EL2450 Hybrid and Embedded Control

Lecture 6: Event-based control and real-time systems

- Event-based control
- Real-time systems

Today's Goal

You should be able to

- Design appropriate sampling times for event-based control
- Describe the main features of a real-time operating system
- Formulate the real-time scheduling problem

Event-based control

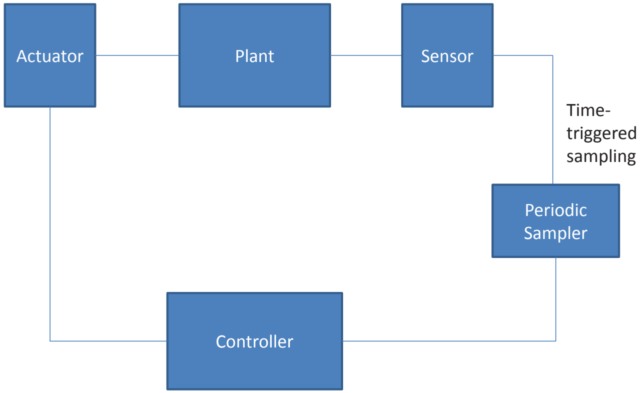
- Periodic, time-triggered sampling at pre-specified instants: does not take into account optimal resource usage
- Sampling time choise: heuristics, rules of thumb
- Unsettling not to formally derive sampling time
- A strategy considering better resource usage: event-triggered control
- Aperiodic sampling based on event-based rule
- Improves energy usage and processor time usability

Stability LTI systems with errors

- Consider $\dot{x} = Ax + Bu$. Assume that $u = Kx$ is a stabilizing controller.
- Now assume measurement errors in the feedback, so that control input is $u = K(x + e)$.
- **Fact**: There exists a quadratic Lyapunov function V and $a, b > 0$ for which

$$\dot{V} \le -a||x||^2 + b||x||||e||$$

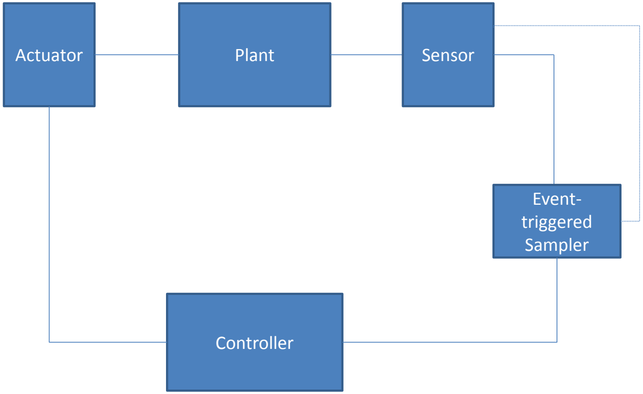
Periodic sampled control



Periodic sampled control

- Plant: $\dot{x} = Ax + Bu$.
- Assume that $u = Kx$ is a stabilizing controller without sampling.
- Controller with periodic sampling and ZOH: $u(t) = Kx(kh), t \in [kh, kh + h)$.

Event-based control



Event-based control

- Event-based control takes into account state or output feedback in order to sample as less as possible in an aperiodic fashion.
- Controller with aperiodic sampling and ZOH: $u(t) = Kx(t_k), t \in [t_k, t_{k+1})$.
- Question: how can we choose the sequence $\{t_k\}, k = 0, \ldots$ so that the stability properties are maintained and the sampling is as rare as possible?
- Answer: this is held by using *feedback* information from the plant in the sampling process!

State feedback stability with event-based control

State error: $e(t) = x(t_k) - x(t), t \in [t_k, t_{k+1})$. The control law can be written as
$u(t) = Kx(t_k) = K(x(t) - x(t) + x(t_k)) = K(x(t) + e(t))$.
The closed loop system can be written as
$\dot{x}(t) = (A + BK)x(t) + BKe(t)$.
Fact: since $A + BK$ is stable, there exists a quadratic Lyapunov function V for which

$$\dot{V} \leq -a||x||^2 + b||x||||e||$$

State feedback stability with event-based control

Idea: the choice of t_k affects the value e. Maintain $\dot{V}$ negative by choosing e appropriately. If $||e(t)|| \leq \sigma ||x(t)||$ and $-a + b\sigma = -c < 0$, then $\dot{V} \leq -c||x||^2$ and asymptotic stability is maintained.
Event-triggering rule: If the last sample was at t_k, the next sample is taken at

$$t_{k+1} = \inf_t \{t > t_k | ||e(t)|| = \sigma ||x(t)||\}$$

State feedback stability with event-based control

Theorem The event-based rule

$$t_{k+1} = \inf_t \{t > t_k | ||e(t)|| = \sigma ||x(t)||\}$$

maintains the asymptotic stability of the original system for appropriate choice of $\sigma > 0$. Furthermore, the inter-sampling times $t_{k+1} - t_k$ are lower bounded by a positive constant τ^* for all k,$\sigma > 0$.

Inter-sampling times lower bound

Proof sketch After an event at t_k we have $||e(t_k)||/||x(t_k)|| = 0$. Next event happens when $||e(t)||/||x(t)|| = \sigma$. Time it takes for $||e||/||x||$ to go from zero to σ is lower bound by a strictly positive constant τ^*. This is due to that solutions of differential inequality

$$\frac{d}{dt}\frac{||e||^2}{||x||^2} \leq 2||A+BK||\frac{||e||}{||x||}+2(||BK||+||A+BK||)\frac{||e||^2}{||x||^2}+2||BK||\frac{||e||^3}{||x||^3}$$

need a minimum τ^* to go from 0 to σ^2.

Time delays

In case of a (small enough) fixed delay δ, we can compute a more conservative $0 < \sigma' < \sigma$ to compensate for the delay. Control law implemented with a delay δ:

$$u(t) = Kx(t_k), t \in [t_k + \delta, t_{k+1} + \delta)$$

Nonlinear systems

- Plant: $\dot{x} = f(x, u)$.
- Assuming that $u = k(x)$ is a stabilizing controller may not be enough. Need *assumption* on properties of controller with error measurements.
- Controller with aperiodic sampling and ZOH: $u(t) = k(x(t_k)), t \in [t_k, t_{k+1})$.
- Closed loop system is $\dot{x} = f(x, k(x + e))$.
- Existence of V similar to Fact of slide 4 not a given for nonlinear case.

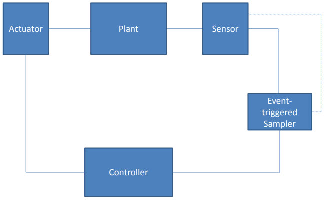
Nonlinear systems

- Assume V and strictly increasing functions $a(||x||), \gamma(||e||), a(0) = 0, \gamma(0) = 0$ such that

$$\frac{\partial V}{\partial x} f(x, k(x+e)) \leq -a(||x||) + \gamma(||e||)$$

- Then the trigger rule $\gamma(||e||) \leq \sigma a(||x||)$, $0 < \sigma < 1$ maintains the stability properties of the nominal closed-loop system (verify).
- $t_{k+1} = \inf_t \{t > t_k | \gamma(||e(t)||) = \sigma a(||x(t)||)\}$
- Robustness properties of closed-loop system need to be explored and not a given as in linear case.

Event-based control



- How can we avoid continuous monitoring of state or output feedback?
- Idea: use prediction of the state evolution. This can be done for linear systems (self-triggered control).
- Another idea: use periodic samples of state and implement control in an event-based fashion (periodic event-based control).

Periodic event-based control

- Plant: $\dot{x} = Ax + Bu$.
- Sample uniformly with period h, yealding

$$x(kh + h) = \Phi x(kh) + \Gamma u(kh)$$

where

$$\Phi = e^{Ah}, \quad \Gamma = \int_0^h e^{As} ds B$$

- h can be chosen as the minimum implementable period of the digital platform, since the interest is in less frequent controller updates.

Periodic event-based control

- Assume $u(kh) = Lx(kh)$ is a stabilizing controller.
- Then, there exists P such that

$$(\Phi + \Gamma L)^T P(\Phi + \Gamma L) - P = -Q$$

  with P, Q positive definite.
- Idea: implement sampled controller in an event-based fashion. Controller given by

$$u(kh) = Lx(k_i h), k \in [k_i, k_{i+1}).$$

- Event sequence is now $\{k_0 h, k_1 h, \ldots\} \subset \{0, h, 2h, \ldots\}$.

State feedback stability with periodic event-based control

- State error:

$$e(kh) = x(k_i h) - x(kh), k \in [k_i, k_{i+1})$$

  The control law can be written as $u(kh) = L(x(kh) + e(kh))$.

- Then $x(kh + h) = (\Phi + \Gamma L)x(kh) + \Gamma Le(kh)$.
- One can now show that $V = x^T Px$ satisfies

$$V(x(kh + h)) - V(x(kh)) \leq -a||x(kh)||^2 + b||e(kh)||^2$$

  for some $a, b > 0$.

State feedback stability with periodic event-based control

If $||e(kh)|| \leq \sqrt{\sigma}||x(kh)||$ and $-a + b\sigma = -c < 0$, then

$$V(x(kh+h)) - V(x(kh)) \leq -c||x(kh)||^2$$

and asymptotic stability is maintained.

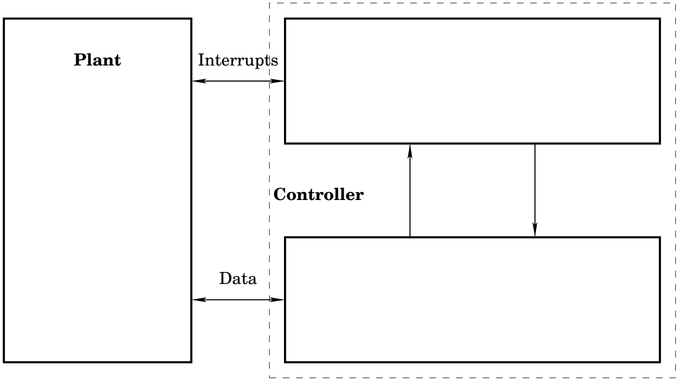
- This yields a triggering rule similarly to continuous-time case: next trigger time $k_{i+1}h$ chosen based on first next violation of $||e(kh)|| \leq \sqrt{\sigma}||x(kh)||$.
- Avoids continuous state measurements.

Comments on event-based control

- Choice of σ represents tradeoff between sampling frequency and decay rate of V.
- Minimum inter-sampling time τ^* can be used for periodic implementations.
- Delays and other dynamics can be considered.
- Trend: distributed event-based control.

Event-Triggered Control System

Need real-time operating system to handle multiple events and concurrency



What is a Real-Time Computer System?

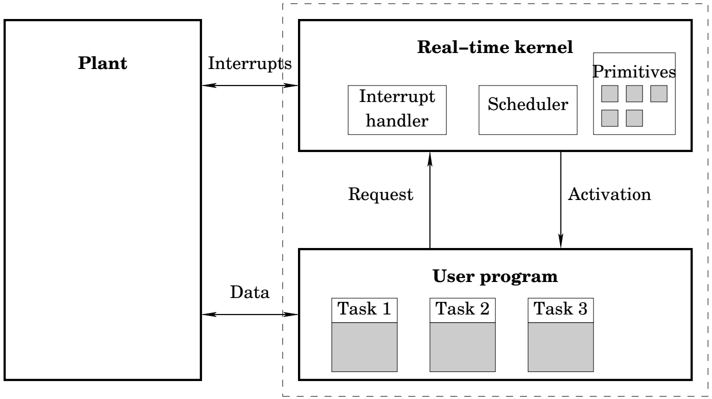
Real-time systems are defined as those computer systems for which correctness depends not only on the logical properties of the computations, but also on the time at which the results are computed.

Properties of Real-Time Systems

A real-time system is a computer system with

- Timing requirements (hard or soft)
- Time-triggered and event-triggered actions
- Concurrent activities
- Digital and analogue signals

Real-Time Operating System



Hard and Soft Real-Time Systems

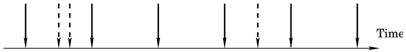
- A **hard real-time system** is a system where it is absolutely necessary that the responses occur within the required deadline
- A **soft real-time system** is a system where deadlines are important but the system still functions if deadlines are occasionally missed

- Safety critical systems are often hard real-time systems
- Most hard real-time systems are control systems
- Most real-time control systems are not hard
- Real-time does not mean high-speed computations

Events and Tasks

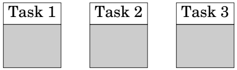
Real-time systems must respond to events:

- Periodic and aperiodic events
- External and internal events



A **task** is the work that has to be done to service an event.

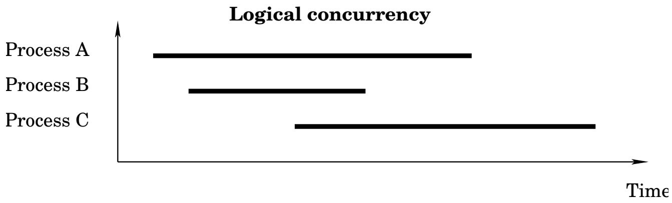
- A control task is an example of a task
- Tasks are often handled independently during design

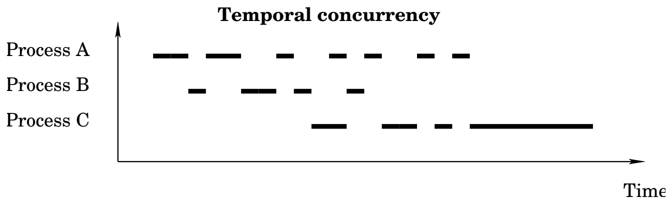


Concurrency

- Real-time systems are concurrent
- Events may occur at the same time
- Processes multiplex their execution on a single CPU
- Lead to concurrent programming
- Requires scheduling and interrupts

Logical and Temporal Concurrency

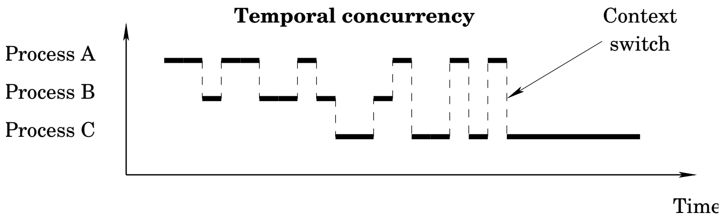




Context Switch

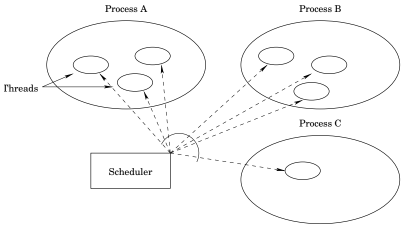
A **context switch** occurs when the system changes the running process

The context of the previously running process is stored and the context of the new process is restored



Processes and Threads

- A **process** is an executed program with its own allocated memory
- A process consists of **threads**, which implement the computations
- Threads are the basic unit of work handled by the scheduler



Example `PID` can be a process executing a PID controller, while `AD_conversion` and `updateState` can be threads inside `PID`.

Scheduling

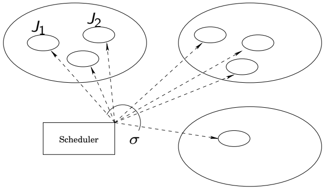
- **Clock-driven scheduling**
  - Decides on which process to execute based on the *time*
  - Schedule is computed off-line and stored for use at run time
  - Appropriate for hard real-time systems
- **Priority-driven scheduling**
  - Decides on which process to execute based on the process *priority*
  - Scheduling is an on-line activity, often hard to predict
  - CPU idle only when no processes need execution
  - Many scheduling strategies can be implemented through priorities

Here we mainly discuss priority-driven scheduling.

Schedule

For a set of tasks $J = \{J_1, \ldots, J_n\}$, a **schedule** is a map $\sigma : \mathbb{R}^+ \mapsto \{0, 1, \ldots, n\}$ assigning a task at each time instant t:

$$\sigma(t) = \begin{cases} k \neq 0, & \text{CPU should execute } J_k \\ 0, & \text{CPU is idle} \end{cases}$$

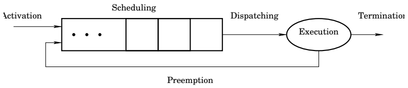


- σ is **feasible** if J can be completed according to specified constraints
- J is **schedulable** if there exists a feasible σ

Scheduling Algorithms

A **scheduling algorithm** sets task execution order (defines σ)
A scheduling algorithm is

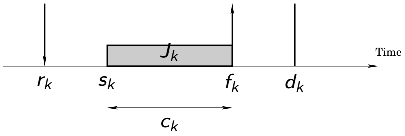
- **preemptive** if the running task can be arbitrarily suspended at any time (otherwise **non-preemptive**)
- **static** if scheduling decisions are based on fixed parameters assigned prior to activation (otherwise **dynamic**)
- **off-line** if σ is generated off-line and stored in a table (otherwise **on-line**)



Timing Constraints

A task J_k can be characterized by the following parameters:

- **Release time** r_k is the time at which J_k becomes ready for execution
- **Computation time** c_k is the time necessary for the CPU to execute J_k without interruption
- **Deadline** d_k is the time before which J_k should be completed
- **Start time** s_k is the (actual) time at which J_k starts executing
- **Finishing time** f_k is the (actual) time at which J_k finishes executing



Independent Periodic Tasks

We mainly focus on scheduling independent periodic tasks.
Tasks are

- **independent** if there are no precedence relations and no resource constraints
- **periodic** if they are activated at a constant rate

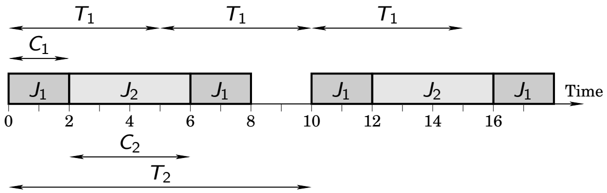
Periodic Tasks

Each periodic task J_k is characterized by its

- **Period** T_k
- **Worst-case computation time** C_k
- **Relative deadline** D_k (deadline relative to current release time)

Scheduled tasks have

- **Worst-case response time** R_k (largest time between release and termination)
- **Phase** ϕ_k (release time of the first task instance)



Schedule Length and Feasibility

The length of a schedule σ is equal to

$$\mathrm{lcm}(T_1, \ldots, T_n)$$

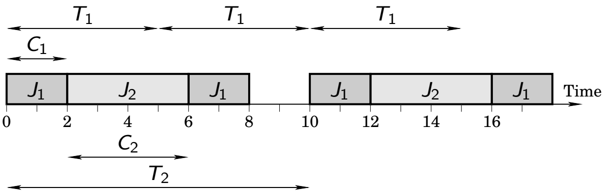
which is the least common multiplier of the task periods.

σ is feasible if all deadlines are met, i.e.,

$$R_k \leq D_k, \qquad \forall J_k \in J$$

Example

J_1 has $T_1 = 5$ and $C_1 = 2$
J_2 has $T_2 = 10$ and $C_2 = 4$
Deadlines $D_1 = T_1 = 5$ and $D_2 = T_2 = 10$ are met with schedule below, since $R_1 = 3 < D_1$ and $R_2 = 6 < D_2$.
Schedule length is $\text{lcm}(T_1, T_2) = \text{lcm}(5, 10) = 10$



Utilization Factor

The **utilization factor** U of a periodic task set J is the fraction of processor time spent in the execution of the task set.
Since C_i/T_i is the fraction for J_i, we have

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Example The previous task set gives

$$U = \frac{2}{5} + \frac{4}{10} = 0.8$$

Hence, CPU used at most 80% of the time.

Properties of Utilization Factor

It is desirable to find a scheduling algorithm that gives a feasible schedule

- If utilization factor $U > 1$, then the task set J is not schedulable
- Even if $U \leq 1$, it might be hard to find a feasible schedule
- Scheduling algorithms have various other properties (see next lecture)

Next Lecture

Real-Time Scheduling

- Scheduling algorithms
- Schedulability analysis