Scheduling Analysis

Part 1

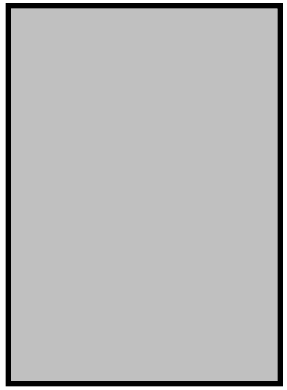
Event Models

Time Driven Scheduling

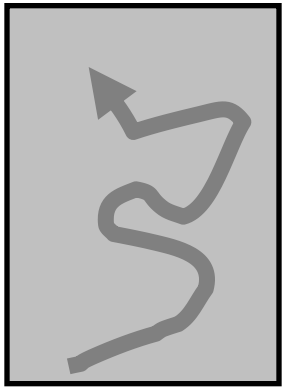
Simulation vs. Analysis

- The problem with simulation
  - Only covers one particular execution path
  - Hard to simulate corner cases
  - Does not scale with problem size
- Analysis guarantees correctness
  - Utilizes abstract models to describe system properties
  - Ongoing research (SymTA/S tool developed at IDA)

State Space

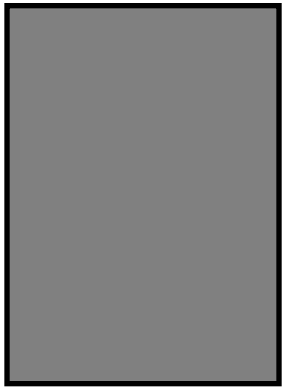


Simulation



Single path

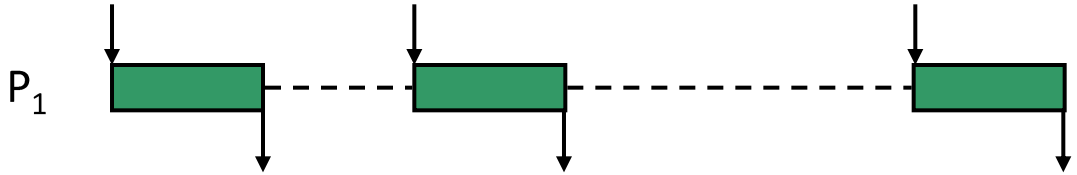
Analysis



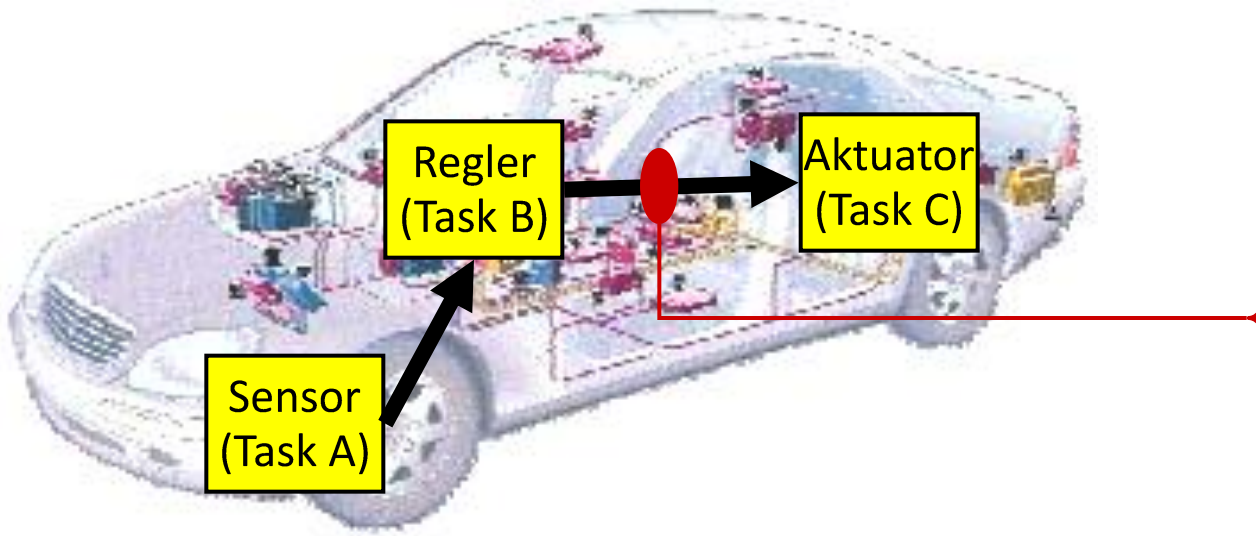
Full coverage

Event Streams

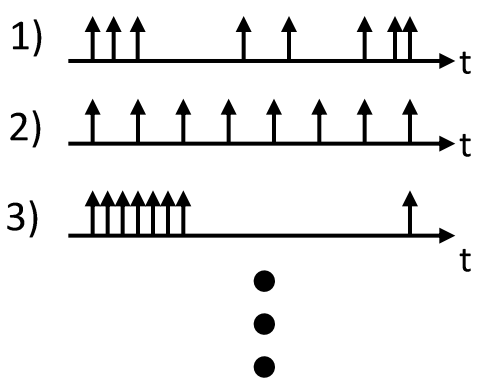
- Tasks are activated by events
  - May trigger successor tasks by generating new events



- Event streams
  - Communication between tasks
  - Properties depending on task behavior → multiple traces possible

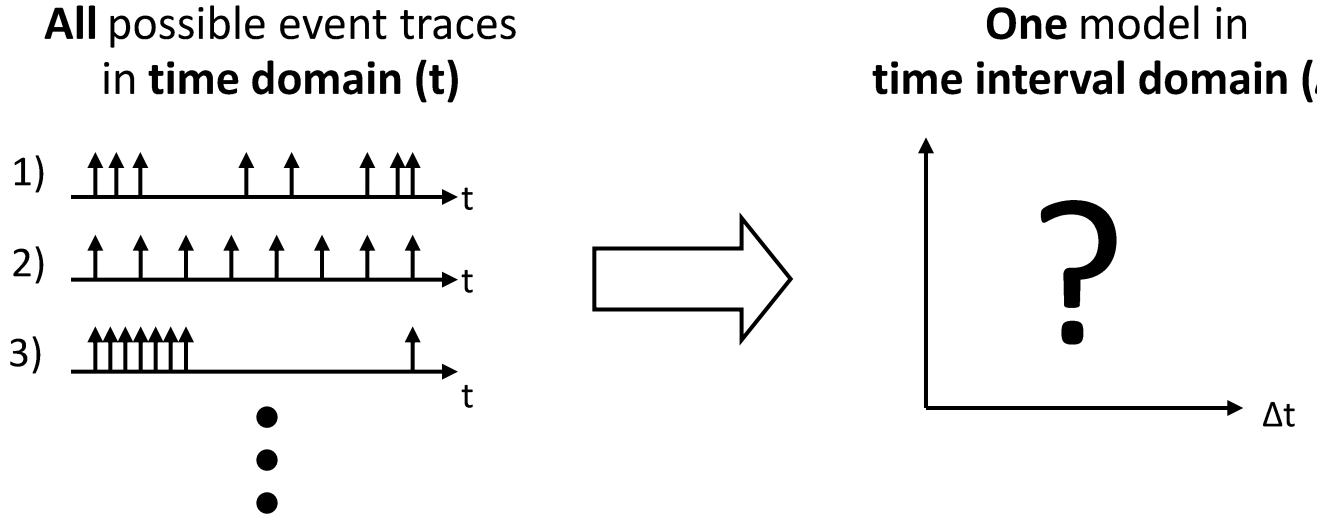


Example traces at event stream:



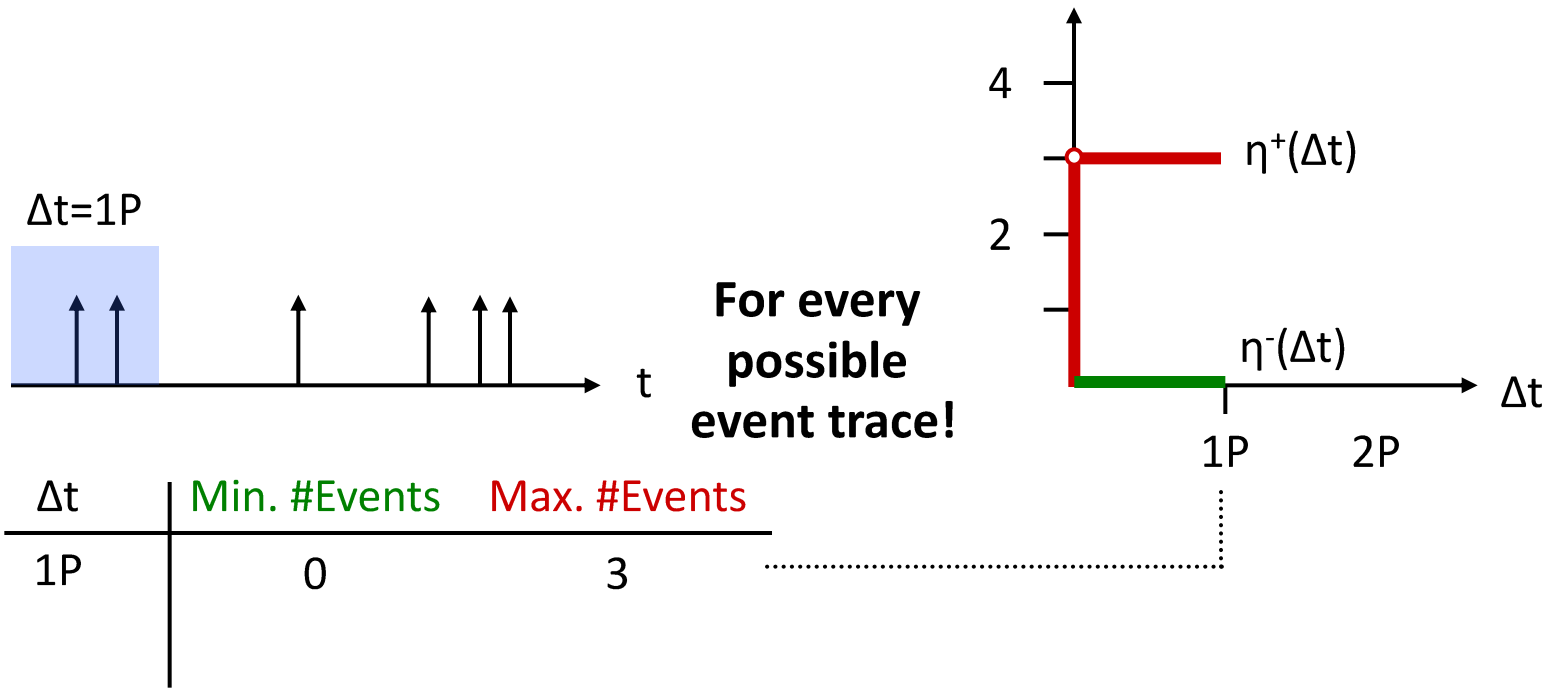
Event Streams - Characterization

- Characterization of event streams
  - Instead of investigating *all* individual event traces and their timing, formal schedulability analysis performs *one* analysis on the **event bounds**
- All activating events within a time window of given size Δt
  - $\eta^+(\Delta t)$: Maximum number of events in window Δt
  - $\eta^-(\Delta t)$: Minimum number of events in window Δt



Time Domain to Time Interval Domain Transformation (1/2)

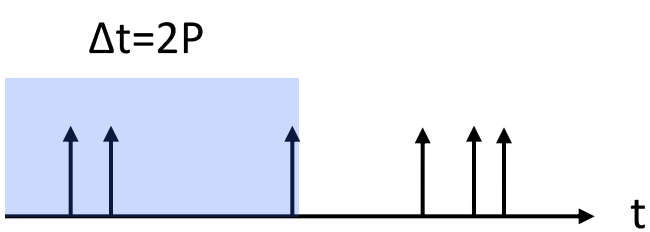
- Arrival Curves in **time interval domain (Δt)**
  - $\eta^+(\Delta t)$: Maximum number of events in window Δt
  - $\eta^-(\Delta t)$: Minimum number of events in window Δt



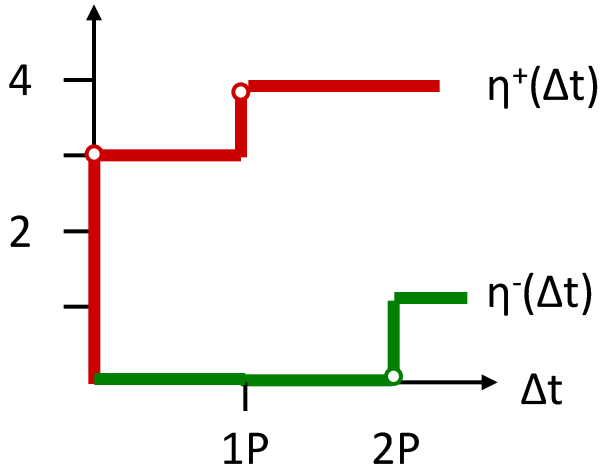
Δt	Min. #Events	Max. #Events
1P	0	3

Time Domain to Time Interval Domain Transformation (2/2)

- Arrival Curves in **time interval domain (Δt)**
  - $\eta^+(\Delta t)$: Maximum number of events in window Δt
  - $\eta^-(\Delta t)$: Minimum number of events in window Δt



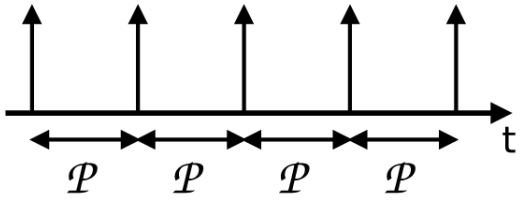
For every possible event stream!



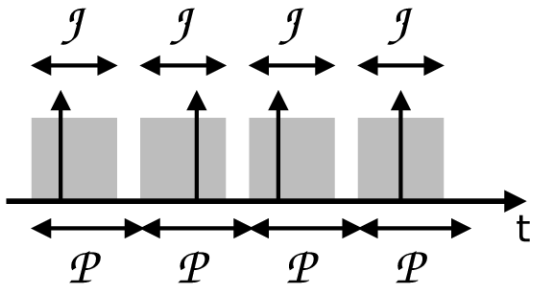
Δt	Min. #Events	Max. #Events
1P	0	3
2P	1	4

Event Models

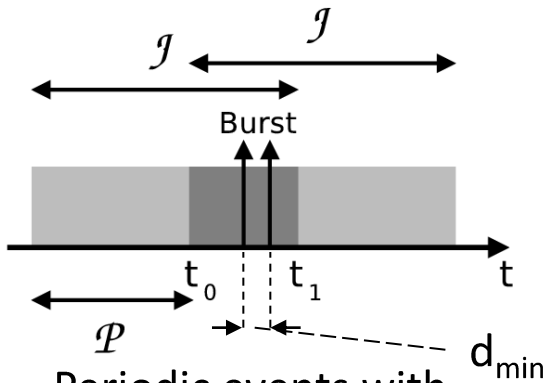
- Abstraction in **time domain (t)**
  - Describe all traces on one event stream through a limited parameter set
- Perform formal analysis on this parameter set: (P,J,d_{min})
  - P = Period
  - J = Jitter
  - d_{min} = Minimum distance between two consecutive events



Periodic events



Periodic events with jitter

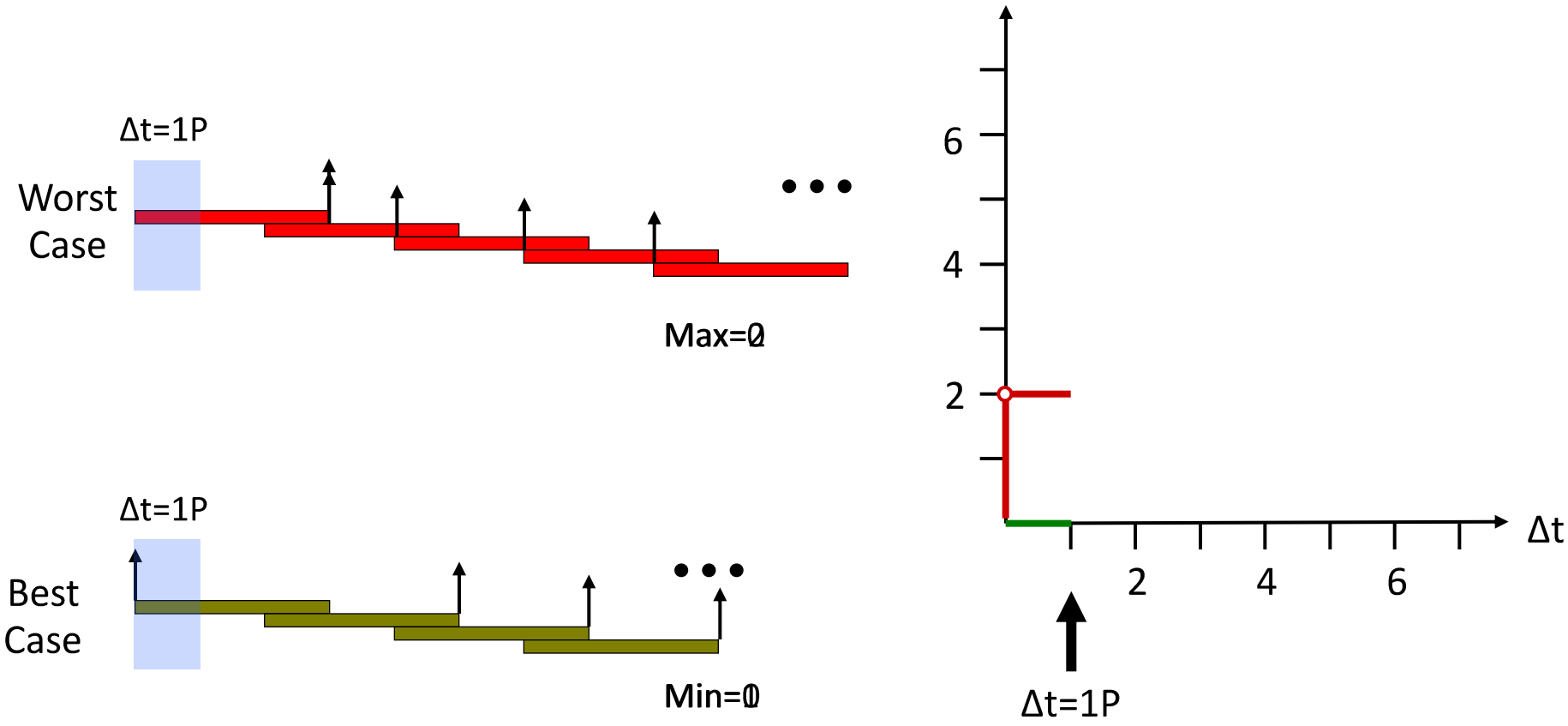


Periodic events with jitter and burst

- Allows conservative transformation from the actual event timing
  - Only one analysis necessary instead of checking all event correlations

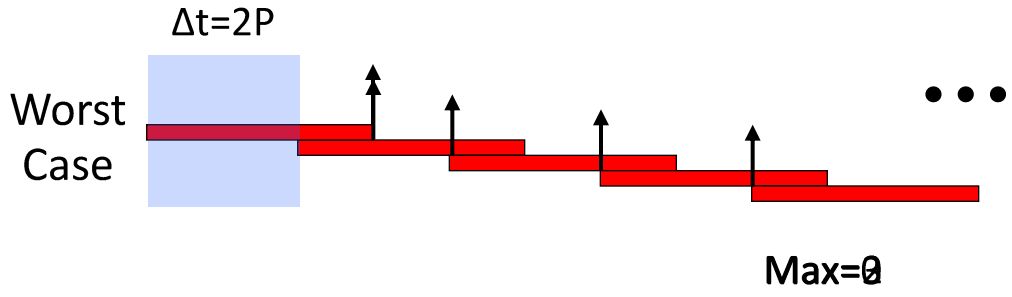
Event Model: Periodic with Jitter (Δt=1P)

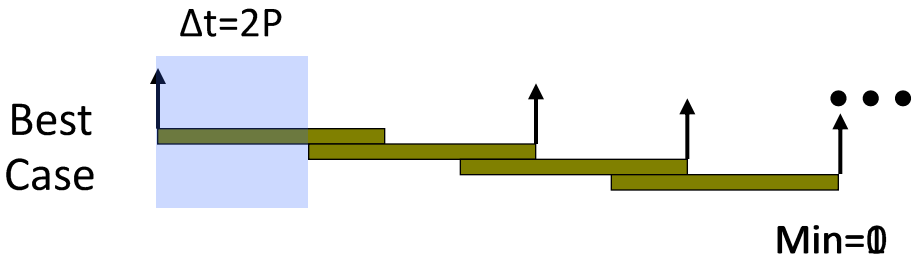
- Event model: (2,3,0)
- Analyze all possible event traces with Δt=1P
  - Due to abstraction (2,3,0), we know the best and worst case streams

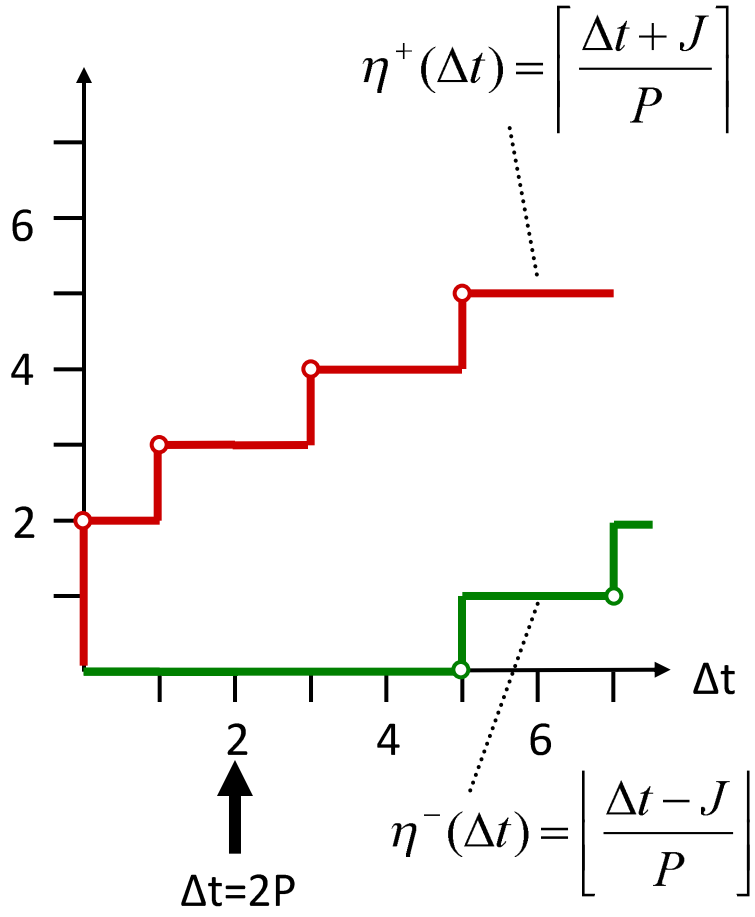


Event Model: Periodic with Jitter (Δt=2P)

- Event model: (2,3,0)
- Analyze all possible event streams with Δt=2P
  - Due to abstraction (2,3,0), we know the best and worst case streams

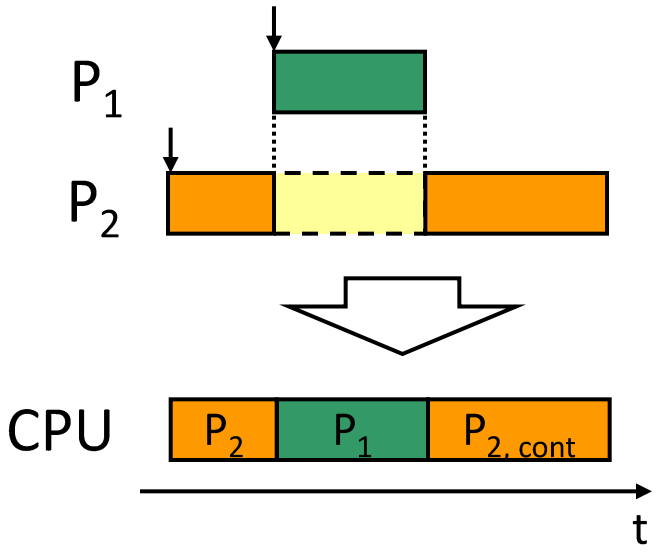






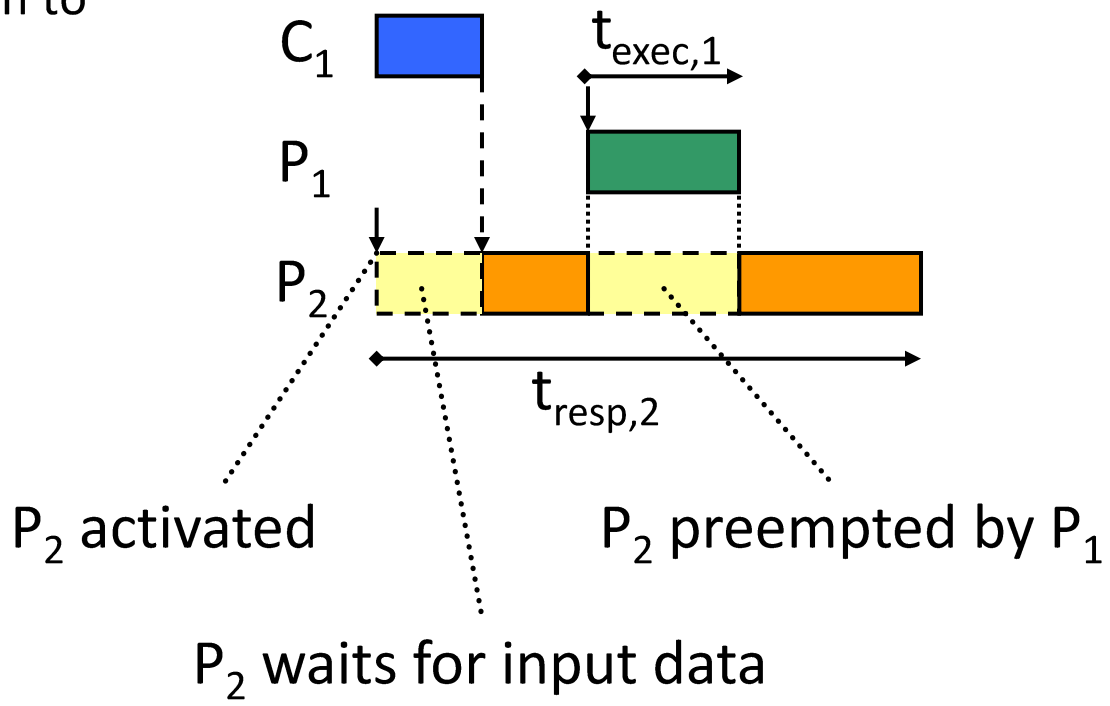
Reminder: Scheduling

- Task
  - Set of instructions
  - (Sub)program
- Scheduling
  - Temporal partitioning of processing or communication resources
  - Sequenced task execution
- Scheduler
  - Assigns tasks to available (usually limited) resources
- Preemption
  - Temporary interruption of a task
  - No cooperation required by the interrupted task
  - Interrupted task is resumed eventually



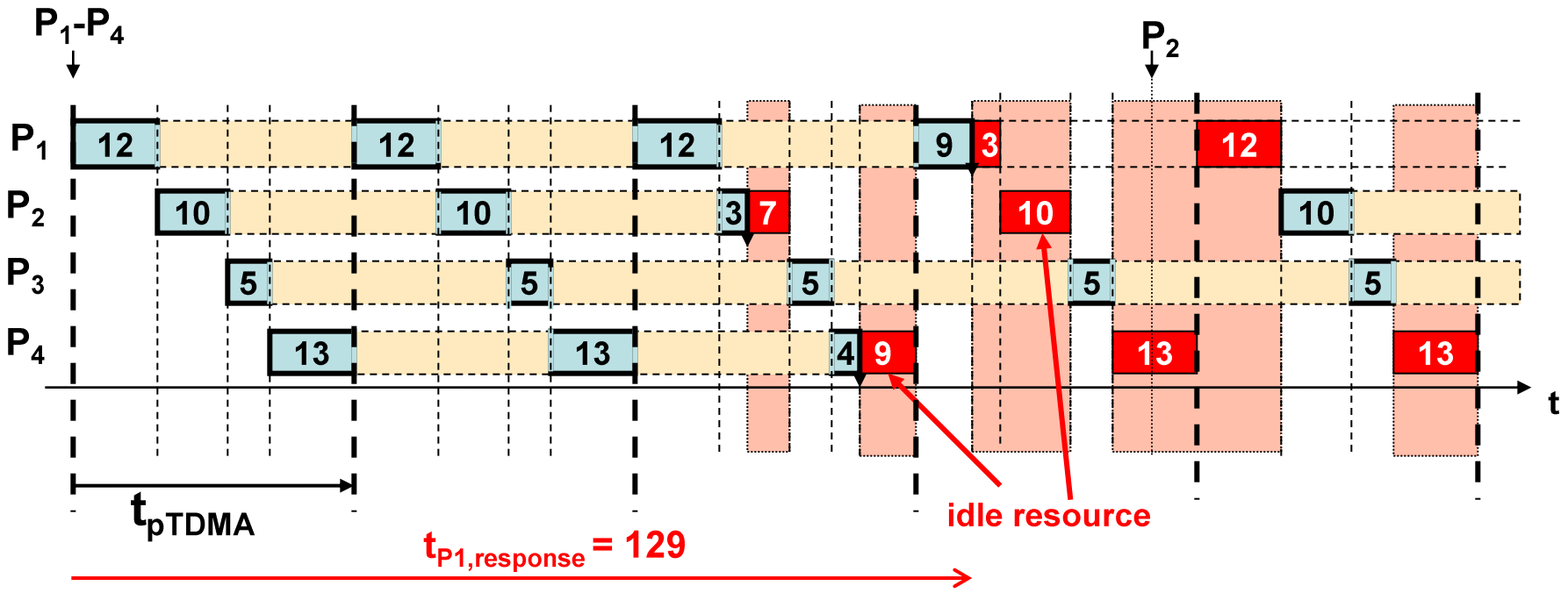
Scheduling Analysis

- What guarantees can be made given a certain environment?
- Worst-case
  - E.g. highest resource usage/bus congestion a certain task experiences
- Response time
  - Time from task activation to end of execution
  - $t_{resp} \geq t_{exec}$
- Gantt-Charts
  - Illustrate schedules
  - Start and finish times
  - Dependencies



TDMA Scheduling

- Static time-driven scheduling
  - Each task assigned to a specific time slot in reoccurring TDMA round
  - Good analytical properties
  - May lead to inefficient resource utilization



TDMA Analysis

Core execution time

Preemption by other tasks

Max. TDMA rounds needed for one execution

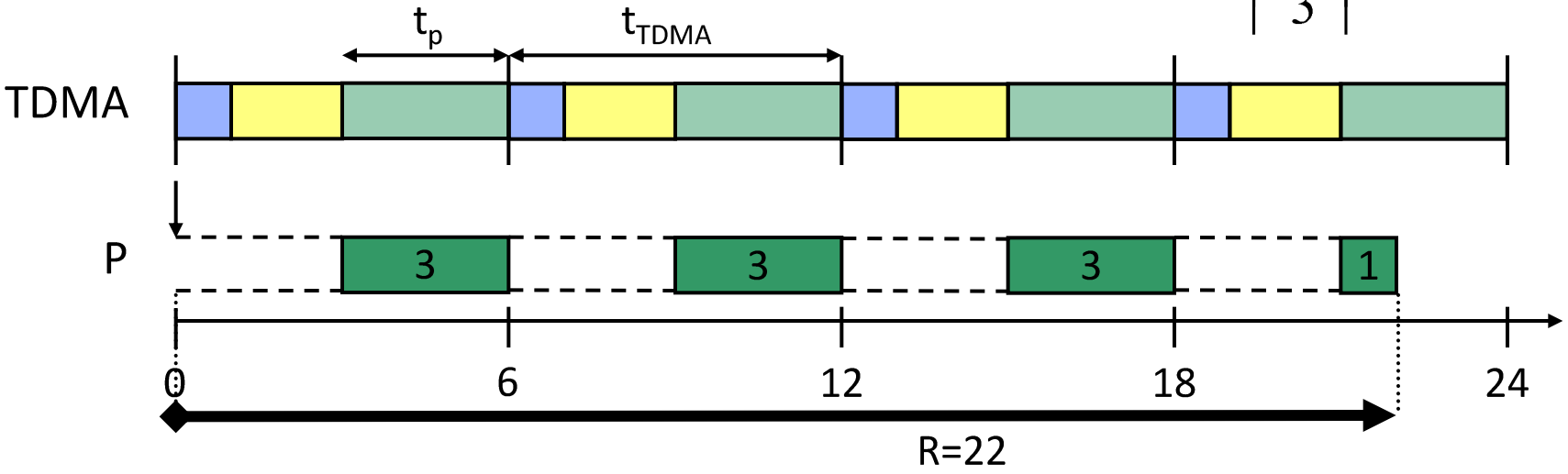
$$R_i = C_i + \left(t_{TDMA} - t_{P_i} \right) \cdot \left\lceil \frac{C_i}{t_{P_i}} \right\rceil$$

Accumulated time of preemption until task finishes

Example:

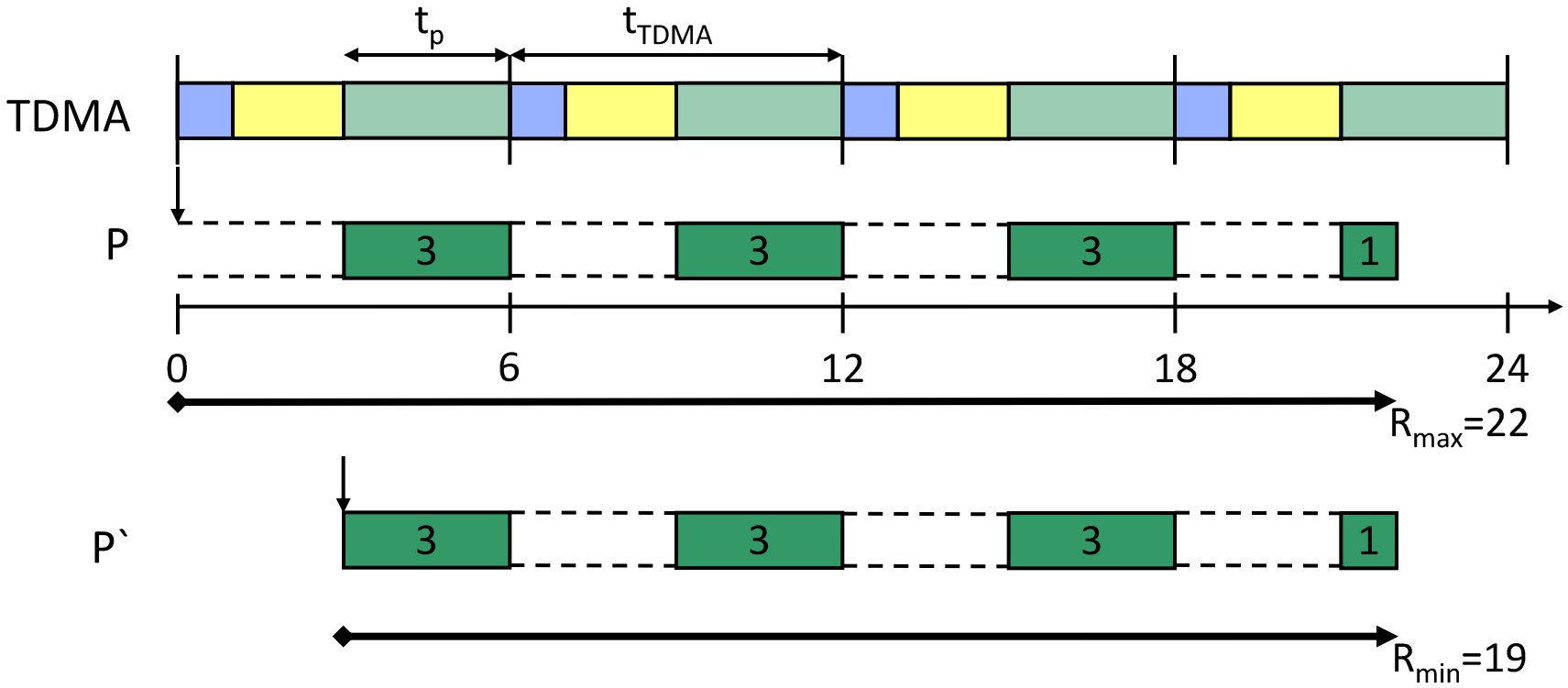
C=10, t_P=3, t_{TDMA}=6

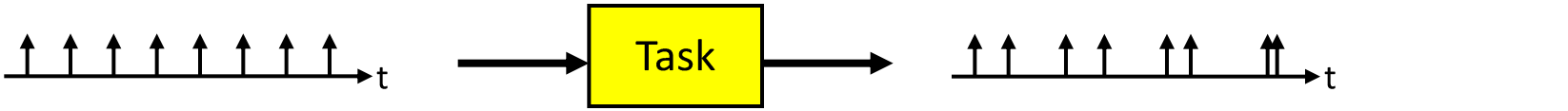
$$R = 10 + (6 - 3) \cdot \left\lceil \frac{10}{3} \right\rceil = 22$$



TDMA Output Jitter

- Different response times cause output jitter



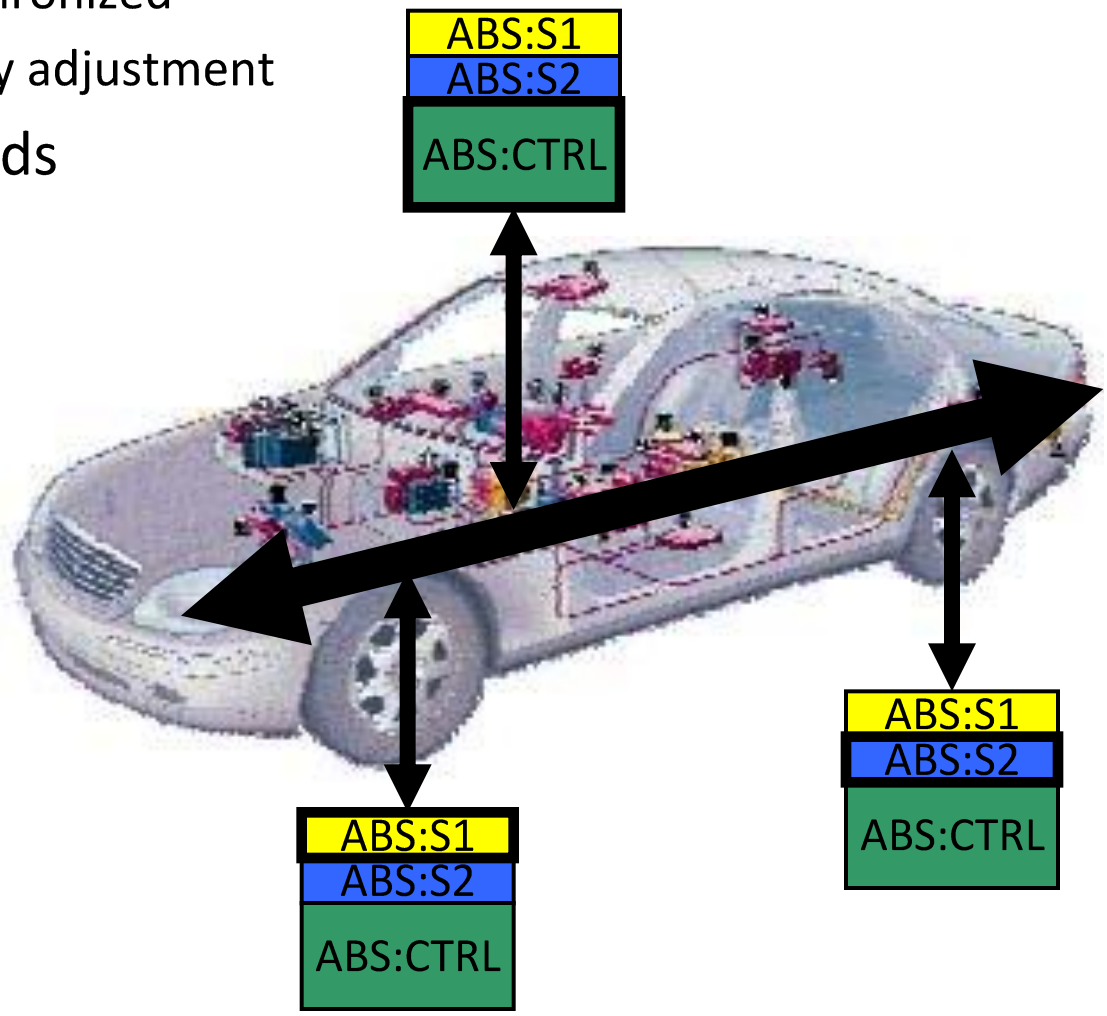


Response time: [19, 22]

Output Jitter: $J_{out}=J_{in}+R_{max}-R_{min}$

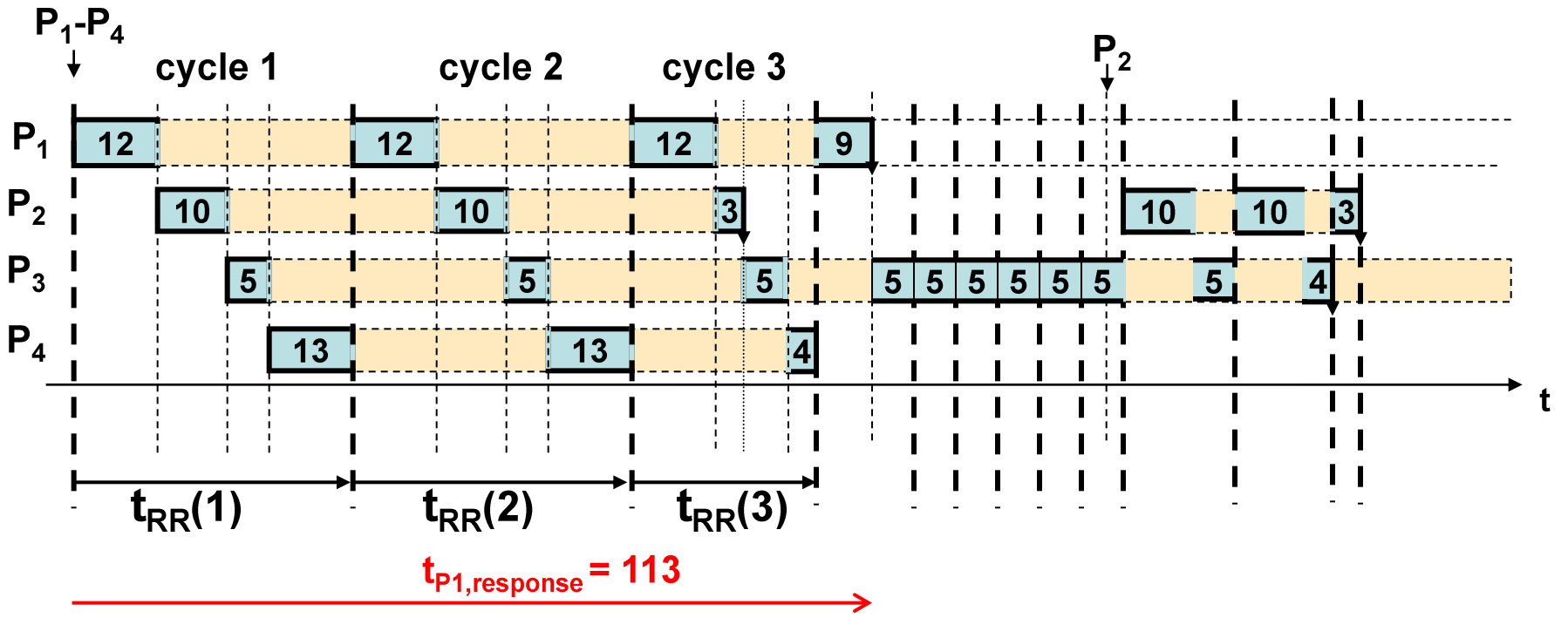
TDMA – Practical Issues

- Clock synchronization in distributed embedded systems
  - All task have to be synchronized
    - Phase and frequency adjustment
- Synchronization Methods
  - Every task knows the entire TDMA schedule
  - “Bus Sniffing”
- After Synchronization
  - No need for dedicated arbitration
- Output event model
  - May create output jitter due to differing R_i



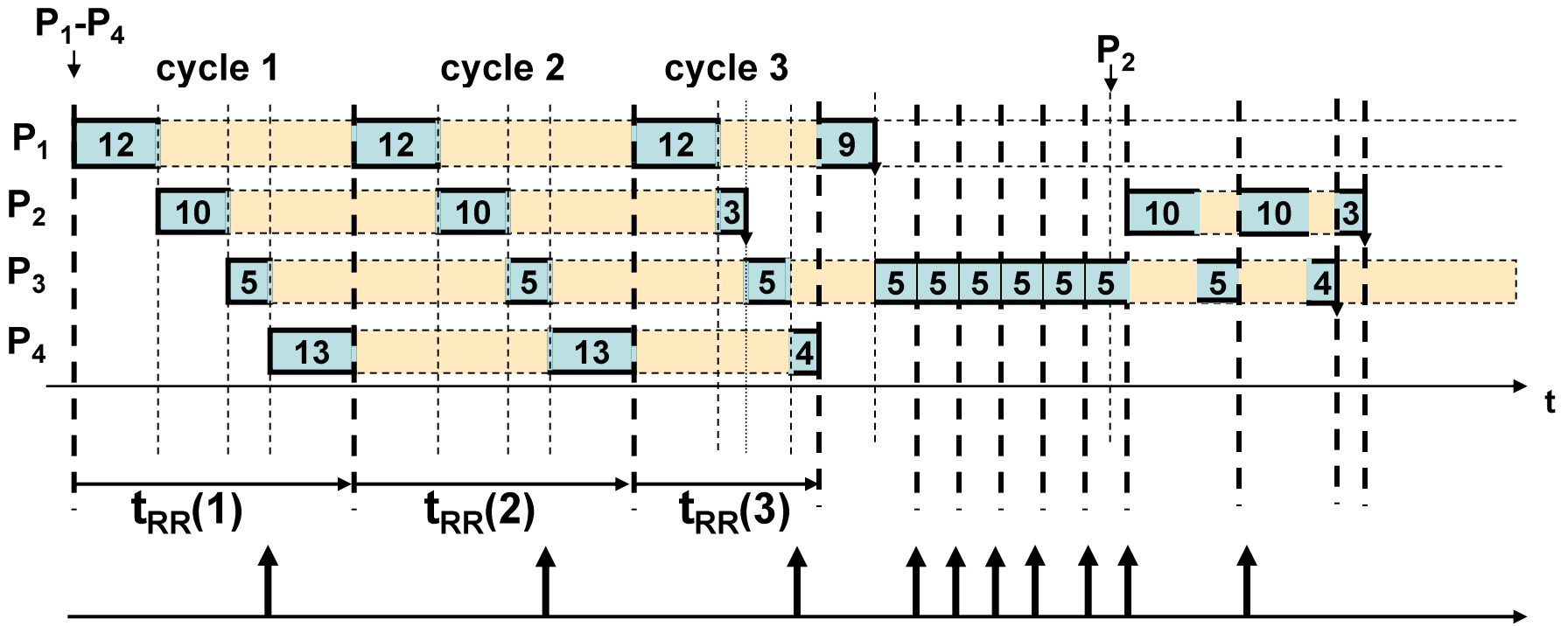
Round Robin Scheduling

- Dynamic time-driven scheduling
  - Cyclic task execution
  - Task release resources when execution finishes (no forced idle times)
    - Better resource utilization than TDMA



Round Robin - Practical Issues (1/2)

- Dynamic behavior makes analysis more difficult than TDMA
- Output event model
  - May create output jitter and bursts
    - Example: During execution P_3 generates an event every 5 clock cycles



Round Robin - Practical Issues (2/2)

- Clock synchronization in distributed embedded systems
  - Synchronization requirements (like TDMA)
  - Implementation more challenging than TDMA
    - Does a task want to send data?
    - Next task to schedule?
  - Control Overhead