Networked and Embedded Control Systems

Josep M. Fuertes, Ricard Villà, Jordi Ayza, Pau Martí, Manel Velasco, José Yépez, Camilo Lozoya, Josep Guàrdia, Frederic Pérez

Automatic Control Dept., Technical University of Catalonia
pau.marti@upc.edu
Application scenarios
Common features

- **Resource-constrained** systems (mass-marked products subject to hard economic constraints)
- Often used in **unpredictable** environments
- The **time** when results are delivered is important
- Many simultaneously running **control** applications
Puzzle

Computing of control systems

(Real-time) Computing

Resource constrained

Control

Control of computing systems
Understanding the puzzle

![Diagram showing resource manager connected to resources, which in turn are connected to control task and plant.](image-url)
Building the puzzle

■ Reality: gap between communities (real-time, control, ...)

■ Why? Misconceptions
  ◆ Real-time engineers assume hard deadlines for control algorithms
  ◆ Control engineers assume determinism in the computing platform

■ Need: Closer interaction between communities

■ Today, emerging areas closing the gap
  ◆ computing of control systems
  ◆ control of computing systems
Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions

2. Control of real-time control systems
   (a) Overview
   (b) Representative examples
Timing and implementation

Simplest mathematical model with

- constant sampling period
- instantaneous input-output latency

2nd HYCON PhD School on Hybrid Systems (2007) - Networked and Embedded Control Systems
Linear time-invariant continuous-time system state-space form

\[
\frac{dx(t)}{dt} = Ax(t) + Bu(t)
\]
\[
y(t) = Cx(t)
\]

Discrete form, with sampling period \( h \) [1]

\[
x_{k+1} = \Phi(h)x_k + \Gamma(h)u_k
\]
\[
y_k = Cx_k,
\]

where \( \Phi(t) \) and \( \Gamma(t) \) are obtained using the following

\[
\Phi(t) = e^{At}, \quad \Gamma(t) = \int_0^t e^{As}Bds,
\]
Timing of the basic model is not realistic

Adding a time delay to model an input/output latency due to the computation of the control algorithm or the insertion of a network
Continuous-time system with time delay $\tau$

\[
\frac{dx(t)}{dt} = Ax(t) + Bu(t - \tau)
\]
\[y(t) = Cx(t)\]  \hspace{1cm} (4)

Discrete form, with $\tau \leq h$

\[
x_{k+1} = \Phi(h)x_k + \Phi(h - \tau)\Gamma(\tau)u_{k-1} + \Gamma(h - \tau)u_k.
\]
\[y_k = Cx_k,\]  \hspace{1cm} (5)

where $\Phi(t)$ and $\Gamma(t)$ are also obtained using (3).
State-space form for (5), extended model:

\[
\begin{bmatrix}
x_{k+1} \\
z_{k+1}
\end{bmatrix} = \begin{bmatrix}
\Phi(h) & \Phi(h - \tau)\Gamma(\tau) \\
0 & 0
\end{bmatrix} \begin{bmatrix}
x_k \\
z_k
\end{bmatrix} + \begin{bmatrix}
\Gamma(h - \tau) \\
I
\end{bmatrix} u_k
\]

where \(z_k \in \mathbb{R}^{m \times 1}\) represent past control signals.

This notation slightly differs from conventional notation [1] to stress dependencies on \(h\) and \(\tau\).

The notation may be still misleading: \(u_k\) is applied \(\tau\) time units after \(x_k\) is taken.
Notation issues:
\( k^{th} \) operation vs. timing

Let’s obtain system (5) by looking at the dynamics from \( x_k \) to \( x_{k+1} \). Denote the system state at time \( t_{k+\tau} \) as \( x_{k+\tau} \). Then

\[
\begin{align*}
\text{From } x_k \text{ to } x_{k+\tau} & \rightarrow x_{k+\tau} = \Phi(\tau)x_k + \Gamma(\tau)u_{k-1} \\
\text{From } x_{k+\tau} \text{ to } x_{k+1} & \rightarrow x_{k+1} = \Phi(h-\tau)x_{k+\tau} + \Gamma(h-\tau)u_k \\
\text{All together } & \rightarrow x_{k+1} = \Phi(h-\tau)(\Phi(\tau)x_k + \Gamma(\tau)u_{k-1}) + \Gamma(h-\tau)u_k \\
& = \Phi(h-\tau)\Phi(\tau)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k \\
& = \Phi(h)x_k + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k \\
& = \text{model (5)}
\end{align*}
\]
For closed loop operation of (6), given

\[ u_k = -K(h, \tau) \begin{bmatrix} x_k \\ z_k \end{bmatrix} \]  \hspace{1cm} (7)

where \( K(h, \tau) \) is the state feedback gain, the system evolution is

\[ \begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \left( \begin{bmatrix} \Phi(h) & \Phi(h - \tau) \Gamma(\tau) \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \Gamma(h - \tau) & 0 \\ I & 0 \end{bmatrix} k(h, \tau) \right) \begin{bmatrix} x_k \\ z_k \end{bmatrix} = \]  \hspace{1cm} (8)

\[ = \Phi_{cl}(h, \tau) \begin{bmatrix} x_k \\ z_k \end{bmatrix} \]

**Note:** \( K \) depends on “future” parameters, \( h \) and \( \tau \)
The extended form (6) also can model networked control systems.

\[ \tau \]

Delays controller-to-actuator \( \tau_{ca} \) and sensor-to-controller \( \tau_{sc} \) can be integrated into \( \tau \) in (6).
Example. Double integrator differential equation:

\[
\frac{d^2 y}{dt^2} = u
\]  \hspace{1cm} (9)

If \( y \) and \( \dot{y} \) are \( x_1 \) and \( x_2 \), a state space form is given by

\[
\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)
\]

\[
y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)
\]  \hspace{1cm} (10)

Discrete-time model, with period \( h \) and delay \( \tau \)

\[
x_{k+1} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \tau \left(h - \frac{\tau}{2}\right) \\ \frac{(h-\tau)^2}{2} \end{bmatrix} u_{k-1} + \begin{bmatrix} \frac{(h-\tau)^2}{2} \\ \tau \end{bmatrix} u_k
\]  \hspace{1cm} (11)
If $h = 0.1s$ and $\tau = 0.01s$, the state space form is given by

$$x_{k+1} = \begin{bmatrix} 1 & 0.1 & 0.001 \\ 0 & 1 & 0.01 \\ 0 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} 0.004 \\ 0.09 \\ 1 \end{bmatrix} u_k$$

(12)

Closing the loop with $u_k = -\begin{bmatrix} 271.7 & 21.86 & 0.23 \end{bmatrix} x_k$, the closed loop poles are $\lambda_1 = -0.3$, $\lambda_2 = -0.1$, $\lambda_3 = -0.9$

- With a faster micro $\tau = 0.005$, poles go at $\lambda_1 = 0.0082 - 0.2850i$, $\lambda_2 = 0.0082 + 0.2850i$, $\lambda_3 = -1.5513$

- With a slower micro $\tau = 0.02$, $\lambda_1 = -0.4910 + 0.9516i$, $\lambda_2 = -0.4910 - 0.9516i$, $\lambda_3 = 0.1316$, with $|\lambda_1| = |\lambda_2| = 1.0708$
Timing and implementation

Random delay (with $\tau \in [0.005 \ 0.015]$, where $\tau_d = 0.01$)

Random period (with $h \in [0.05 \ 0.15]$, where $h_d = 0.1$)
Timing and implementation

- Timing is a key aspect !!!!
- The extended model (5) permits to model timing aspects.
- And the implementation should also enforce the timing.
Let’s implement a controller: an infinite loop with a periodic activity to be executed every sampling period $h$

Periodic activity:

$$y_k = x_k$$

$$u_k = -Lx_k;$$

$$\text{write}_\text{output}(u_k);$$

Which is the right code?
Let’s implement a controller: an infinite loop with an algorithm to be executed every sampling period $h$. **First attempt:**

```plaintext
loop
    PeriodicActivity;
    WaitTime(h);
end loop
```

The computation time of *PeriodicActivity* is not accounted for.
Let’s implement a controller: an infinite loop with an algorithm to be executed every sampling period \( h \). **Second attempt:**

```plaintext
loop
    Start = CurrentTime();
    PeriodicActivity;
    Stop = CurrentTime();
    C := Stop - Start;
    WaitTime(h - C);
end loop
```

An interrupt causing suspension may occur between the assignment and `WaitTime`; or overrun problem.
Methodologies for guaranteeing timeliness are required.

Real-time technology ([2] or [3]) is the candidate: *in real-time computing the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced [4]*.
Timing and implementation

Task (or message) basic parameters:

Tasks: periodic, sporadic, aperiodic
hard, soft, best effort
time or event triggered
pre-emptive, not pre-emptive
Timing and implementation

- Scheduling problem: how to assign tasks to the processor/network such that the set of (timing) constraints is met.

- Scheduling approaches:
  - **offline scheduling**: the time axis is divided in intervals of equal length (time slots), each task is statically allocated in a slot in order to meet the desired request rate, and the execution in each slot is activated by a timer.
  - **online scheduling**: each task is assigned a priority, scheduling feasibility is verified using analytical techniques, and tasks are executed on a priority-based kernel.
Timing and implementation

Example \( (D = h) \)

<table>
<thead>
<tr>
<th>task</th>
<th>( h )</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200 ms</td>
<td>75 ms</td>
</tr>
<tr>
<td>B</td>
<td>300 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>C</td>
<td>100 ms</td>
<td>25 ms</td>
</tr>
</tbody>
</table>

What happens if tasks execute less than C?
Run-time schedule with execution time of first job of A is 2
Observations on off-line scheduling

■ Disadvantages
  ◆ Keeping strict periodic execution is not possible
  ◆ Hard to build, modify or expand
  ◆ Lacks flexibility to adapt to resource availability or varying application demands

■ Advantages
  ◆ Simple implementation enforcing precise timing (no real-time operating system required)
  ◆ Low run-time overhead
Example of what strict timing may mean....

Game: Let’s try drawing a time line to execute tasks, keeping a constant distance between consecutive job start-times

```
<table>
<thead>
<tr>
<th>task</th>
<th>h</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200 ms</td>
<td>75 ms</td>
</tr>
<tr>
<td>B</td>
<td>300 ms</td>
<td>25 ms</td>
</tr>
</tbody>
</table>
```

Assume A takes longer, e.g., C= 100 ms.

Can we still keep the constant distance?
Example of **hard to modify**: Let’s draw a time-line to execute

<table>
<thead>
<tr>
<th>task</th>
<th>$h$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200 ms</td>
<td>75 ms</td>
</tr>
<tr>
<td>B</td>
<td>300 ms</td>
<td>50 ms</td>
</tr>
</tbody>
</table>

Assume a new task $C$ with $h = 100$ ms and $C = 25$ ms. We have to rebuild the whole schedule !!!!
Timing and implementation

Note on offline scheduling: cyclic executives are the traditional offline scheduling approach for control applications [6], and widely used in industry.

![System Flow Diagram]

- System Initialization
- Data Input
- Task Processing
- Data Output
- Idle Loop
- Periodic Interrupt
Observations about on-line scheduling

**Disadvantages**

- Keeping strict periodic execution is not possible
- Needs operating system support (overhead)

**Advantages**

- Easy to build, analyze, modify or expand
- Provides flexibility to adapt to resource availability or varying application demands
Previous example with a new schedule given by Fixed Priority online scheduling. Which is the priority assignment?

<table>
<thead>
<tr>
<th>task</th>
<th>( h )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>200 ms</td>
<td>75 ms</td>
</tr>
<tr>
<td>B</td>
<td>300 ms</td>
<td>50 ms</td>
</tr>
<tr>
<td>C</td>
<td>100 ms</td>
<td>25 ms</td>
</tr>
</tbody>
</table>
Timing and implementation

Enforcing the timing assumed in mathematical models using real-time technology

<table>
<thead>
<tr>
<th>Hard real-time periodic task model</th>
<th>Naif control task model</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ task period equal to sampling period $T = h$</td>
<td></td>
</tr>
<tr>
<td>■ sampling and actuation occur at the beginning and termination of each job execution</td>
<td></td>
</tr>
<tr>
<td>■ and task deadline bounding the time delay, $D \geq \tau$</td>
<td></td>
</tr>
</tbody>
</table>
In the previous model for implementing controllers,

- if $D = \tau$: the expected timing from the model in closed loop operation (8) is perfectly kept!!! However, task set schedulability is severely reduced !!!

- if $D > \tau$: task set schedulability is increased (more tasks can be executed) at the expenses of introducing time uncertainty in sampling and actuation operations.
Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions

2. Control of real-time control systems
   (a) Overview
   (b) Representative examples
(RT) Computing introduces time uncertainty in periods and delays

- Job released at $r_k = kh$
- Sampling jitter: $\{h_k\}$
- Latency jitter: $\{\tau_k\}$
Problems and solutions

(RT) Computing introduces time uncertainty in periods and delays

Approaches:

■ Ignore the problem

■ **Design the controller to be robust against time uncertainty**

■ Design the computing to minimize or eliminate the time uncertainty
Problems and solutions

Traditional approaches to time uncertainty

- Smith predictor
- Modified z-transform
- State-space lifting techniques

Limitation: ideal delays or mutirate (periodic) systems.

Alternative solutions
Remember the standard model (6)

\[
\begin{bmatrix}
  x_{k+1} \\
  z_{k+1}
\end{bmatrix} = \begin{bmatrix}
  \Phi(h) & \Phi(h - \tau)\Gamma(\tau) \\
  0 & 0
\end{bmatrix} \begin{bmatrix}
  x_k \\
  z_k
\end{bmatrix} + \begin{bmatrix}
  \Gamma(h - \tau) \\
  I
\end{bmatrix} u_k
\]

where \( x_{k+1} = x(kh + h) \) and \( z_{k+1} = z(kh + h) \)

If \( h \) and \( \tau \) vary at each job execution, the model is given by

\[
\begin{bmatrix}
  x_{k+1} \\
  z_{k+1}
\end{bmatrix} = \begin{bmatrix}
  \Phi(h_k) & \Phi(h_k - \tau_k)\Gamma(\tau_k) \\
  0 & 0
\end{bmatrix} \begin{bmatrix}
  x_k \\
  z_k
\end{bmatrix} + \begin{bmatrix}
  \Gamma(h_k - \tau_k) \\
  I
\end{bmatrix} u_k (13)
\]

where \( x_{k+1} = x(\sum_{i=0}^{k+1} h_i) \) and \( z_{k+1} = z(\sum_{i=0}^{k+1} h_i) \)
(13) is a family of models. Observations:

- Given the \( \{h_k, \tau_k\} \) values, a specific system is obtained and it can be analyzed
- Controllability and observability
- Stability
Given the \( \{h_k, \tau_k\} \) values, a specific system is obtained and it can be analyzed. Example.

\[
\frac{d^2y}{dt^2} = u
\]

Let’s locate the continuous closed loop poles at \( \lambda_{1,2} = -1.5 \pm 10 \cdot i \).

<table>
<thead>
<tr>
<th>task</th>
<th>h</th>
<th>C=\tau</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>120 ms</td>
<td>40 ms</td>
</tr>
<tr>
<td>B</td>
<td>90 ms</td>
<td>40 ms</td>
</tr>
</tbody>
</table>

\[
K_{120,40} = \begin{bmatrix} 75.8572 & 10.1051 & 0.3435 \\ 83.5998 & 9.7351 & 0.3225 \end{bmatrix}
\]

In isolation
Given the \( \{h_k, \tau_k\} \) values, a specific system is obtained and it can be analyzed. Example.

**Offline schedule (in ms)**

![Offline schedule diagram]

---

2nd HYCON PhD School on Hybrid Systems (2007) - Networked and Embedded Control Systems
Given the \( \{h_k, \tau_k\} \) values, a specific system is obtained and it can be analyzed. Example.

\[
\begin{align*}
\tau_1 &= 40 \\
h_1 &= 90
\end{align*}
\begin{align*}
\tau_2 &= 40 \\
h_2 &= 110
\end{align*}
\begin{align*}
\tau_3 &= 80 \\
h_3 &= 160
\end{align*}
\]

Let’s approach A as a switched system:

\[
\Phi_{cl}(h, \tau) \in \{ \Phi_{cl}(90, 40), \Phi_{cl}(110, 40), \Phi_{cl}(160, 80) \}
\]

with

\[
\begin{align*}
K_{90,40} &= \begin{bmatrix} 83.5998 & 9.7351 & 0.3225 \end{bmatrix} \\
K_{110,40} &= \begin{bmatrix} 78.4894 & 10.0117 & 0.3377 \end{bmatrix} \\
K_{160,80} &= \begin{bmatrix} 65.0282 & 12.7871 & 0.8149 \end{bmatrix}
\end{align*}
\]
Given the \( \{h_k, \tau_k\} \) values, a specific system is obtained and it can be analyzed. Is the previous analysis enough?

Example of unstable switched sequence \( A_2 A_1 A_2 A_1 \ldots \) where each subsystem \( A_i \) is stable. Given

\[
x_{k+1} = Ax_k, \quad k \geq 0, \quad x_0 = x_0, \quad A \in \{A_1, A_2\}
\]

(14)

with \( A_1 = \begin{bmatrix} 0.9 & 0.2 \\ -0.2 & -0.9 \end{bmatrix} \), \( A_2 = \begin{bmatrix} 0.9 & -0.2 \\ 0.2 & -0.9 \end{bmatrix} \), \( x_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \)
Controllability

■ Is (13) controllable? Yes. (See proof in appendix 1)

■ Therefore, we can find \( \{u_k\} \) to bring the system to \( x_{eq} \)

■ However, to compute \( \{u_k\} \), we need to know beforehand \( \{h_k, \tau_k\} \)

■ ... which in the general case, it’s not known !!!

Observation (feasibility problem): to compute \( u_k \) we need to know \( \{h_k, \tau_k\} \)
Observability

■ Is (13) observable? Yes, if the output matrix outputs the additional variable. (See proof in appendix 2)

■ No feasibility problems exist (past $h_k$, $\tau_k$, and $u_k$ are known).

Remark

Although (13) is controllable and observable, the admissible time variability is not known.
Stability. Looking at matrices

■ Single closed loop matrix

\[ \text{stable} \iff \rho(\Phi_{cl}(h, \tau)) < 1 \quad (15) \]

■ Sequence of closed-loop matrices

\[ \text{stable} \iff \rho(\Phi_{cl}(h_1, \tau_1) \cdot \Phi_{cl}(h_2, \tau_2) \cdot \ldots \cdot \Phi_{cl}(h_n, \tau_n)) < 1 \quad (16) \]

■ Closed-loop matrices randomly taken from a finite set \( \Omega \) [7]

\[ \Omega \text{ stable} \iff \exists P > 0 : \Omega^T P \Omega - P < 0, \forall \Omega \in \Omega^K, K \geq 0 \quad (17) \]
Problems and solutions

Stability. ... References

■ Delays: For time-varying but bounded delays, simply checked in a Bode plot [8]

■ Sampling periods: For uncertain sampled data systems, treated as hybrid system and using Lyapunov functions with discontinuities [9]
Problems and solutions

(RT) Computing introduces time uncertainty in periods and delays

Approaches:

- Ignore the problem
- Design the controller to be robust against time uncertainty
- Design the computing to minimize or eliminate the time uncertainty
Design the computing to minimize/eliminate the time uncertainty

Naïf approach [10]

One-sample approach [5], [11]

One-shot approach [12]
Design the computing to minimize/eliminate the time uncertainty

Model (6) in closed-loop form is based on two synchronization points, on a time reference given by the sampling instants.

\[
\begin{bmatrix}
  x_{k+1} \\
  z_{k+1}
\end{bmatrix} =
\begin{bmatrix}
  \Phi(h) & \Phi(h - \tau)\Gamma(\tau) \\
  0 & 0
\end{bmatrix}
\begin{bmatrix}
  x_k \\
  z_k
\end{bmatrix} +
\begin{bmatrix}
  \Gamma(h - \tau) \\
  I
\end{bmatrix} u_k
\]

\[
u_k = \begin{bmatrix} K_1 & K_2 \end{bmatrix}
\begin{bmatrix}
  x_k \\
  z_k
\end{bmatrix} = K_1 x_k + K_2 z_k \quad \text{with} \quad K_1 \in \mathbb{R}^{1 \times n}, \quad K_2 \in \mathbb{R}^{1 \times m}
\]
Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

Constructing the one-shot task model: changing time coordinates to a time reference given by the actuation instants.

\[ x_{k+\tau+1} = \Phi(h)x_{k+\tau} + \Gamma(h)u_k, \quad \text{with} \quad u_k = Kx_{k+\tau} \quad \text{with} \quad L \in \mathbb{R}^{1 \times n}. \quad (18) \]

\( x_{k+\tau} \) has to be predicted from \( x_k \):

\[ x_{k+\tau} = \Phi(\tau)x_k + \Gamma(\tau)u_{k-1}. \quad (19) \]
Problems and solutions

Design the computing to minimize/eliminate the time uncertainty

- All closed loop dynamics given by (18) and (19) (i.e., one-shot) can be obtained by (6) (standard).

- All closed loop dynamics given by (6) can be obtained by (18) and (19) if $m = n$.

The standard model is more general.... but one-shot admits irregular sampling

![Diagram of control system with time steps and release points](image-url)
Design the computing to minimize/eliminate the time uncertainty

Evaluation: Naif, one-sample, switching, one-shot, split [13]

<table>
<thead>
<tr>
<th></th>
<th>h</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>$T_2$</td>
<td>20</td>
<td>6</td>
</tr>
</tbody>
</table>

Voltage stabilizer (RCRC circuit)

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -918.2 & -90.9 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 918.2 \end{bmatrix} u(t)$$
Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions

2. Control of real-time control systems
   (a) Overview
   (b) Representative examples
Control of real-time control systems, also known as

- (Optimal) Sampling period selection
- Feedback scheduling
- Event-based scheduling
Objective: to efficiently use resources when control loops share limited resources. Main flavors:

- Maximize aggregated control loop performance by fully and cleverly exploiting the available resources
  \[\Downarrow\]
  Feedback scheduling (FS)

- Minimize resource utilization while bounding inter-sampling dynamics
  \[\Downarrow\]
  Event-based scheduling (ES)
Overview * Feedback scheduling

Common formulation: optimization problem

\[
\begin{align*}
\text{minimize (maximize):} & \quad \text{penalty (benefit) on control performance} \\
\text{with respect to:} & \quad \text{sampling periods / job execution} \\
\text{subject to:} & \quad \text{closed loop stability} \\
& \quad \text{task set schedulability}
\end{align*}
\]

Two type of results

- Optimal sampling periods (e.g., [14], [15],[16],[17],[18])
- Optimal job sequence (e.g., [19], [20], [21])

\[1\]Based on bounding the inter-sampling dynamics.
Overview * Event-based scheduling

Common idea: to bound the inter-sample dynamics or to ensure stability (e.g., [22], [23], [24], [25], [26], [27]). Approaches:

- Integrate an analog event detector, e.g. [22] or [24]
- Assume a coordinator aware of all plant states, e.g., [23]
- Enforce a minimum inter-execution time, e.g., [26]
- Observe the occurrence of the event (self-triggered), e.g., [25], [27]
### Overview * Taxonomy

<table>
<thead>
<tr>
<th>Which</th>
<th>What</th>
<th>Who</th>
<th>When</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criterion</td>
<td>it is solved</td>
<td>Solution</td>
<td>Timing Constraints</td>
<td>Sched.</td>
</tr>
<tr>
<td>[22] Arz99</td>
<td>Bound d.</td>
<td>ET Task</td>
<td>Online</td>
<td>Periods Aperiodic ET Missing</td>
</tr>
<tr>
<td>[23] Zha99</td>
<td>Bound d.</td>
<td>ET Coord.</td>
<td>Online</td>
<td>Job Aperiodic ET EFS</td>
</tr>
<tr>
<td>[17] Pal05</td>
<td>Optimizat.</td>
<td>TT Coord.</td>
<td>Offline</td>
<td>Periods Static periodic EDF</td>
</tr>
<tr>
<td>[18] Hen05</td>
<td>Optimizat.</td>
<td>TT Coord.</td>
<td>Online</td>
<td>Periods Varying periodic EDF</td>
</tr>
<tr>
<td>[25] Tab06</td>
<td>Bound d.</td>
<td>ET Task</td>
<td>Online</td>
<td>Periods Aperiodic TT WiP</td>
</tr>
<tr>
<td>[27] Lem07</td>
<td>Bound d.</td>
<td>ET Task</td>
<td>Online</td>
<td>Periods Aperiodic TT Elastic S.</td>
</tr>
</tbody>
</table>
Overview * FS vs. ES * Kernel

- **time-driven operation**

- **control-driven operation**
Contents

1. Real-time computing of control systems
   (a) Timing and implementation
   (b) Problems and solutions

2. Control of real-time control systems
   (a) Overview
   (b) Representative examples
      ■ periods (optimization)
      ■ sequences (optimization)
      ■ event-based
[14] On task schedulability in real-time control systems

- Set of $n$ control tasks sharing a CPU
- Performance index for each control task (cost):
  \[ \Delta J(f_i) = J_D(f_i) - J \]
- A minimum frequency for each task $f_{mi}$ must be guaranteed
Representative examples (periods)

[14] On task schedulability in real-time control systems

\[
\begin{align*}
\min \quad \Delta J &= \sum_{i=1}^{n} \omega_i \Delta J_i = \sum_{i=1}^{n} \omega_i \alpha_i e^{-\beta_i f_i} \\
\text{with respect to} \quad &f_1, f_2, \ldots, f_n \\
\text{subject to} \quad &\sum_{i=1}^{n} C_i f_i \leq A, \quad 0 < A \leq 1 \\
&f_i \geq f_{mi}, \quad i = 1, \ldots, n
\end{align*}
\]

(20)

Solution:
It \textit{statically} sets several frequencies \( \geq f_{mi} \) and the rest = \( f_{mi} \)
Representative examples (periods)

[16] Optimal state feedback based resource allocation for resource-constrained control tasks. Key observation:

![Graphs showing plant dynamics and perturbations with and without resource management (SM).](image)
[16] Optimal state feedback based resource allocation for resource-constrained control tasks

- Set of $n$ control tasks sharing a CPU
- A minimum resource share is guaranteed per task: $r_{i,\text{min}} = \frac{c_i}{h_{i,\text{max}}}$
- Performance index for each control task (benefit): $\alpha_i r_i + \beta_i$
- Instantaneous feedback: $e_i = |x_k|$
[16] Optimal state feedback based resource allocation for resource-constrained control tasks

\[
\max \sum_{i=1}^{n} \omega_i p_i(r_i)e_i = \sum_{i=1}^{n} \omega_i (\alpha_i r_i + \beta_i)|x_k|
\]

with respect to \(r_1, r_2, \ldots, r_n\)

subject to \(\sum_{i=1}^{n} \Delta r_i \leq U_s(t)\) and \(\Delta r_i \geq 0, \quad i = 1, \ldots, n\)

where \(r_i = r_{i,\text{min}} + \Delta r_i\) and \(U_s(t) = \text{available_slack}(t)\)

**Solution:** Assign all slack to the task whose plant has the largest error, where slack is the unused and thus available resources

**Drawback:** Instantaneous feedback, e.g. \(|x_k|\), may be not helpful in certain scenarios
Representative examples (periods)

System Output

Response 1

Response 2

Finite horizon

Instantaneous
[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

- Set of $n$ control tasks sharing a CPU
- Performance index for each control task based on a finite horizon prediction (cost):

$$J(x_0, h, T_{fbs}) = x_0^T S x_0 + T_{fbs} \bar{J}$$  \hspace{1cm} (22)

where

- $\bar{J} = \frac{1}{h} \left( \text{tr} \ S(h) R_1(h) + J_v(h) \right)$ is the stationary cost per time unit
- $x_0^T S x_0$ is the transient cost, where $S$ is the solution to:
[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

- the algebraic Riccati equation (23) for optimal controllers providing the optimal cost (24) for a standard quadratic cost function (25)

\[
S = \Phi^T S \Phi + Q_1 - (\Phi^T S \Gamma + Q_{12})(\Gamma^T S \Gamma + Q_2)^{-1}(\Gamma^T S \Phi + Q_{12}^T)
\]

\[
J = x_0^T S x_0 + \sum_{k=0}^{N-1} \left( \text{tr} \ S(h) R_1(h) + J_v(h) \right)
\]

\[
J = E_v \left\{ \sum_{k=0}^{N-1} \left( x(kh)^T Q_1 x(kh) + 2x(kh)^T Q_{12} u(kh) + u(kh)^T Q_2 u(kh) + J_v(h) \right) \right\}
\]

- the Lyapunov equation (26) for an arbitrary state feedback control law \( u(kh) = -K x(kh) \), to be evaluated in (24)

\[
S = (\Phi - \Gamma L)^T S (\Phi - \Gamma L) + Q_1 - Q_{12} L - L^T Q_{12}^T + L^T Q_2 L
\]

Note that \( \Phi, \Gamma, Q_1, Q_{12}, Q_2, J_v, R_1, \) and \( S \) all depend on the sampling interval \( h \).
[18] Optimal on-line sampling period assignment for real-time control tasks based on plant state information

\[
\min \sum_{i=1}^{n} J_i(x_i(t_0), h_i, T_{fbs})
\]

with respect to \( h_1, h_2, \ldots, h_n \)

subject to \( \sum_{i=1}^{n} \frac{C_i}{h_i} \leq 1 \)

\( h_i \geq 0, \quad i = 1, \ldots, n \) \hfill (27)

Solution:
Finding an analytical solution in the general case is not possible. But an approximate general solution exists and works [28]
Representative examples (periods)

Operation of [16] \((T_{fbs} \rightarrow h_i)\) or [18] \((T_{fbs} \gg h)\).
Representative examples (sequences)

[19] Integration of off-line scheduling and optimal control

- Set of \(n\) control tasks sharing a CPU
- Repeated cycle divided into \(p\) slots, cycle of length \(T_p\)
- Tasks execute within slots
- LQ controllers

\[
\min_{\tilde{u}_i} E \left[ \tilde{x}_i^T S_i \tilde{x}_i(n) + \sum_{i=1}^{n} \begin{bmatrix} \tilde{x}_i \\ \tilde{u}_i \end{bmatrix}^T \begin{bmatrix} \tilde{Q}_1 & \tilde{Q}_{12} \\ \tilde{Q}_{12}^T & \tilde{Q}_2 \end{bmatrix}_{i} \begin{bmatrix} \tilde{x}_i \\ \tilde{u}_i \end{bmatrix} \right]
\]

such that \( \tilde{x}_i(k+1) = \tilde{A}_i \tilde{x}_i(k) + \tilde{B}_i \tilde{u}_i(k) + \tilde{G}_i \tilde{v}_i(k) \)

- Let \(s\) denote a scheduling sequence and \(S_p\) a set of schedules
Representative examples (sequences)

[19] Integration of off-line scheduling and optimal control

Finding the optimal schedule formulated as a combinatorial optimization problem

\[
\min_{\text{when } s \in S_p} f_{\text{per}}(s, p)
\]

where \( f_{\text{per}}(s, p) \) is a performance measure derived from (28)

**Solution:** A periodic schedule \( \hat{s}(t) = s_1 s_2 \ldots s_p s_1 \ldots \), where \( \hat{s}(t) \) indicates the controller run at time \( t \), and a periodic linear feedback law such that \( u_{\hat{s}}(t) = K_t x_{\hat{s}(t)}(t) \)
Representative examples (event-based)

[23] Stable and real-time scheduling of a class of hybrid dynamic systems

- \(N\) continuous dynamic plants

\[
\dot{x}_i = A_i x_i + b_i u_i, \quad i = 1, \ldots, N
\]  

(30)

- Discrete-event scheduling

\[
Event(i, T_k) = \begin{cases} 
1 & \text{if } \|x_i(T_k)\| = \max_{j=1,\ldots,N} \|x_j(T_k)\| \text{ at } T_k \\
0 & \text{otherwise} 
\end{cases}
\]  

(31)

During \(t \in [T_k, T_k + h]\) plant\(_i\) runs in closed loop (rest in open loop)

- Objective: to ensure stability
Representative examples (event-based)

[23] Stable and real-time scheduling of a class of hybrid dynamic systems

- Control design specification: to ensure asymptotical and exponential stability for all plants

- Outcome:
  - Sufficient conditions
  - Stabilizing feedback gains

Observation: similar to previous feedback-scheduling approaches but using an event-based scheduling and single feedback gains.
[25] Preliminary results on state-triggered scheduling of stabilizing control tasks

- Closed loop continuous time system with discrete controller

\[
\dot{x} = f(x, k(x + e)) \quad \text{where} \quad e(t) = x(t_i) - x(t)
\]

- Event-triggered executions:

\[
|e(t)| \leq \sigma |x(t)|
\]

to enforce stability
[25] Preliminary results on state-triggered scheduling of stabilizing control tasks

- avoids accumulation points
- provides estimates of the time between consecutives executions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Static approach</td>
<td>109.05</td>
<td>105.82</td>
</tr>
<tr>
<td>Off-line RM [14]</td>
<td>121.85</td>
<td>96.59</td>
</tr>
<tr>
<td>On-line instantaneous FS [16]</td>
<td>90.63</td>
<td>64.41</td>
</tr>
<tr>
<td>On-line finite horizon FS [28]</td>
<td>100.61</td>
<td>86.99</td>
</tr>
<tr>
<td>Heuristic on-line cyclic scheduling [19]</td>
<td>62.43</td>
<td>62.48</td>
</tr>
</tbody>
</table>
Miscellaneous - No jitters

**Implementation:** one-shot task model.
Implementation: FS - [16] (left) and [18]+[28] (right)
TrueTime (http://www.control.lth.se/truetime/)

Simulation of Networked and Embedded Control Systems

- Matlab/Simulink-based simulator for real-time control systems.
- Facilitates co-simulation of controller task execution in real-time kernels, network transmissions, battery-powered devices, and continuous plant dynamics.
Summary

■ Networked and embedded control systems are everywhere
  - Resources
  - Timing
  - Dynamic behavior

■ Overcoming separation of concerns
  - Real-time computing of control systems
  - Control of real-time control systems
Appendix 1 (controllability)

Controllability. Is (13) controllable? Yes.

Proof. We assume that the standard system (2) is controllable

\[ W_c = \text{det}(\begin{bmatrix} \Gamma & \Phi & \cdots & \Phi^{n-1} & \Gamma \end{bmatrix}), \quad \text{det}(W_c) \neq 0 \]  

(33)

Let us define

\[ \phi_a(h_k, \tau_k) = \begin{bmatrix} \Phi(h_k) & \Phi(h_k - \tau_k) & \Gamma(\tau_k) \\ 0 & 0 & 0 \end{bmatrix} \]  

(34)

\[ \Gamma_a(h_k, \tau_k) = \begin{bmatrix} \Gamma(h_k - \tau_k) \\ I \end{bmatrix} \]  

(35)

\[ x_a(k) = \begin{bmatrix} x(k) \\ u(k - 1) \end{bmatrix} \]  

(36)
Appendix 1 (controllability)

Let the system state at $k = n$ be

$$x(n) = \prod_{i=1}^{n} \phi_a(h_{n-i+1}, \tau_{n-i+1})x(0) + W_c U$$  \hspace{1cm} (37)

with

$$W_c = \begin{bmatrix} \Gamma_a(h_n, \tau_n) & \ldots & \left( \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1}) \right) \Gamma_a(h_1, \tau_1) \\ \text{for } j=n \\ \ldots \\ \Gamma_a(h_1, \tau_1) & \ldots & \left( \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1}) \right) \Gamma_a(h_1, \tau_1) \\ \text{for } j=1 \\ \end{bmatrix}$$  \hspace{1cm} (38)

$$U = \left[ u^T(n-1) \ldots u^T(0) \right]^T$$
Substituting (34) and (35) into (38) we obtain

\[
W_c = \begin{bmatrix}
\underbrace{\Gamma_0(h_n \tau_n)}_{j=n} & \cdots & \underbrace{\left( \prod_{i=1}^{n-2} \Phi(h_{n-i+1}) \right) \Gamma_1(h_2, \tau_2)}_{j=1} & \cdots & \underbrace{0}_{j=1} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
I & \cdots & I & \cdots & I
\end{bmatrix}
\]

(39)

For MIMO systems, (13) is controllable if \( \det(W_c) \neq 0 \). Developing the determinant from the last row, and setting \( \tau_k = 0 \) and \( h_k = h \), we obtain condition (33)
Appendix 1 (controllability)

\[ \text{det}(W_c) \text{ is a continuous function of a continuous variable} \]

\[ \text{det}(W_c) : \mathbb{R}^{n \times n} \rightarrow \mathbb{R} \]

\[ (h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) \rightarrow \text{det}(W_c [h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n]) \]

If the original system (2) is controllable, then

\[ \exists (h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) \mid \text{det}(W_c) \neq 0 \]

And due to continuity

\[ \exists B(\{(h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n), \delta\} \mid \text{det}(W_c) \neq 0 \]
Appendix 2 (observability)

Observability. Is (13) observable? Yes, if the output matrix outputs the additional variable.

Proof. We assume that the standard system (2) is observable

\[ W_o = \det \begin{pmatrix} C \\ C\Phi(h) \\ \vdots \\ C\Phi^{n-1}(h) \end{pmatrix}, \quad \det(W_o) \neq 0 \]  \quad (40)

and we use definitions (34), (35) and (36), and we set as output matrix

\[ C_a = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \]  \quad (41)
Without losing generality, if \( u_k = 0 \), the initial state can be observed in \( n \) steps, being \( n \) the order of (13)

\[
\begin{align*}
y_a(0) &= C_a x_a(0) \\
y_a(1) &= C_a x_a(1) = C\phi_a(h_1, \tau_1)x_a(0) \\
&\vdots \\
y_a(n) &= C_a \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1})x_a(0) 
\end{align*}
\]

(42)

From (42), the observability matrix is

\[
W_o = \begin{bmatrix}
C_a \\
\vdots \\
C_a \prod_{i=1}^{n-1} \phi_a(h_{n-i+1}, \tau_{n-i+1})
\end{bmatrix}
\]

(43)
Substituting (41) and (34) into (43) we obtain

$$W_o = \begin{bmatrix}
C & 0 & 0 \\
0 & I & 0 \\
\vdots & \vdots & \vdots \\
C \prod_{i=1}^{n-1} \Phi(h_{n-i+1}) & C \prod_{i=1}^{n-2} \Phi(h_{n-i+1}) \Phi(h_1 - \tau_1) \Gamma(\tau_1) & 0
\end{bmatrix}$$

(44)

For MIMO systems, $W_o \in \mathbb{R}^{2n \times n}$. Therefore, we can construct $W_o^*$ with $n$ rows of $W_o$. Then, (13) is observable if $\det(W_o^*) \neq 0$. 
Appendix 2 (observability)

For $W_o^*$ we pick rows 2, 3, 5, 7, $\ldots$, $n - 1$ of $W_o$

$$W_o^* = \begin{bmatrix}
0 & I \\
C\Phi(h_1) & C\Phi(h_1 - \tau_1)\Gamma(\tau_1) \\
\vdots & \vdots \\
C \prod_{i=1}^{n-1} \Phi(h_{n-i+1}) & C \prod_{i=1}^{n-2} \Phi(h_{n-i+1})\Phi(h_1 - \tau_1)\Gamma(\tau_1)
\end{bmatrix}$$

(45)

With constant period and $\tau = 0$ we obtain

$$W_o^* = \begin{bmatrix}
0 & I \\
C\Phi & 0 \\
\vdots & \vdots \\
C\Phi^{n-1} & 0
\end{bmatrix}$$

(46)
Developing the determinant of (46) by the first row

\[
\det(W_o^*) = \pm \det \left( \begin{bmatrix} C \Phi \\ \vdots \\ C \Phi^{n-1} \end{bmatrix} \right) = \pm \det \left( \begin{bmatrix} C \\ \vdots \\ C \Phi^{n-2} \end{bmatrix} \right) \det(\Phi) \quad (47)
\]

Note: \( \det(\Phi) \neq 0 \) and recall (40) \( \Rightarrow \) \( \det(W_o^*) \neq 0 \).

As before, \( \det(W_o) \) is a continuous function of a continuous variable. If the original system (2) is controllable, then

\[
\exists (h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n) \quad | \quad \det(W_o^*) \neq 0
\]

And due to continuity

\[
\exists B((h_1, h_2, \ldots, h_n, \tau_1, \tau_2, \ldots, \tau_n), \delta) \quad | \quad \det(W_o^*) \neq 0
\]
References (1)

[22] Årzén, K.-E., “A Simple Event-Based PID Controller,” 14th World Congress of IFAC, January, 1999