Mating Control and Computing

Daniel Simon

DEMAR team
INRIA Sophia Antipolis - Méditerranée
LIRMM, CNRS UMR 5506, Montpellier
http://www.lirmm.fr/demar
Feedback control paradigm

U → Dynamic Process → Y
A → S
control objective
output

"Make precise systems from uncertain components"
Stabilize unstable systems

Robustness w.r.t. modeling uncertainties
Adaptability w.r.t. operating conditions

Danger of instability
Feedback control paradigm

U = K(e) Y

controller

Dynamic Process

output

objective

error

output
Feedback control paradigm

- **Stabilize** unstable systems
Feedback control paradigm

“Make precise systems from uncertain components”

- **Stabilize** unstable systems
- **Robustness** w.r.t. modeling uncertainties
Feedback control paradigm

“Make precise systems from uncertain components”

- **Stabilize** unstable systems
- **Robustness** w.r.t. modeling uncertainties
- Adaptability w.r.t. operating conditions
- Performance profile shaping
Feedback control paradigm

“Make precise systems from uncertain components”

- **Stabilize** unstable systems
- **Robustness** w.r.t. modeling uncertainties
- Adaptability w.r.t. operating conditions
- Performance profile shaping

Danger of instability
Control objectives

Design and implementation for:

**Control oriented goal**

Reaching a specified control performance under shared execution resources constraints;
Control objectives

Design and implementation for:

Control oriented goal
Reaching a specified control performance under shared execution resources constraints;

Computing oriented goal
Improving the robustness and adaptability of self-managed operating systems thanks to feedback loops.
Control objectives

Design and implementation for:

Control oriented goal
Reaching a specified control performance under shared execution resources constraints;

Computing oriented goal
Improving the robustness and adaptability of self-managed operating systems thanks to feedback loops.

Conflicting constraints!
Implementation

Continuous time and analogue control

- Biology
- Fluid level
- Fluid pressure
- Continuous electronics
Implementation

Numerical control and sampling

- Sampling and A/D conversion
- Networking
- Computing
- Control update and D/A conversion
Implementation

Numerical control and sampling
- Sampling and A/D conversion
- Networking
- Computing
- Control update and D/A conversion

Timing constraints required by the control system’s dynamics

⇒ Real-time system
Current practice

**Traditional controller design**: separation of concerns between control and computing:

**Assumptions**
- Perfectly periodic process, No jitter, No data-loss
- Perfect implementation, based on worst case
- Dedicated hardware/software

**Consequences**
- Over-constrained systems
- Under-using the resources
- Costly components and tools
Current practice

**Traditional controller design**: separation of concerns between control and computing:

- **Assumptions**
  - Perfectly periodic process, No jitter, No data-loss
  - Perfect implementation, based on worst case
  - Dedicated hardware/software

- **Consequences**
  - Over-constrained systems
  - Under-using the resources
  - Costly components and tools

$\Rightarrow$ formally provable, but costly and inflexible/unrealistic
NCS induced computing constraints

- Limited computing power → computing duration are not negligible
- Variable computing power → they are not constant
- Distributed information sources → delays, data loss or corruption
- Many sources of induced delays: computing, preemption, networking...
- Implementation induced uncertainties (in addition to the usual process uncertainties)

deterioration of a signal transferred through a network.
Needs for co-design

- Large sets of interconnected sensors/controllers/actuators; **Networked Control Systems - Cyber Physical Systems**
- Dynamic and uncertain environments;
- Heterogeneous components, asynchronous nodes;
- Cheap off-the-shelf hardware/software components.
Needs for co-design

- Large sets of interconnected sensors/controllers/actuators; Networked Control Systems - Cyber Physical Systems
- Dynamic and uncertain environments;
- Heterogeneous components, asynchronous nodes;
- Cheap off-the-shelf hardware/software components.

⇒ Co-design approach: early integration of constraints from control and computing

- Trade off: performance/robustness/energy/costs...
- Holistic view: process + algorithm + execution resources
Bridging the gap...

Gathering control & computing

control dpt.  computing dpt.

- Continuous time
- Uncertain systems
- Complex dynamics
- ODE

- Discrete events
- Determinism
- Complex data sets
- Automata

Heretical but fruitful!
Pioneering works 1999/2000

Stankovic, Lu, Son and Tao: The Case for Feedback Control
Real-Time Scheduling, *ECRTS Euromicro* 1999

Flexible scheduling of web servers


Sharing a computing resource between process controllers
Pioneering works 1999/2000

New class of control problem:
Control of $n$ identical pendulums under CPU power constraint

$n$ LQ controllers, period $h_i$, exec. time $C_i$, single CPU

Cost function $J(h) = \frac{1}{h} \int_0^h (x^T(t) \ u^T(t)) \ Q \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \ dt$
Pioneering works 1999/2000

New class of control problem:
Control of \( n \) identical pendulums under CPU power constraint

\( n \) LQ controllers, period \( h_i \), exec. time \( C_i \), single CPU

Cost function \( J(h) = \frac{1}{h} \int_0^h (x^T(t), u^T(t)) Q \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} dt \)

Problem: \( \min_{h_i} J = \sum_{i=1}^n J_i(h_i) \) under constraint \( \sum_{i=1}^n C_i/h_i \leq U_d \)
Pioneering works 1999/2000

New class of control problem:
Control of $n$ identical pendulums under CPU power constraint

$n$ LQ controllers, period $h_i$, exec. time $C_i$, single CPU

Cost function $J(h) = \frac{1}{h} \int_0^h (x^T(t), u^T(t)) Q \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} dt$

Problem: $\min_{h_i} J = \sum_{i=1}^n J_i(h_i)$ under constraint $\sum_{i=1}^n C_i/h_i \leq U_d$

theoretic feedback-scheduler: $h(k+1) = f(\Delta c(k), U_{ref}(k))$
Pioneering works 1999/2000

New class of control problem:
Control of $n$ identical pendulums under CPU power constraint

$n$ LQ controllers, period $h_i$, exec. time $C_i$, single CPU

Cost function $J(h) = \frac{1}{h} \int_0^h (x^T(t) \ u^T(t)) \ Q \left( \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \right) \ dt$

Problem: $\min_{h_i} J = \sum_{i=1}^n J_i(h_i)$ under constraint $\sum_{i=1}^n C_i/h_i \leq U_d$

theoretic feedback-scheduler: $h(k + 1) = f(\Delta c(k), U_{ref}(k))$  
on-line solving of Riccati and Lyapunov equations  
too costly for real-time implementation...
Pioneering works 1999/2000

New class of control problem:
Control of $n$ identical pendulums under CPU power constraint

$n$ LQ controllers, period $h_i$, exec. time $C_i$, single CPU

Cost function $J(h) = \frac{1}{h} \int_0^h (x^T(t) u^T(t)) Q \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} dt$

Problem: $\min_{h_i} J = \sum_{i=1}^n J_i(h_i)$ under constraint $\sum_{i=1}^n C_i/h_i \leq U_d$

theoretic feedback-scheduler: $h(k+1) = f(\Delta c(k), U_{ref}(k))$

on-line solving of Riccati and Lyapunov equations too costly for real-time implementation...

...although quite simple: LTI process, no delay, no jitter, no data loss, no uncertainty
New perspective: Control & Computing Co-Design

- Feedback loops are _not_ hard real-time
New perspective: Control & Computing Co-Design

- Feedback loops are \_not\_ hard real-time
- New degrees of freedom to implement control
  - Asynchronous sampling, anytime control
  - Varying sampling intervals
  - Data loss and (m,k)-firm policies
New perspective: Control & Computing Co-Design

- Feedback loops are _not_ hard real-time
  New degrees of freedom to implement control
  - Asynchronous sampling, anytime control
  - Varying sampling intervals
  - Data loss and (m,k)-firm policies
- Computing devices can be object of feedback control
New perspective: Control & Computing Co-Design

- Feedback loops are _not_ hard real-time
  New degrees of freedom to implement control
    - Asynchronous sampling, anytime control
    - Varying sampling intervals
    - Data loss and (m,k)-firm policies

- Computing devices can be object of feedback control
  Benefits of feedback
    - Adaptive w.r.t. operating conditions
    - Robustness w.r.t. process modeling uncertainties
Working directions

- Computing-aware control
- Control-aware computing
- Feedback control of computing devices
- Feedback scheduling
- Control of/over networks
- Integrated control and computing
Stability and robustness
Stability and robustness

Robustness assessment for Linear SISO systems:
- gain margin, phase margin
- delay margin
- module margin
- jitter margin
Stability and robustness

Robustness assessment for Linear SISO systems:
- gain margin
- phase margin
- delay margin $\frac{\Phi_M}{\omega_c}$
- module margin
- jitter margin
Jitter and control

Jitter is claimed to be harmful. . .

. . . how to minimize jitter?
Jitter and control

Jitter is claimed to be harmful...  
...how to minimize jitter?


Strategies to reduce jitter:

- **RM**, send control ASAP
- **DM**, advanced deadlines
- Non-Preemptive scheduling
Jitter and control

Jitter is claimed to be harmful... how to minimize jitter?


Strategies to reduce jitter:

- **RM**, send control ASAP
- **DM**, advanced deadlines
- Non-Preemptive scheduling
- send at period boundaries
Jitter and control

Jitter is claimed to be harmful... 

... how to minimize jitter?


Strategies to reduce jitter:
- RM, send control ASAP
- DM, advanced deadlines
- Non-Preemptive scheduling
- send at period boundaries

⇒ add a systematic delay

null jitter, but worst control performance!
Control of systems with delays

Generalization to MIMO sampled data systems:

\[ \tau(t) = t - t_k + \delta_k, \quad \dot{\tau}(t) = 1, \forall t \in [t_k, t_k+1) \]

Stability conditions provided by L-K functionals of the form

\[
V(\tau(t), x_t) = x^T(t)Px(t) + (\tau_m - \tau(t)) \int_{t_k}^{t} \dot{x}(s)R\dot{x}(s)ds + (\tau_m - \tau(t))(x(t) - x(t_k))^T S(x(t) - x(t_k))
\]

\[ \dot{V}(\tau(t), x_t) = \ldots < 0 \quad \text{Sufficient conditions (LMI)} \]

Continuous-time approaches using Lyapunov Theorem:

takes into account time-varying delays, data loss, plant uncertainties, actuators saturations, ...

... See for instance, Fridman, Seuret, Richard, Naghshtabrizi, ...
Robust control and scheduling

Slackened real-time scheduling (with A. Seuret - Gipsa&LAAS)
CIFRE Airbus, EU patent 11306103.0-2224

- Saves computing cycles
- Reduced latencies
- Deadline miss
- Sampling disturbances
Robust control and scheduling

Slackened real-time scheduling (with A. Seuret - Gipsa&LAAS)
CIFRE Airbus, EU patent 11306103.0-2224

\[
\dot{x}(t) = (A+\Delta_\mu A(t))x(t) + (B+\Delta_\mu B(t))u(t)
\]

Stability of systems with delays, varying sampling and uncertainties
(Lyapunov-Krasovskii functionals)

- \( N \), max. consecutive deadline miss?
- \( T_{slot} \) for a given failure rate?

- Saves computing cycles
- Reduced latencies

- Deadline miss
- Sampling disturbances
Robust control and scheduling

Slackened real-time scheduling (with A. Seuret - Gipsa&LAAS)
CIFRE Airbus, EU patent 11306103.0-2224

- Saves computing cycles
- Reduced latencies
- Deadline miss
- Sampling disturbances

$\dot{x}(t) = (A + \Delta \mu A(t))x(t) + (B + \Delta \mu B(t))u(t)$

Stability of systems with delays, varying sampling and uncertainties (Lyapunov-Krasovskii functionals)

CASE 1
Computing-aware control: Adaptive control rate

- CPU load adaptation using varying control intervals
- frequency scaling capabilities of modern chips
- accommodation with scheduled sensors
- stability whatever $h$ change $h \in [h_{\text{min}}, h_{\text{max}}]$
- specified performance level

self triggered, event triggered, switched systems, Lyapunov, Lie algebras, ...
Computing-aware control: Adaptive control rate

Control with varying intervals:

Linear Parameter Varying (with O. Sename - Gipsa-lab)
- LPV model, \( h \) is a varying parameter (among others)
- \( H_\infty \) synthesis with frequency dependent templates
Computing-aware control: Adaptive control rate

Control with varying intervals:

Linear Parameter Varying (with O. Sename - Gipsa-lab)
- LPV model, \( h \) is a varying parameter (among others)
- \( H_\infty \) synthesis with frequency dependent templates

- sampling dependent robust controller \( K(z,h) \)
- implementation with standard RT tools
- easy integration in QoS layer

\[
\begin{align*}
\Delta(k) &\quad 0 \\
0 &\quad \Delta(k)
\end{align*}
\]

\[
P(z)
\]

\[
K(z)
\]

\[
P'(z)
\]
Control-aware computing

Software architecture
- Application analysis -> reducing latencies on critical paths
- off-line optimization

Control aware flexible scheduling policies
- weakly-hard constraints
- \((m, k)\)-firm dropping policies
- Control-aware QoS, e.g. elastic tasks
- Bandwidth allocation: Control Server
Control task structure

- Splitting the control algorithm
- Relative urgency/importance
- Latency reduction on critical paths

while(1) {
    Wait-Clock
    A/D-Conversion
    Calculate-Output \( u(k) = f(y(k), \hat{x}(k - 1), ...) \)
    Update-State \( \hat{x}(k) = g(y(k), \hat{x}(k - 1), ...) \)
    D/A-Conversion
}

Fast stabilizing loop/ slow navigation loop

Implementation : multitasking/multi-rate (Orccad features)
Control task structure

- Splitting the control algorithm
- Relative urgency/importance
- Latency reduction on critical paths

while(1){
  Wait-Clock
  A/D-Conversion
  **Calculate-Output** \( u(k) = f(y(k), \hat{x}(k - 1), ...) \)
  D/A-Conversion
  **Update-State** \( \hat{x}(k) = g(y(k), \hat{x}(k - 1), ...) \)
}
Control task structure

- Splitting the control algorithm
- Relative urgency/importance
- Latency reduction on critical paths

while(1){
    Wait-Clock
    A/D-Conversion
    Calculate-Output $P_{high}$ $u(k) = f(y(k), \hat{x}(k - 1), ...)$
    D/A-Conversion
    Update-State $P_{low}$ $\hat{x}(k) = g(y(k), \hat{x}(k - 1), ...)$
}
Control task structure

- Splitting the control algorithm
- Relative urgency/importance
- Latency reduction on critical paths

while(1){
    Wait-Clock
    A/D-Conversion
    Calculate-Output $P_{high}$ $u(k) = f(y(k), \hat{x}(k - 1), ...)$
    D/A-Conversion
    Update-State $P_{low}$ $\hat{x}(k) = g(y(k), \hat{x}(k - 1), ...)$
}

- Fast stabilizing loop/ slow navigation loop
- Implementation : multitasking/multi-rate (Orccad features)
Flexible scheduling policies

- Finding scheduling parameters?
  - Cost functions period/delay w.r.t. performance
- Off-line non-linear optimization (simplex)
- Bandwidth allocation: Control Server
  - each controller reserves a CPU fraction (CBS)
- Minimization of QoS: Elastic Tasks
  - QoS sensitivity w.r.t. period (stiffness)
Control aware computing: Feedback scheduling

Feedback schedulers:

- Control of the scheduling parameters
- Simple dynamics, complex data sets
- Discrete process, fluid models
- Free sensors and actuators
Web server modeling

Exogenous Inputs:

Control Input:

Server State:

Outputs:

input/output latency:

rejection rate:

\[
L(N_e, M, t) = a(M, t)N_e^2 + b(M, t)N_e + c(M, t)
\]

\[
\dot{\alpha}(t) = -\frac{1}{\Delta} \left( \alpha(t) - \frac{N_e(t)}{AC(t)} \cdot \left( 1 - \frac{T_o(t)}{T_i(t)} \right) \right)
\]
Control objectives

- Maximizing availability \( AC = \frac{N_e}{1 + \gamma_L (L - L_{\text{max}})} \)
- Maximizing performance \( AC = \frac{\alpha N_e}{\alpha - \gamma_\alpha (\alpha - \alpha_{\text{max}})} \)

\( \gamma_L \) and \( \gamma_\alpha \): tuning parameters

(a) Performance - Latency  
(b) Availability - Rejection rate
Control of a H.264 video decoder

Statement:
Constant display rate (25 fps)
Decoding several frames ahead of display

Model: \( \hat{q}_{k+1} = (1 - \alpha) \hat{q}_k + \alpha q_{k-1} \)

Frame controller: Damping of frames decoding overruns

\[
d_{r_{k+1}} = t_{k+1} + \beta \times \delta_k, \quad 0 < \beta < 1
\]
Control of a H.264 video decoder

Implementation under Linux, Posix Real-time library

Experience with deadline control

Experience without control

estimated energy saving: upto 24 %
Feedback scheduling a robot controller
The ORCCAD model

RobotTasks
- Feedback Control
- Cyclic real-time data flow
- Event-based view

RobotProcedures
- Discrete Events Control
- Incremental design
- Exception processing
- Mission definition

Bottom up approach, from control to applications
The ORCCAD model

Feedback control action

- Control algorithm definition
- Invariant structure for RT life
- Modular design
- Functional parameters
- Timing parameters

Event based behaviour

- Precondition (opt. timeout)
- Synchronization
- Exceptions
  - Weak T1
  - Strong T2
  - Fatal T3

Postcondition (opt. timeout)
Feedback scheduling a robot controller (Cont’d)

Intervals

CPU load

Torques

Cost of the FS < 1%

- Negligible or moderate computing cost
- Robust w.r.t. deadlines miss
- No further need for wcet evaluation :=)
- Feasible with off-the-shelf hard/soft

Overview Statements Computing-aware control Control-aware computing
Integrating Control and Scheduling

- Hybrid complex dynamics
- Non convex cost functions
- QoS formulation gathering control and computing
- Case studies:
  - LQ control + \((m, k)\)-firm policy
  - Feedback scheduled MPC
  - \(H_\infty\) varying sampling + elastic scheduler
  - Convex optimization for LQ period selection
  - Asynchronous (event-based) control
Integrating Control and Scheduling
Integrating Control and Scheduling

- Cost effective implementation of real-time control
- Average cases rather than worst cases
- Self-management of computing devices
- New viewpoint on control systems safety
- Provability, Certification?
any questions?

Ordonnancement dans les Systèmes Temps Réel
Hermes, Maryline Chetto ed., 2014

Chapitre 11: Conception conjointe
commande-ordonnancement
any questions?

Thank you for your attention