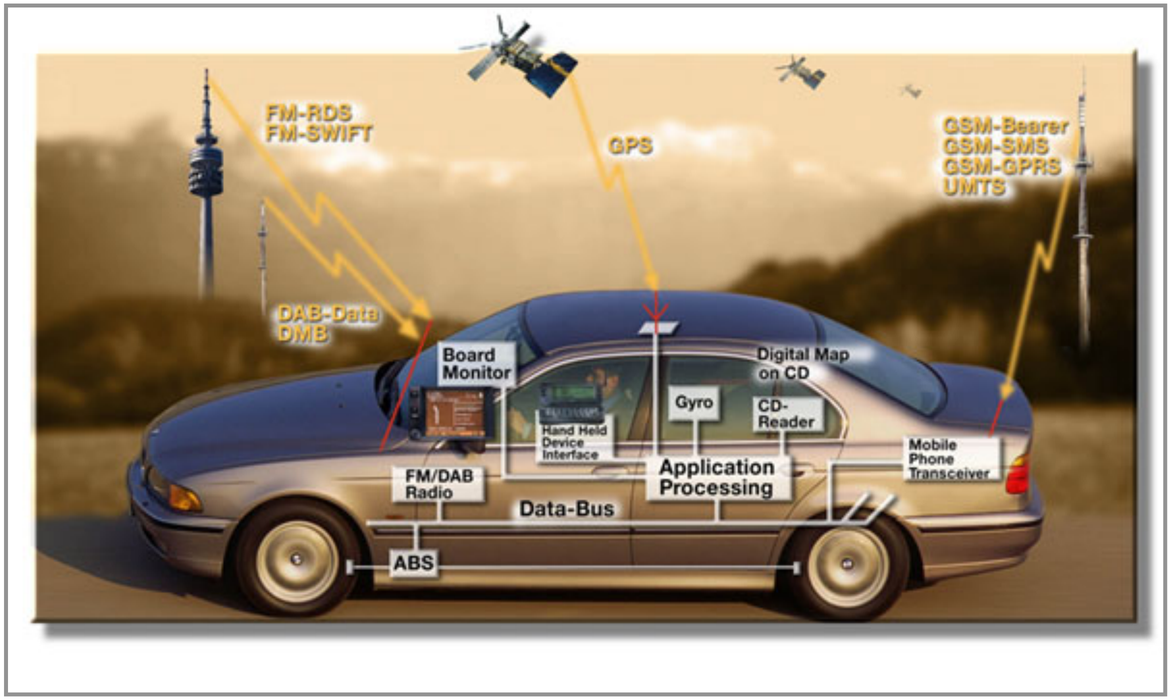
General Concepts

Introduction to real-time systems

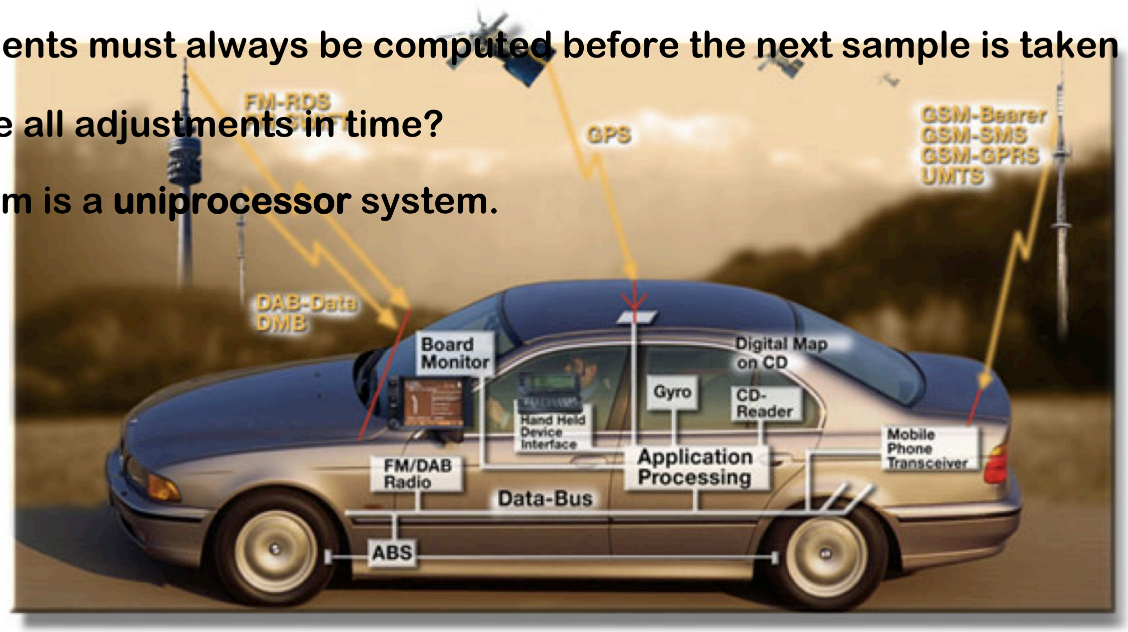
Review

- What is a real-time system?
- What is an embedded system?
- What characteristic of a real-time system is probably the most important?



The schedulability question: Drive-by-Wire Example

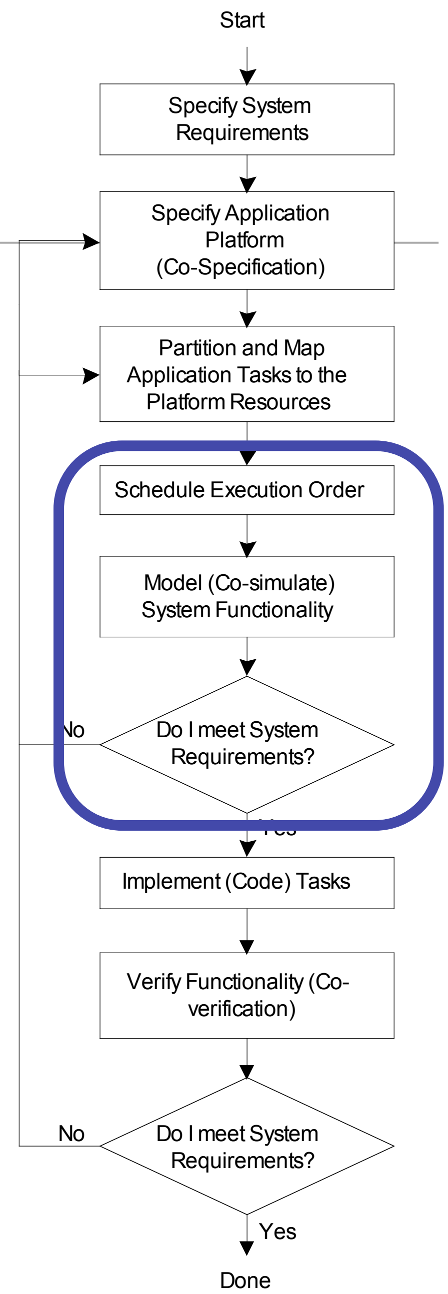
- Consider a control system in a future vehicle
  - Steering wheel sampled every 10 ms – wheel positions adjusted accordingly (computing the adjustment takes 4.5 ms of CPU time)
  - Brakes sampled every 4 ms – break pads adjusted accordingly (computing the adjustment takes 2ms of CPU time)
  - Velocity is sampled every 15 ms – acceleration is adjusted accordingly (computing the adjustment takes 0.45 ms)
  - For safe operation, adjustments must always be computed before the next sample is taken
- Is it possible to always compute all adjustments in time?
- The underlying computer system is a uniprocessor system.



The system design process

- Designing any computer system involves many steps.
- Some steps are common to many types of systems.
- A few steps are more important in a real-time system.
    - Scheduling is one such operation.
    - How do we know if a set of tasks can be scheduled in a predictable manner?

- We will touch upon other parts of the design process later in the course.

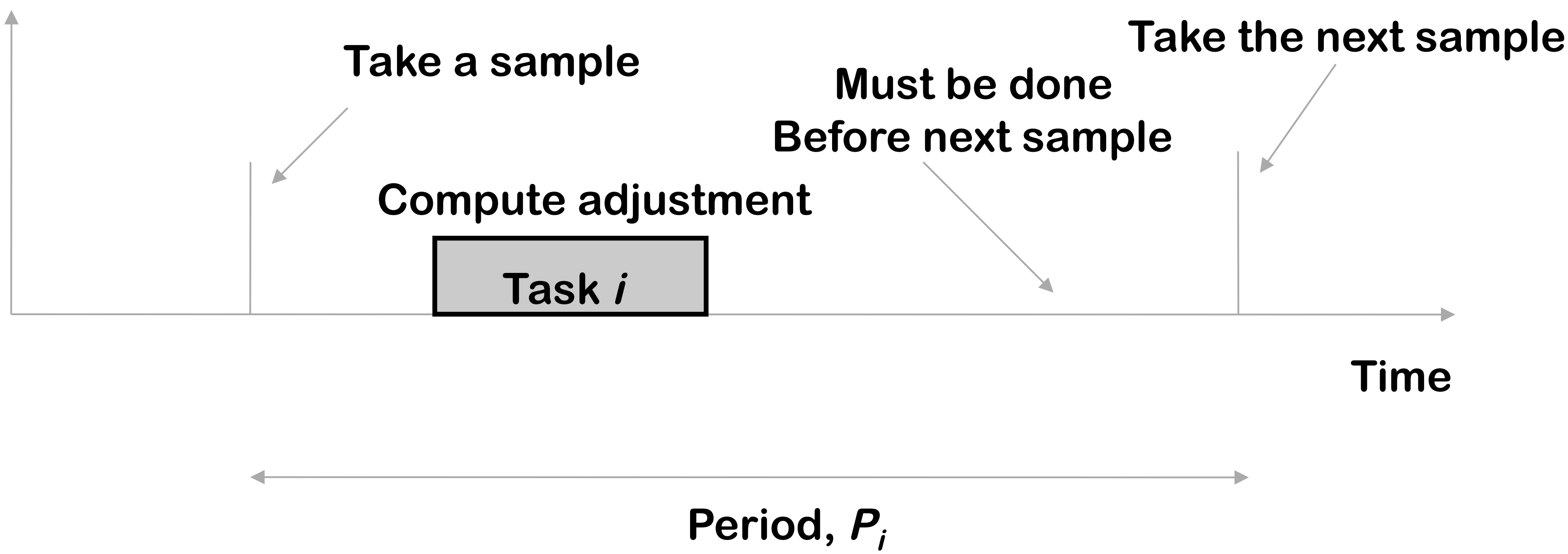


The schedulability question: Drive-by-Wire Example

- Consider a control system in a future vehicle
  - Steering wheel sampled every 10 ms – wheel positions adjusted accordingly (computing the adjustment takes 4.5 ms of CPU time)
  - Brakes sampled every 4 ms – break pads adjusted accordingly (computing the adjustment takes 2ms of CPU time)
  - Velocity is sampled every 15 ms – acceleration is adjusted accordingly (computing the adjustment takes 0.45 ms)
  - For safe operation, adjustments must always be computed before the next sample is taken
- Is it possible to always compute all adjustments in time?
- The underlying computer system is a **uniprocessor** system.

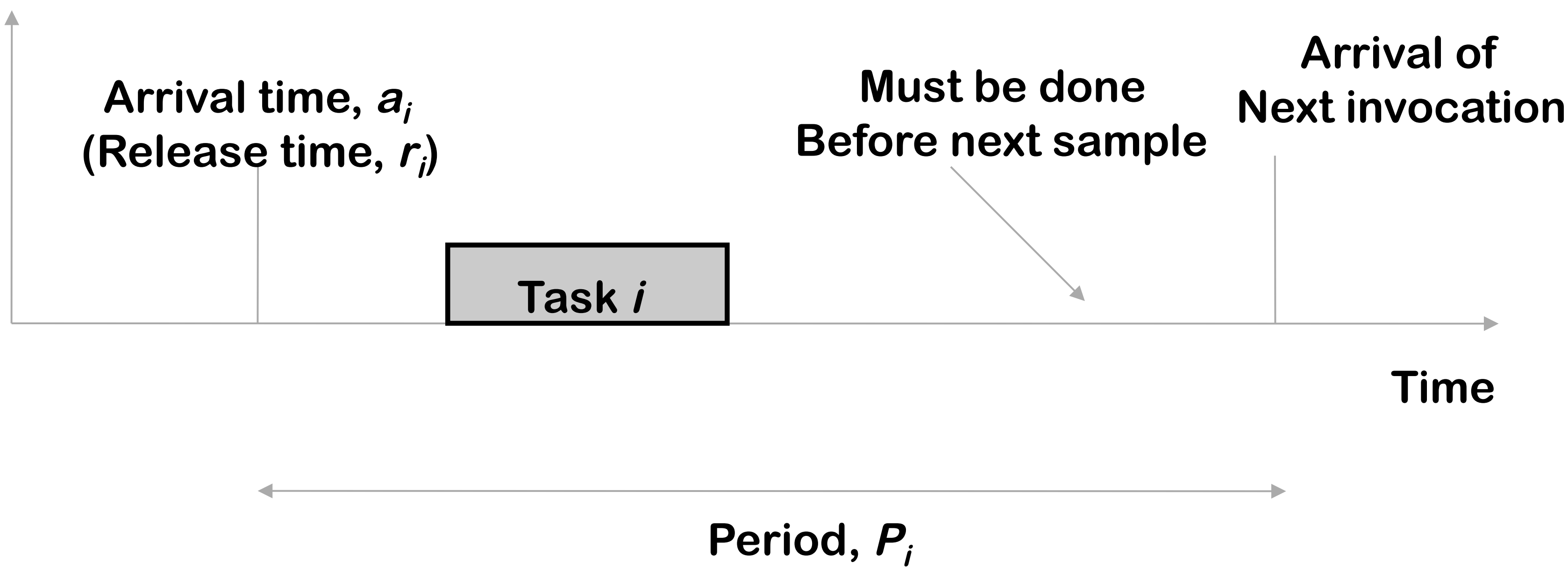
Some terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



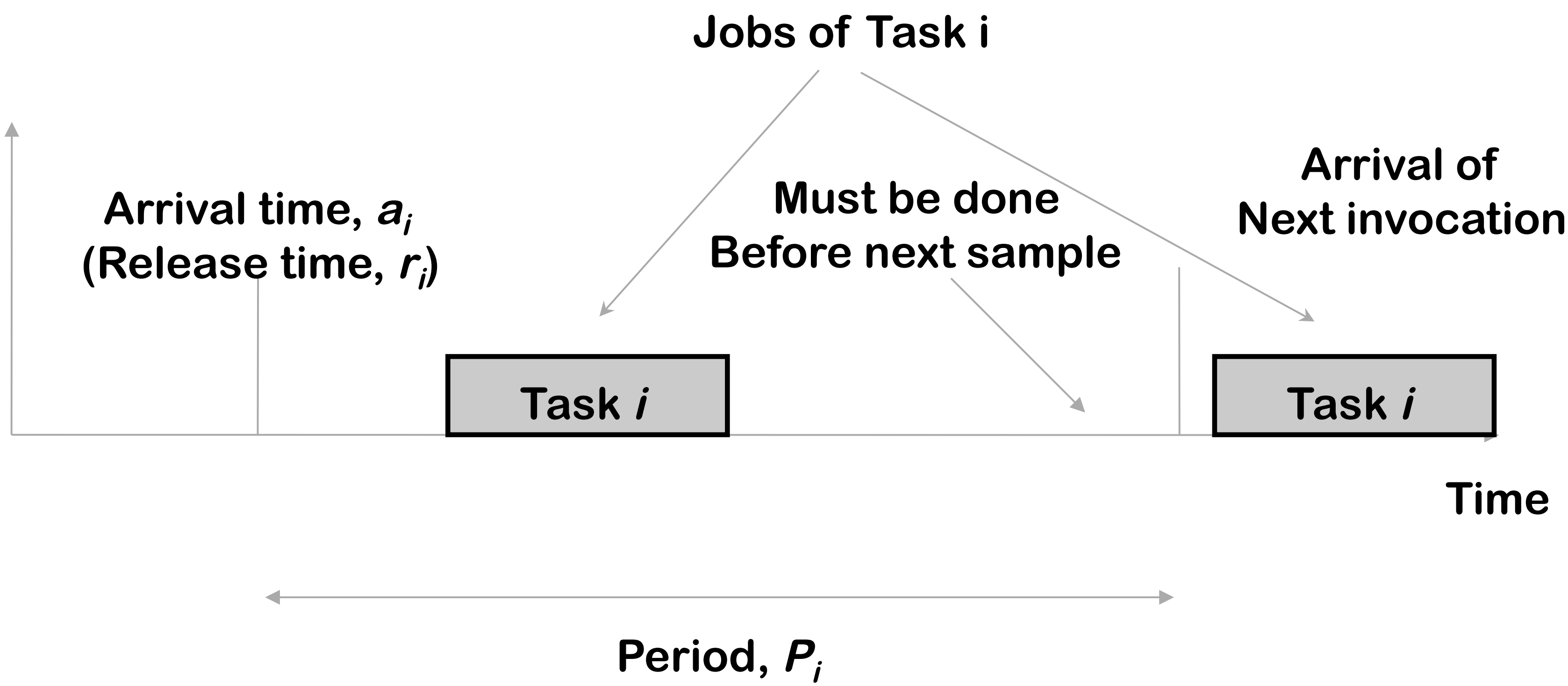
Some terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



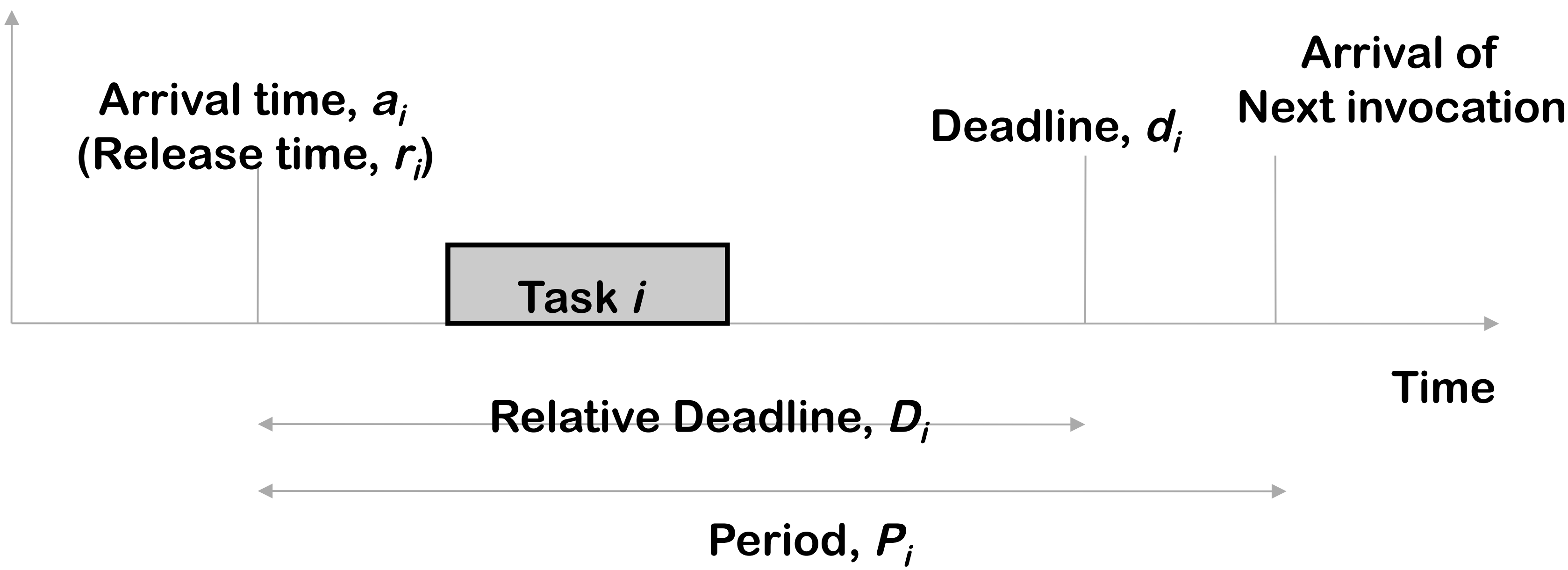
Some terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.
- Each invocation of a task is sometimes called a “job.”
- A common assumption is that arrival times for the first job of all tasks is 0.



Some terminology

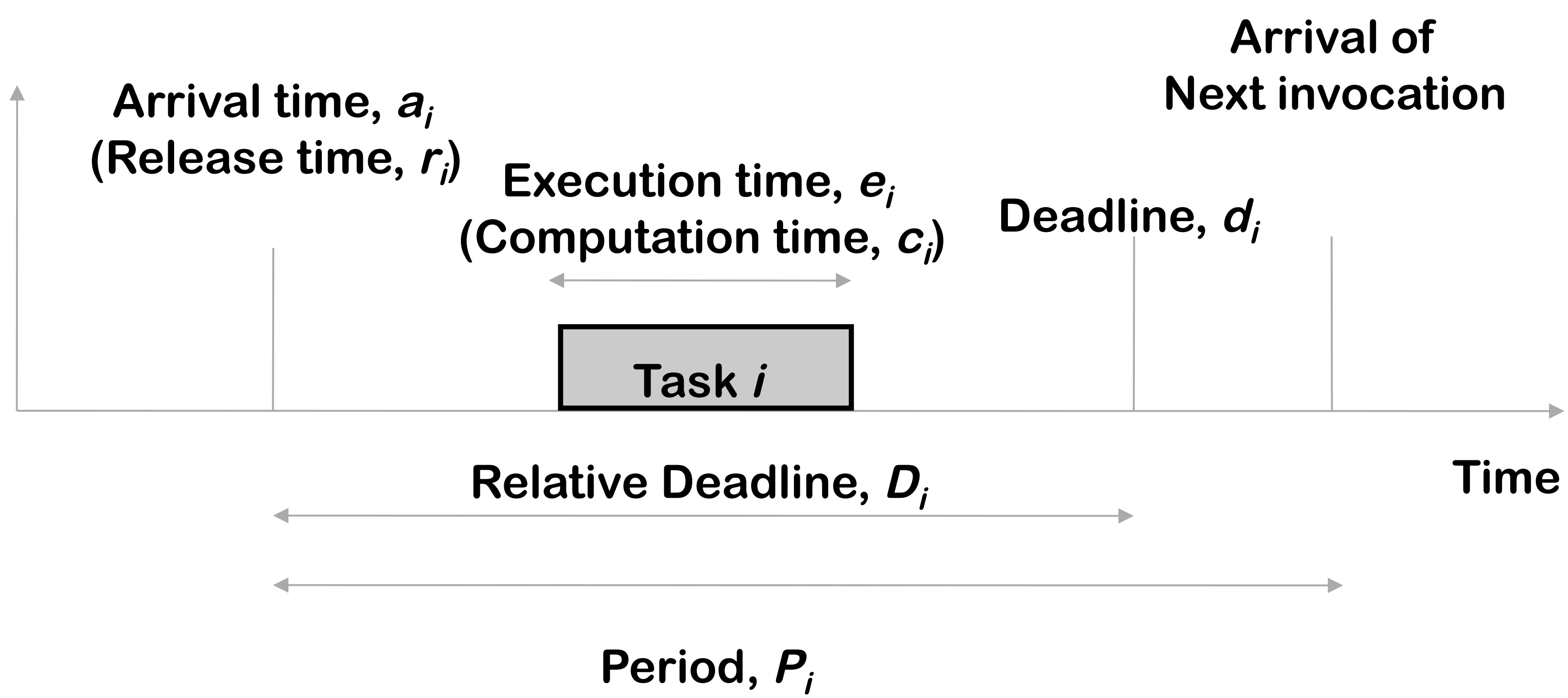
- Tasks, periods, arrival-time, deadline, execution time, etc.



(absolute deadline) d_i = (release time) r_i + (relative deadline) D_i

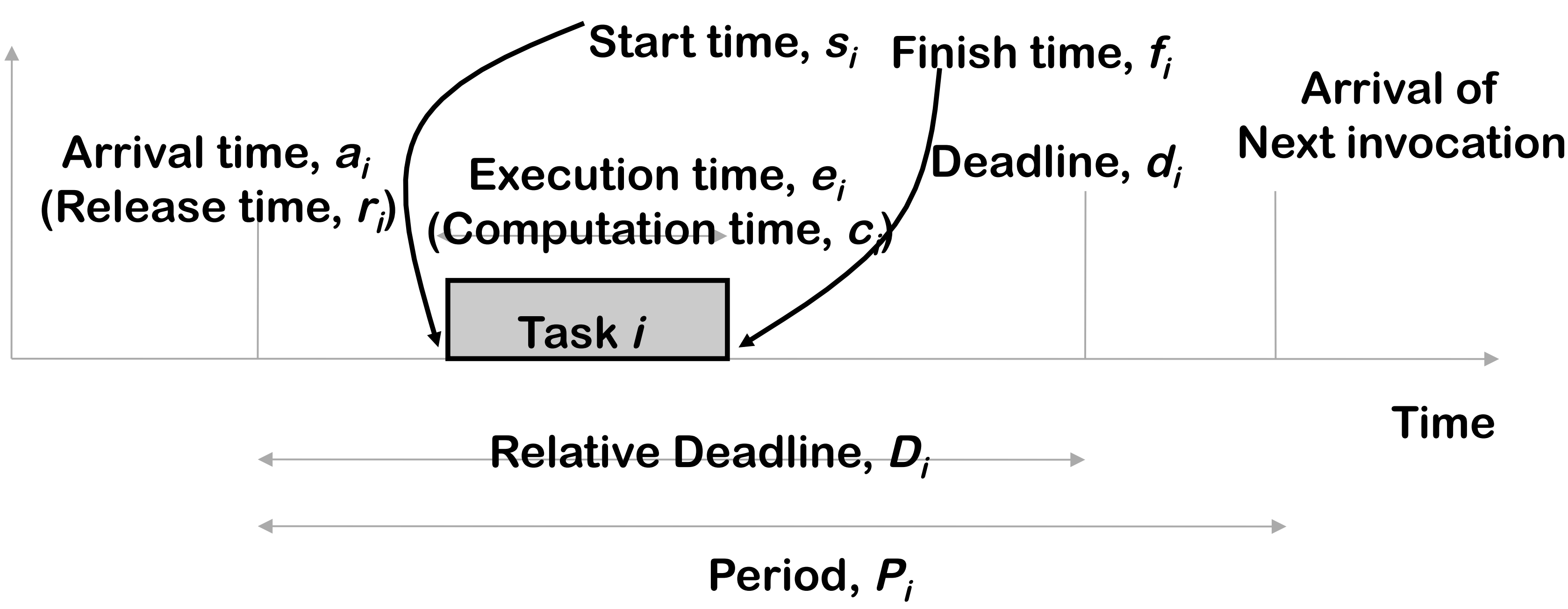
Some terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



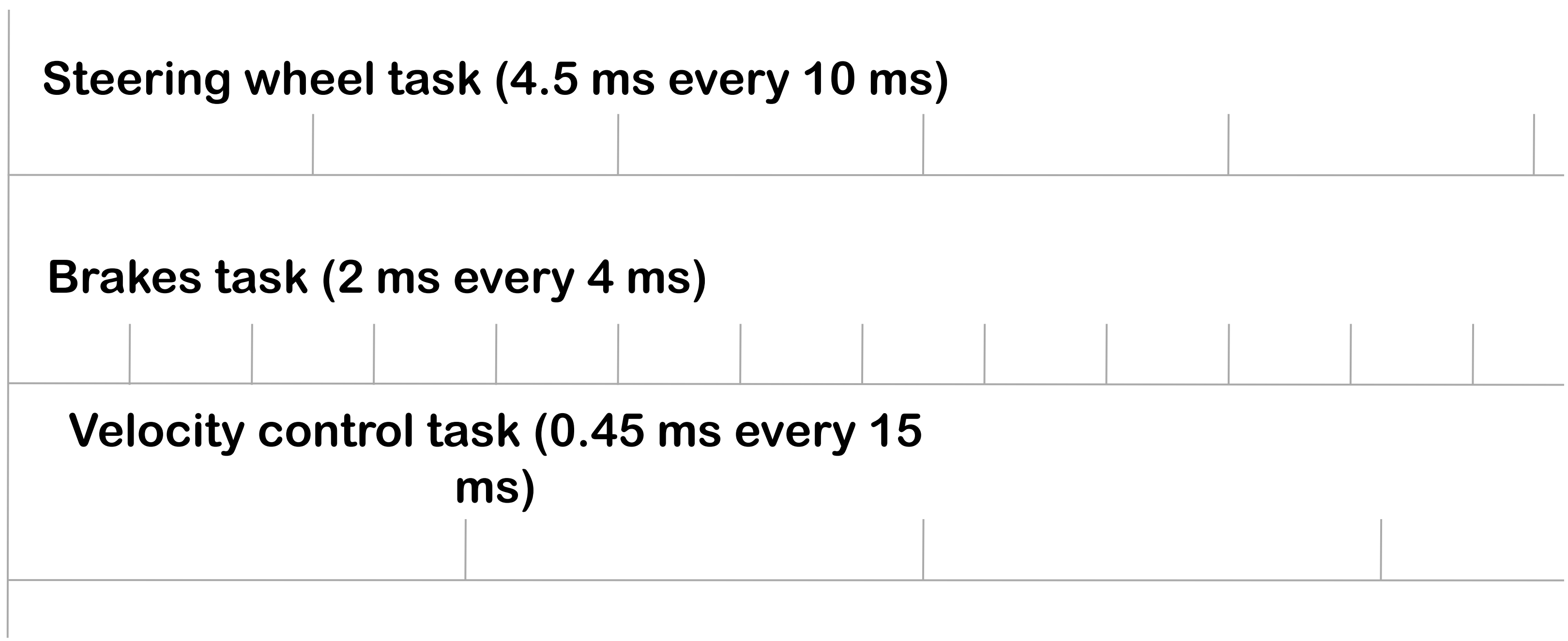
Some terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



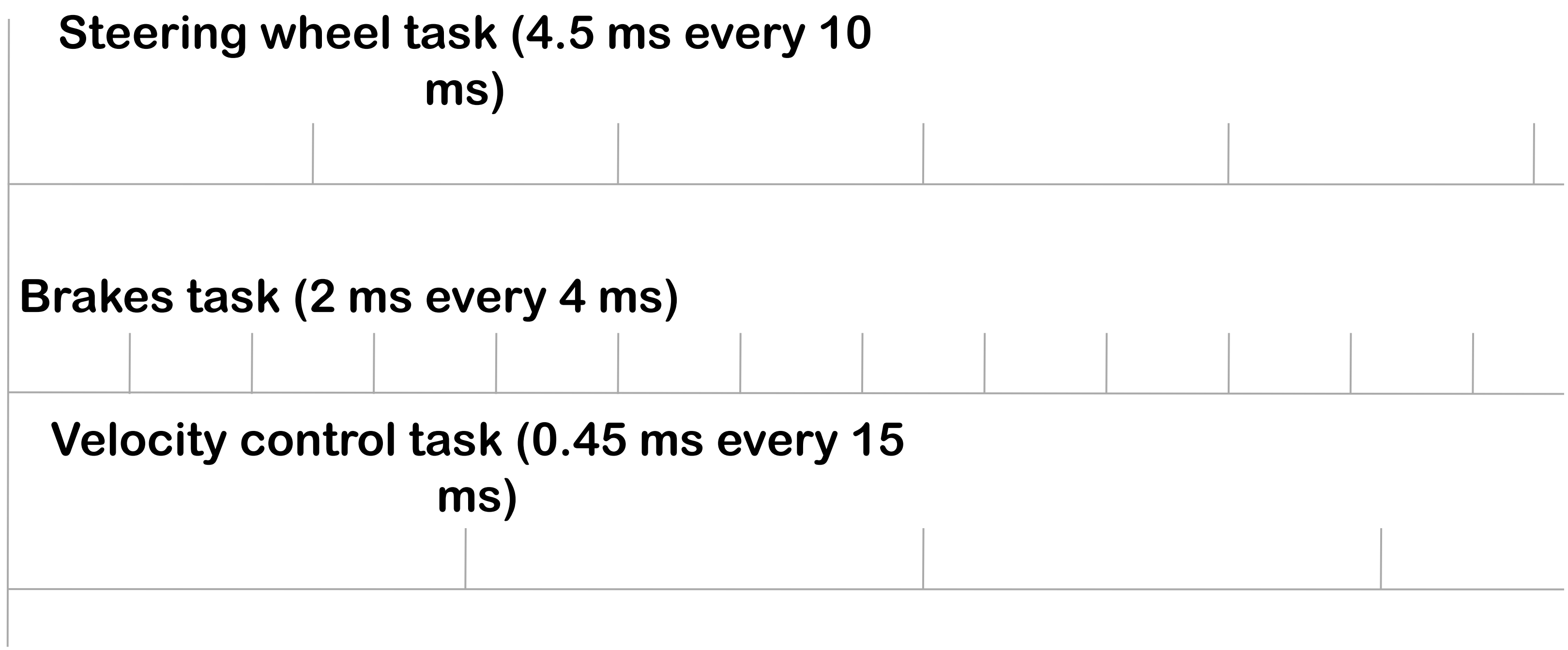
Back to the Drive-by-Wire example

- Find a schedule that makes sure all task invocations meet their deadlines
- Often, relative deadlines are equal to the period lengths



Back to the Drive-by-Wire example

- **Sanity check #1: Is the processor over-utilized? (e.g., if you have 5 assignments due this time tomorrow and each takes 6 hours, then 5x6 = 30 > 24 -> you are overutilized)**
  - **Hint: Check if processor utilization > 100%**



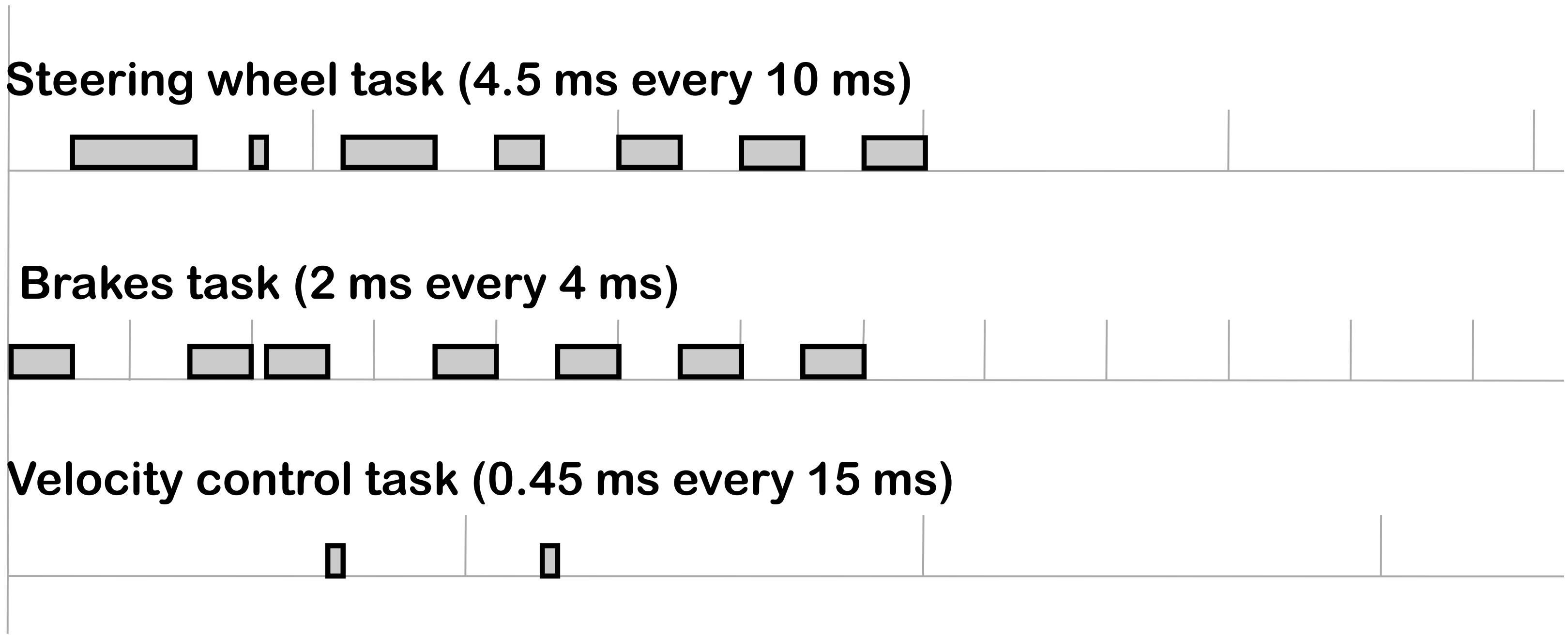
Utilization of a task set

- **For a set of tasks $\{T_i\}$ with execution times $\{e_i\}$ and periods $\{P_i\}$, the utilization, U, is the fraction of time, in the long run, for which the task set will use the system.**

$$U := \sum_i \frac{e_i}{P_i}$$

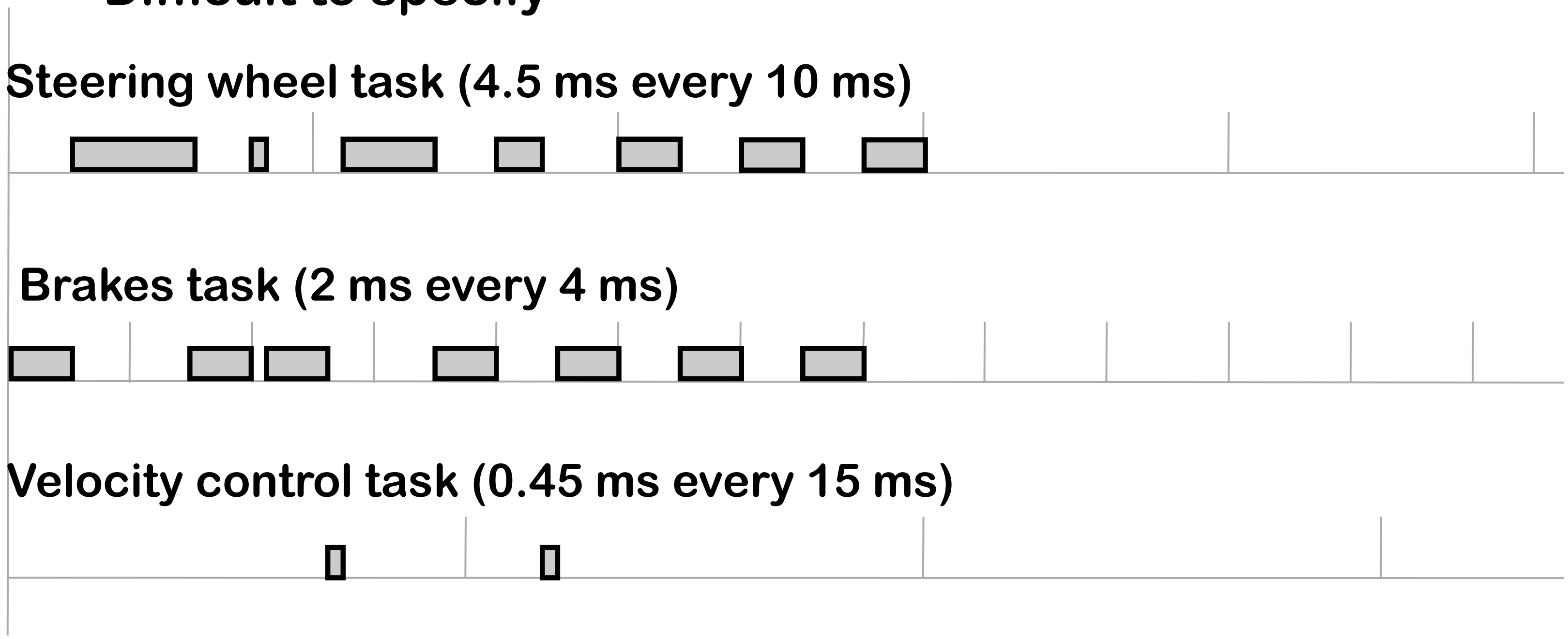
Task scheduling

- In what order should tasks be executed?
  - Hand-crafted schedule (fill timeline by hand)
  - Cyclic executive scheduling



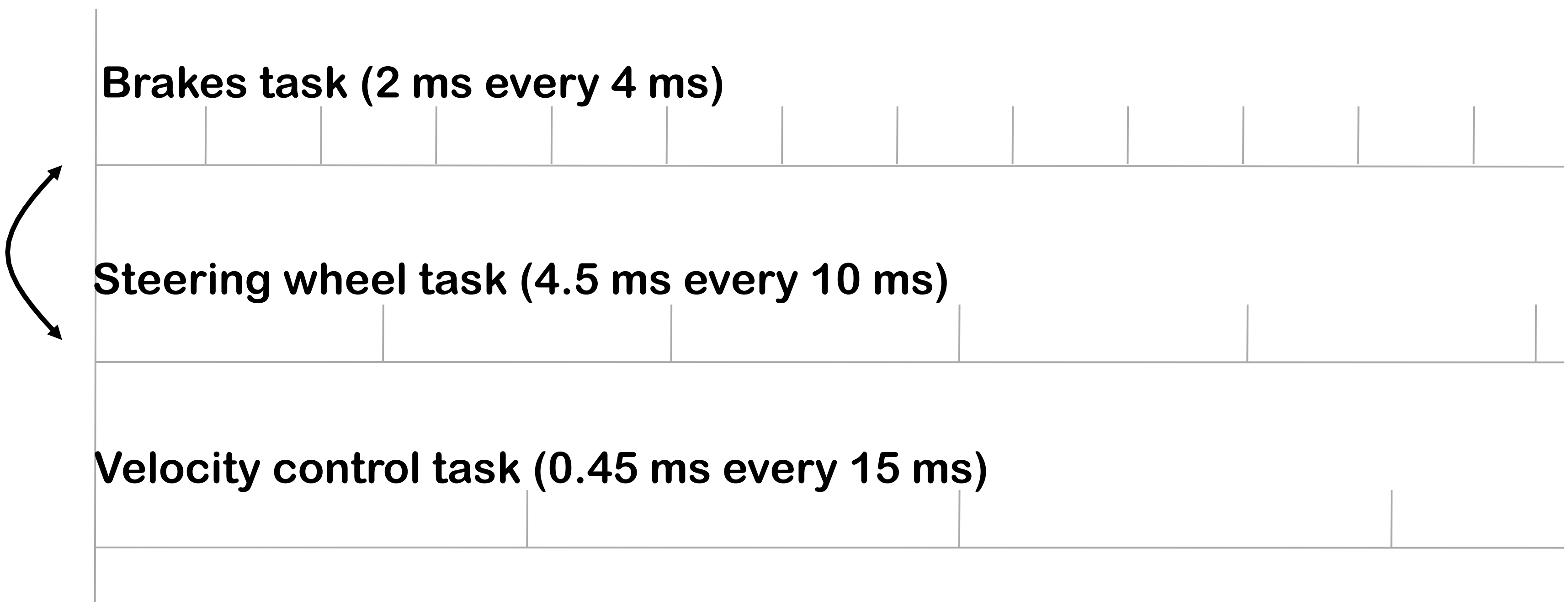
Task scheduling

- Cyclic executive scheduling
  - Why is it called a “cyclic” executive?
  - What are the problems with cyclic executive scheduling?
    - Hard to adjust the schedule if tasks change
    - Difficult to specify



Task scheduling

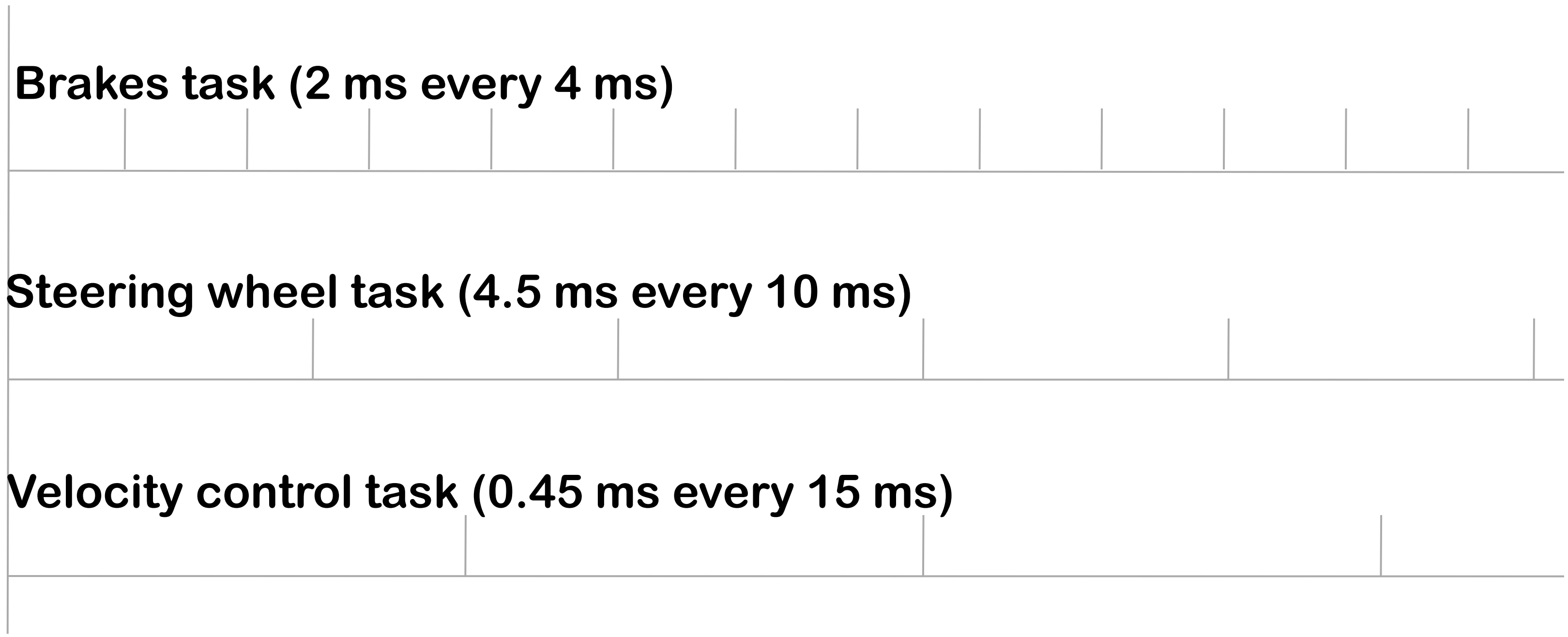
- In what order should tasks be executed?
  - Cyclic executive scheduling or
  - Priority based schedule (assign priorities; schedule is implied)



Intuition: Urgent tasks should be higher in priority

Task scheduling

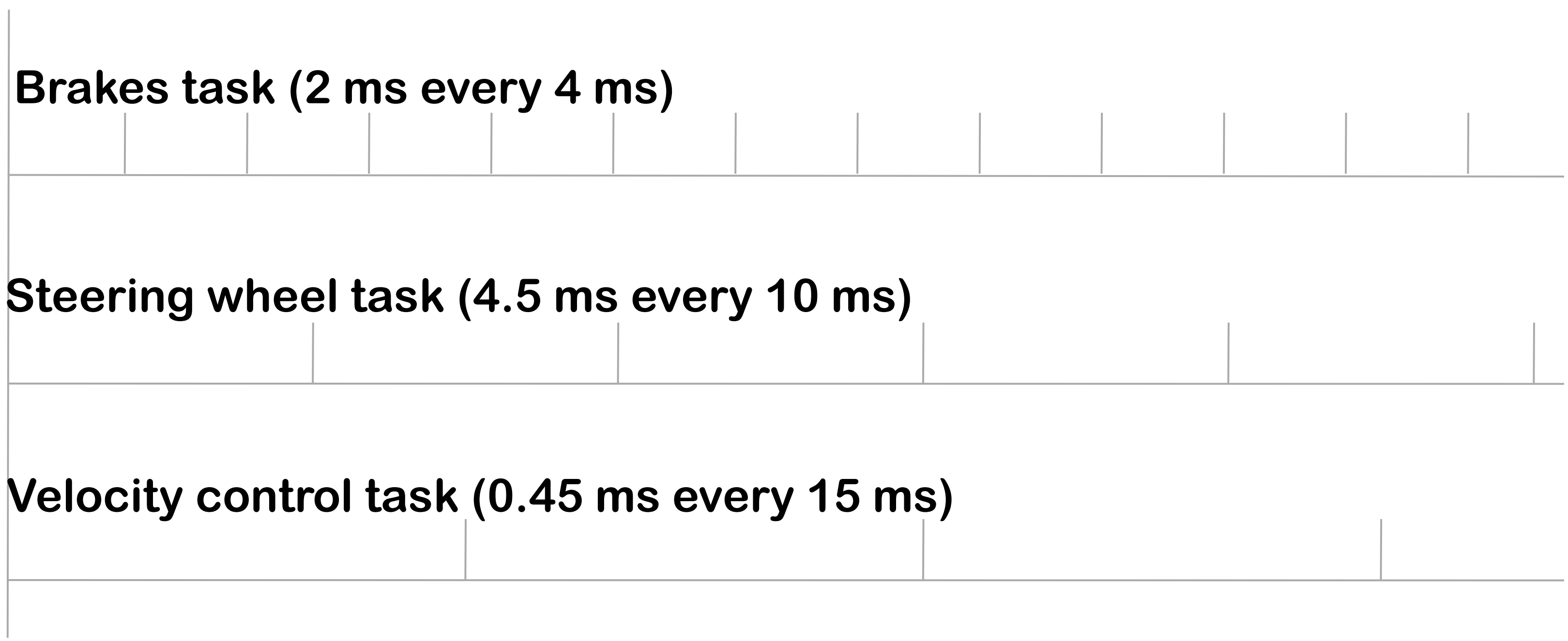
- Preemptive versus non-preemptive?
  - Preemptive: Higher-priority tasks can interrupt lower-priority ones
  - Non-preemptive: They can’t



In this example, will non-preemptive scheduling work?

Task scheduling

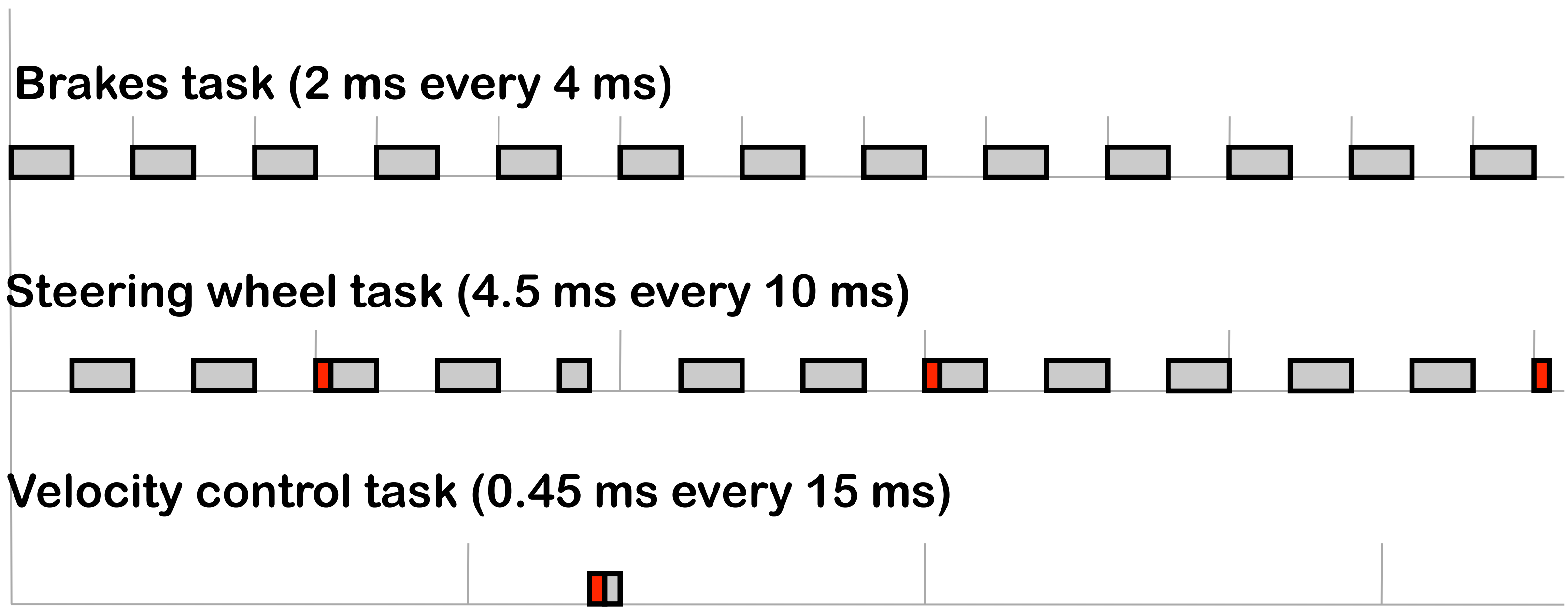
- Preemptive versus non-preemptive
  - Preemptive: Higher-priority tasks can interrupt lower-priority ones
  - Non-preemptive: They can't



In this example, will non-preemptive scheduling work?
Hint: Compare relative deadlines of tasks to execution times of others

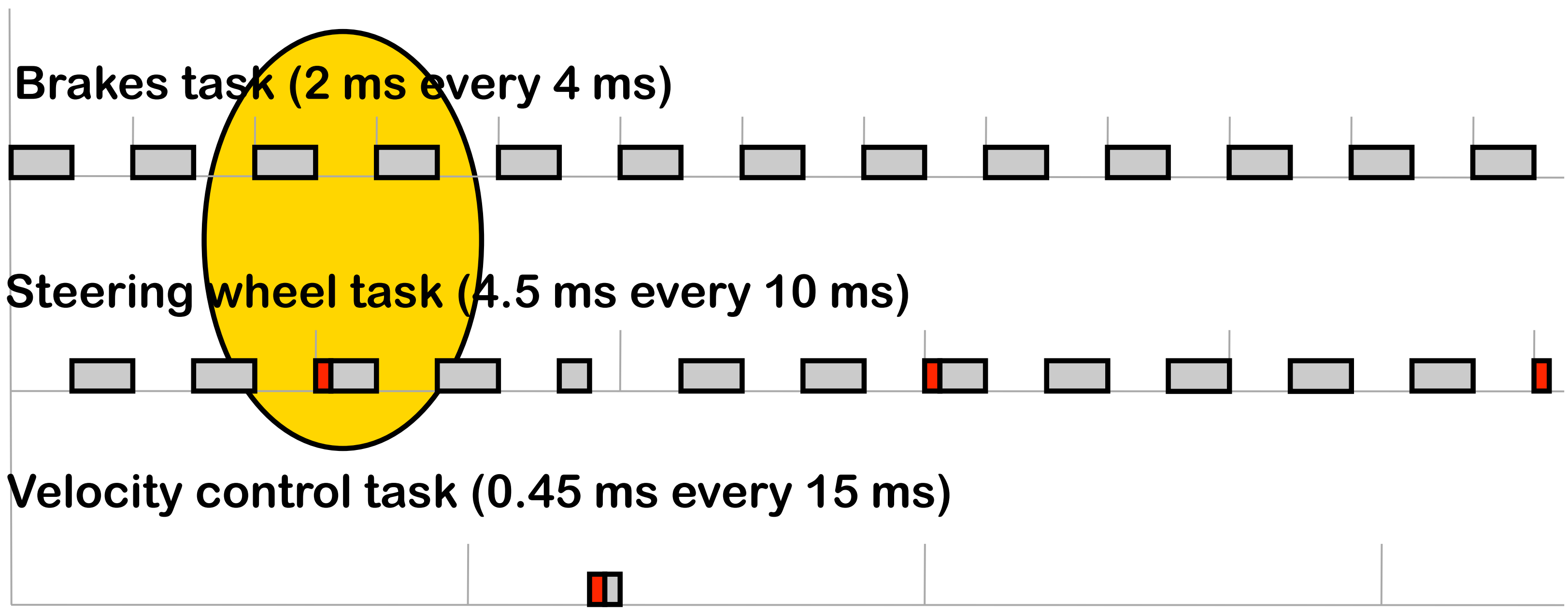
Timeline

- **Even with preemption, deadlines are missed!**
- **Average utilization < 100%**



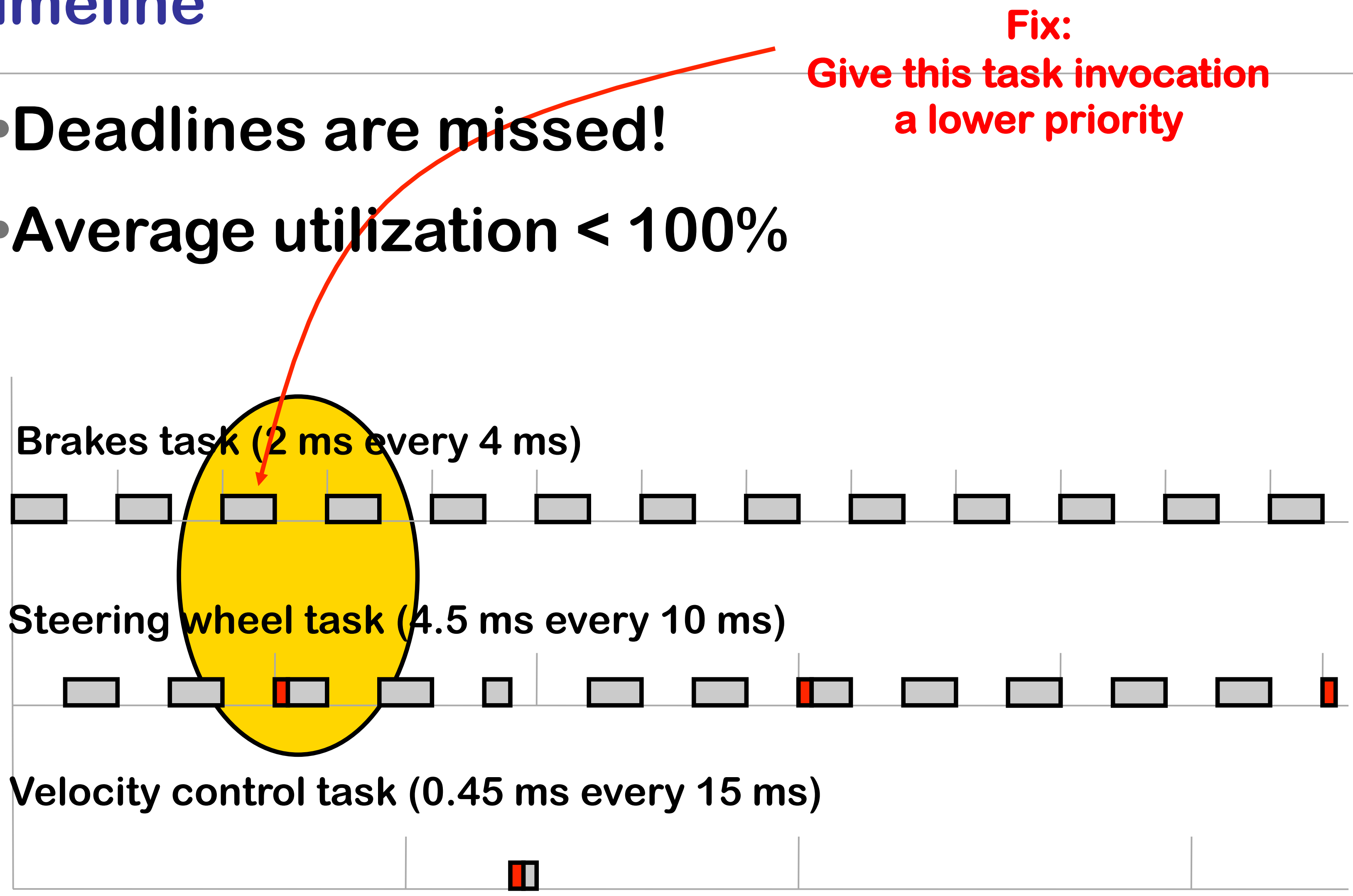
Timeline

- Deadlines are missed!
- Average utilization < 100%



Timeline

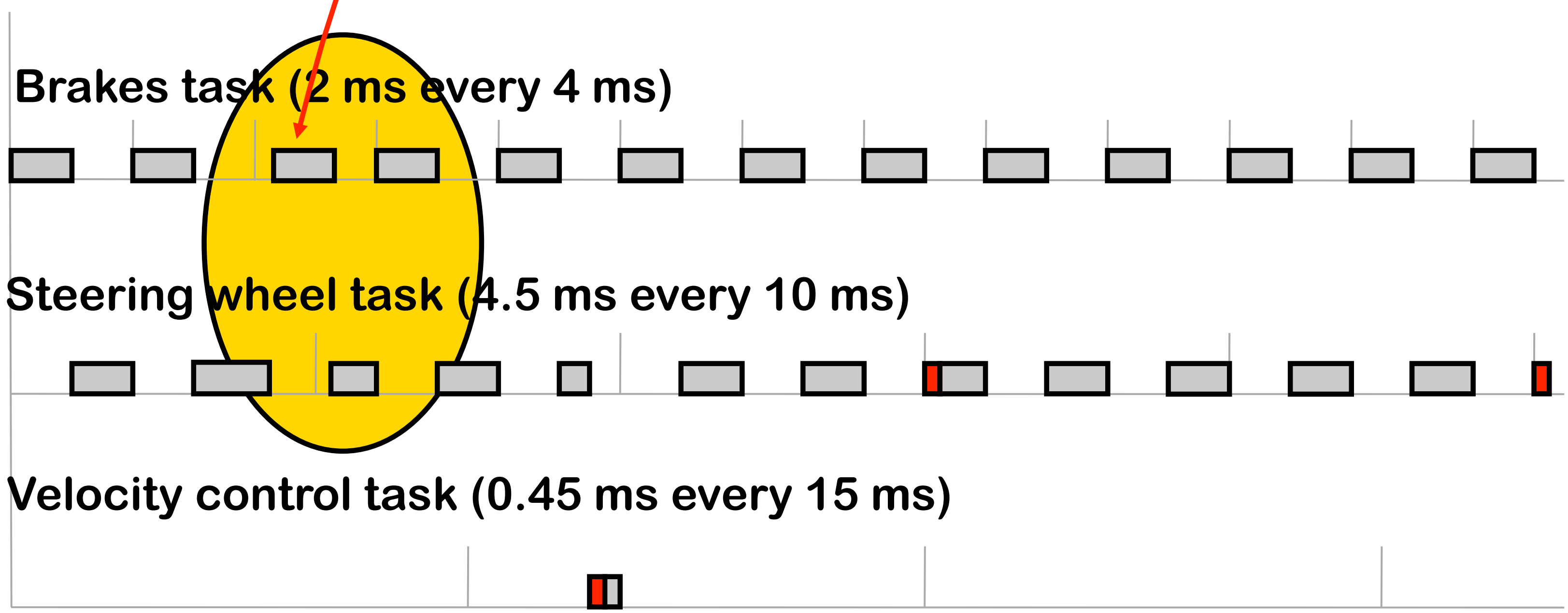
- Deadlines are missed!
- Average utilization < 100%



Timeline

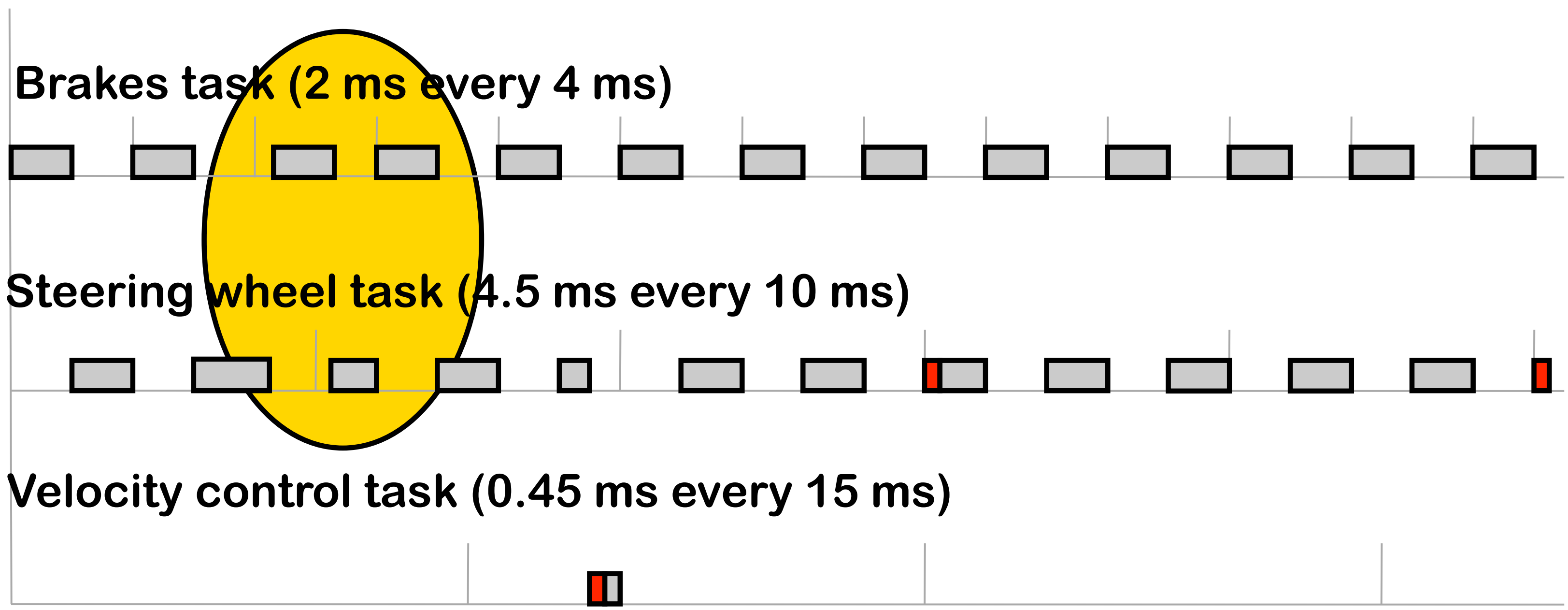
- Deadlines are missed!
- Average utilization < 100%

Fix:
Give this task invocation a lower priority



Task scheduling

- Static versus Dynamic priorities?
  - Static: All jobs (instances) of the same task have the same priority
  - Dynamic: Jobs (instances) of same task may have different priorities



Intuition: Dynamic priorities offer the designer more flexibility and hence are more capable to meet deadlines

Examples of policies

- Static priority policies
  - **Rate monotonic priority**: tasks with shorter periods get higher priority
  - **Deadline monotonic priority**: tasks with shorter deadlines get higher priority
  - Rate monotonic priorities and deadline monotonic priorities are identical if relative deadlines are equal to the periods
  - **Shortest job first policy**
- Dynamic priority policies
  - **Earliest deadline first**: jobs with the earliest absolute deadline take highest priority
  - **First In, First Out**: jobs with earliest arrival time take highest priority

Interesting questions

- What is the optimal dynamic priority scheduling policy? (Optimal: meets all deadlines as long as any other policy in its class can)
  - Can it meet all deadlines as long as the processor is not over-utilized? [U <= 1]
- What is the optimal static priority scheduling policy?
  - When can it meet all deadlines?
  - Can it meet all deadline as long as the processor is not over-utilized?

Interesting questions

- What is the optimal dynamic priority scheduling policy? (Optimal: meets all deadlines as long as any other policy in its class can)
  - Can it meet all deadlines as long as the processor is not over-utilized? [U <= 1]
- What is the optimal static priority scheduling policy?
  - When can it meet all deadlines?
  - Can it meet all deadline as long as the …sor is not over-utilized?

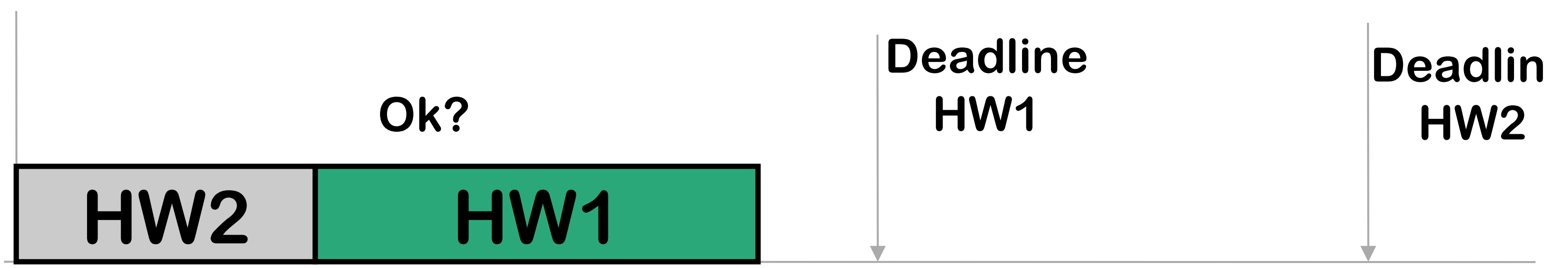
Utilization Bounds

Utilization bounds for schedulability

- U^* is called a utilization bound for a given scheduling policy S if and only if all task sets with utilization less than or equal to U^* can be scheduled using the policy S and there exists at least one task set with utilization (U^* +epsilon) that cannot be scheduled using policy S.
- Of course, the maximum value that U^* can attain is 1.

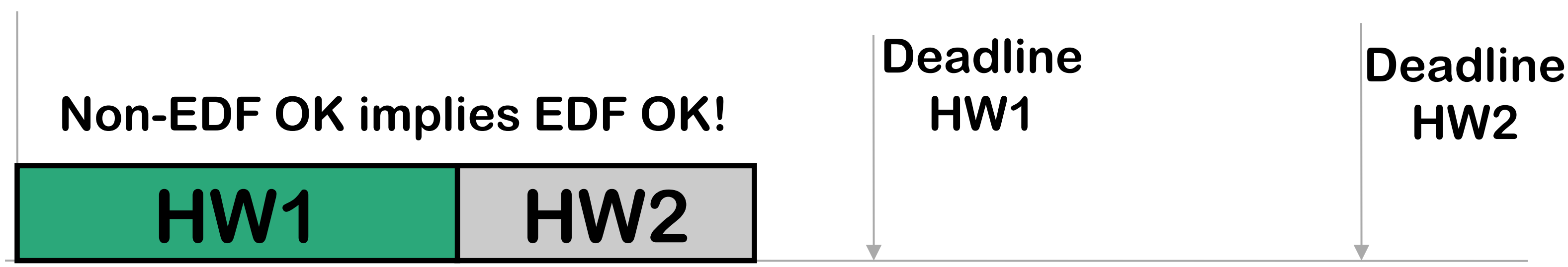
Optimality result for EDF scheduling

- EDF is the optimal dynamic priority scheduling policy
  - Priorities correspond to absolute deadlines
  - It can meet all deadlines whenever the processor utilization is less than 100%
  - Intuition:
    - You have HW1 due tomorrow and HW2 due the day after, which one do you do first?
    - If you started with HW2 and met both deadlines you could have started with HW1 (in EDF order) and still met both deadlines
    - EDF can meet deadlines whenever anyone else can



Optimality result for EDF scheduling

- EDF is the optimal dynamic priority scheduling policy
  - It can meet all deadlines whenever the processor utilization is less than 100%
  - Intuition:
    - You have HW1 due tomorrow and HW2 due the day after, which one do you do first?
    - If you started with HW2 and met both deadlines you could have started with HW1 (in EDF order) and still met both deadlines
    - EDF can meet deadlines whenever anyone else can



Why study static-priority policies?

- EDF is the optimal dynamic scheduling policy and has a utilization bound of 1.
    - The utilization bound is 1 (or 100%) when tasks have periods equal to their relative deadlines.
- EDF, however, is hard to implement in most systems.
    - Complexity is high.
    - Job queues need to be reordered often (high overhead!).
    - Most hardware subsystems allow only static priorities.

What you should know

- Definitions
  - *Tasks*
  - *Task invocations*
  - *Release/arrival time,*
  - *Absolute deadline and relative deadline*
  - *Period*
  - *Start time and finish time*
- Preemptive versus non-preemptive scheduling
- Priority-based scheduling
- Static versus dynamic priorities
- Utilization and Schedulability
- Optimality of EDF

Another example

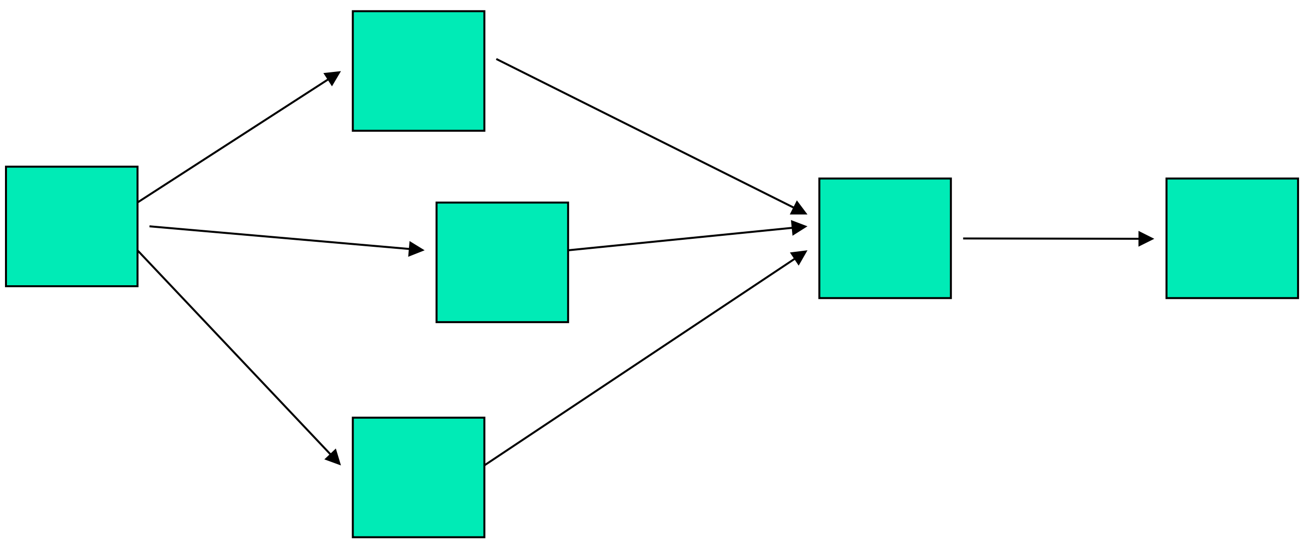
- You are the system administrator of Ameritrade.com (online stock trading site)
- You offer the following guarantee to your premium customers:
  - Stock trades of less than a $100,000 value go through in 8 seconds or you charge no commission.
  - Stock trades of more than a $100,000 value go through in 3 seconds or you charge no commission.
- Non-premium customers do not enjoy these guarantees
- Your job is to ensure that the premium customers are always served within their agreed-upon maximum latencies. What needs to be done?

For later: Aperiodic tasks

- Periodic tasks vs. aperiodic tasks
  - Aperiodic tasks may arrive at any time
  - Periodic tasks arrive at regular intervals [strictly P_i]
- Sporadic tasks
  - Successive arrivals have a minimum separation distance [greater than or equal to P_i].
- How does the lack of periodicity affect scheduling, and schedulability analysis?

For later: Precedence constraints

- In the discussion thus far, we focused on tasks that have no dependencies.
- What if tasks have precedence constraints?
  - Tasks can execute only if their predecessors have finished execution.



For later: Resource constraints

- In addition to the CPU, tasks may need resources
  - Memory
  - Disk
  - Shared data structures
- Types of resources
  - Space multiplexed: An example is the memory system. Different tasks may use different portions of the resource.
  - Time multiplexed: Only one task can access the resource at a time. An example is a data structure protected by a lock.
- How do resource constraints affect scheduling?