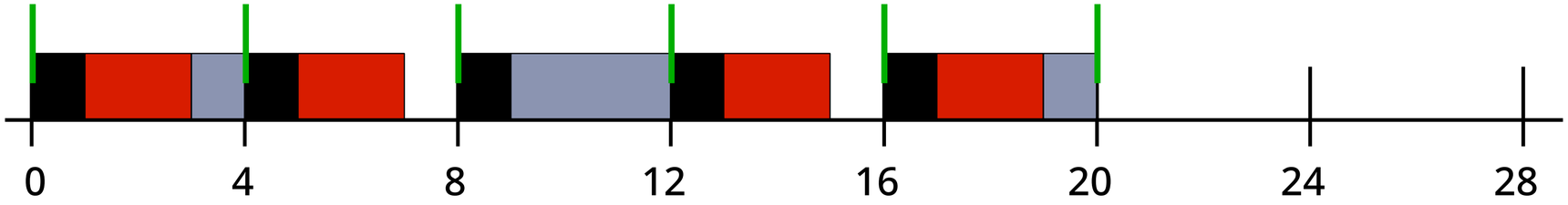
We need to split and recombine jobs …

Job Slicing

- decompose jobs in **slices**
- combine multiple slices into **scheduling blocks:** subroutines called back-to-back

Example: (4,1) (5,2) (20,5)

- cut (20,5) in (20,1) (20,3) (20,1)
- frame size: 4



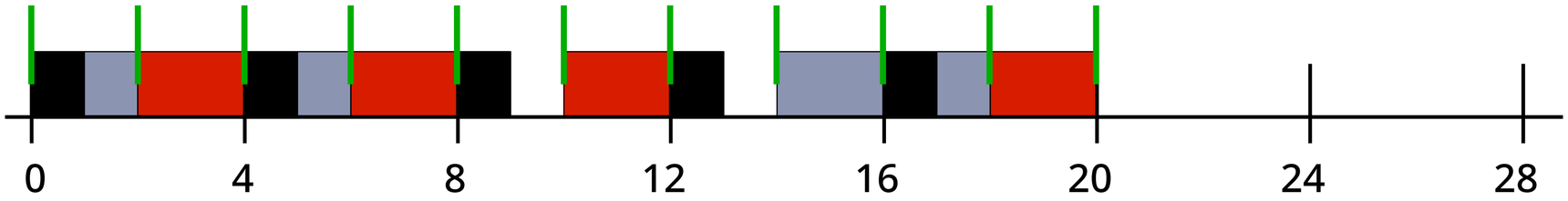
Problems

- if job 2 of T_1 does not fully use its WCET, T_2 runs early
- if job 3 of T_2 overruns, scheduler only detects at time 16

Job Slicing

Better solution

- (4,1) (5,2) (20,5)
- cut (20,5) in (20,1) (20,1) (20,2) (20,1)
- frame size: 2



Cyclic Executive



```
current time t := 0
current frame k := 0


at every f time units DO
   react if prior scheduling block has not completed
   t := t+f
   k := k+1 mod hyperperiod
   get scheduling block from cyclic schedule
   execute scheduling block
   /* slack time here … */
DONE
```

Accommodating Aperiodic and Sporadic Jobs



- use slack time in minor frames that is not allocated to scheduling block
- **slack stealing:** execute aperiodic jobs before planned scheduling block to improve their response time

Acceptance Test for Sporadic Jobs

- task model: preemptible sporadic jobs with known deadline and WCET
- sum(slack times in all frames before d) ≥ WCET
- generate sporadic slices that fit in frames (similar to a server)
- static flavor: store slices in time-driven schedule
- dynamic flavor: queue slices according to EDF

Additional Considerations



Mode Changes

- precompute cyclic schedule for all modes
- reconfigure cyclic schedule when mode changes

Critical Sections

- split jobs into slices at critical section boundaries
- schedule ensures exclusivity, no lock/unlock operations needed

Interference from Hardware

- analysable impact of caches or context switches
- scheduling on multiple processor possible

Technische Universität Dresden

barkhausen institut

Partitioned Systems

Partitioned Systems



Usage scenario

- no interference between subsystems
  - prevents misbehaving subsystems to damage one another
  - no timing anomalies
- separate, systematic test of subsystems, deterministic behavior
- prevents some timing covert channels

Space Partitioned Systems



allocate each resource to a system partition

Examples

- disk partitioning
- main memory
- IO devices
- caches
- cores in multicore processors
- accelerators (like GPUs)

Time Partitioned Systems



divide time into slots, allocate each slot to a system partition

Examples

- CPU execution
- busses: memory, PCI, field busses (like CAN)
- network: Time-Triggered Ethernet

Problems

- synchronizing time partitions across multiple resources
- interaction of different resources, example: DMA and memory bus
- hidden resource state may introduce side channels