

Co-Design of Real-Time Control Systems: The Control Server Approach

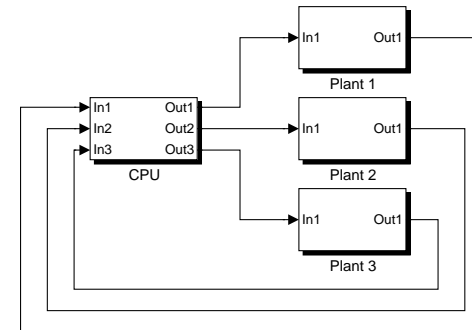
Anton Cervin and Johan Eker

Department of Automatic Control
Lund Institute of Technology
Sweden

This work has been sponsored by ARTES



The Control-Scheduling Co-Design Problem

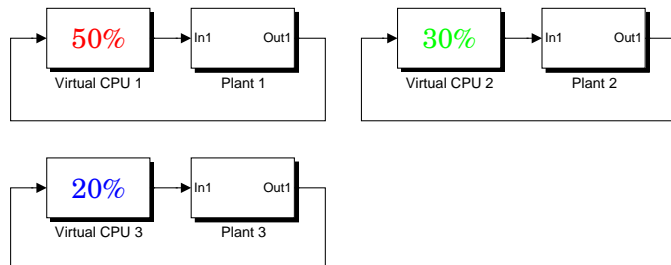


- Multiple plants controlled by a CPU with limited resources
- Controller **and** scheduling parameters should be chosen to optimize the overall control performance (QoS)
- Complex problem, due to latencies and jitter



The Control Server Approach

Divide the CPU into many virtual CPUs:



- A share of the processor is assigned to each control task
- Each control loop can be analyzed independently
- Simple to do trade-offs between the loops



Ideal Processor Sharing

How to create the virtual CPUs?

An ideal abstraction:

- Divide the time into slots
- Distribute the slots to the tasks according to their shares
- Let the slot-size $\rightarrow 0$



Problem: Overhead $\rightarrow \infty$



A Key Observation

The **observable behavior** of a task consists only of

- the **input actions**
- the **output actions**

By controlling the timing of the inputs and the outputs, we can emulate ideal processor sharing with very little overhead!

5



The Control Server: Overview

Features:

- isolation between unrelated tasks
- short input-output latencies
- minimal jitter
- a simple interface between control and real-time design
- predictable control and real-time behavior during overruns
- the possibility to combine several tasks into a composite task with predictable control and real-time behavior

Achieved by a combination of **reservation-based scheduling (CBS)** and **time-triggered I/O**.

6



The Constant Bandwidth Server (CBS)

- Proposed in [Abeni and Buttazzo, 1998]
- Based on EDF (earliest-deadline-first) scheduling
- Originally developed to handle soft and aperiodic tasks
- Achieves CPU reservations through **budgets** and **deadline postponements**

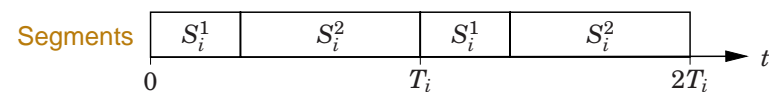
7



The Control Server Task Model

A Control Server task τ_i is described by

- a utilization factor U_i
- a period T_i
- a release offset ϕ_i
- a set of **segments** (subtasks) $S_i^1 \dots S_i^{n_i}$



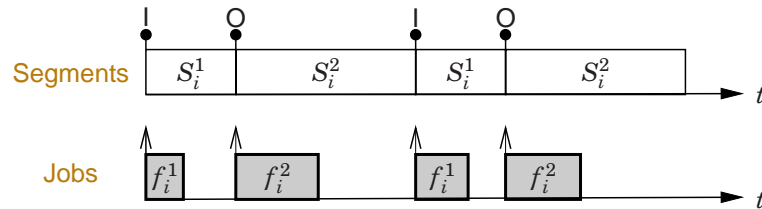
8



The Model, Cont'd

Each segment S_i^j is associated with

- a set of inputs (I) – read at the beginning
- a set of outputs (O) – written at the end
- a code function f_i^j – released at the beginning



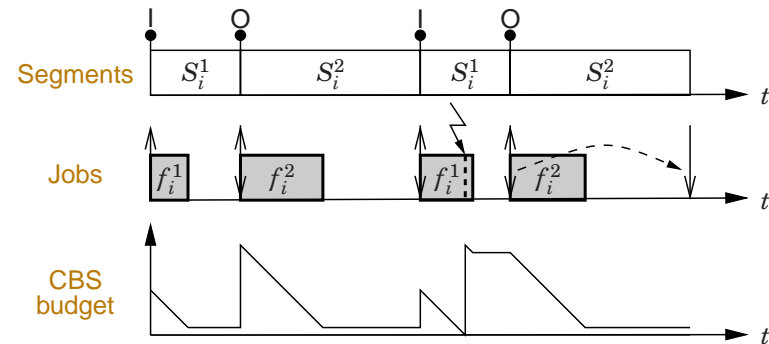
9



The Model, Cont'd

The jobs of a task are served by a CBS with

- a bandwidth U_i
- a budget proportional to the current segment length



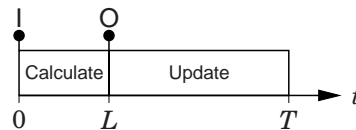
10



Real-Time Control Components

A control algorithm can often be divided into two parts:

Calculate and Update:



Choose segment lengths proportionally to the execution times of the parts \Rightarrow

- Sampling period T depends on U
- Input-output latency L depends on U

11



Control Performance

In general, the performance of a controller depends on

- the sampling period
- the input-output latency
- the jitter

In traditional scheduling models, the latency and the jitter are very complicated functions of the scheduling parameters.

In our model, the performance of the control component depends only on U !

12



Control Performance Index

Using the **Jitterbug** toolbox [Lincoln and Cervin, 2002], we can compute a **quadratic performance index** on the form

$$J(U) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T x^T(t) Q x(t) dt$$

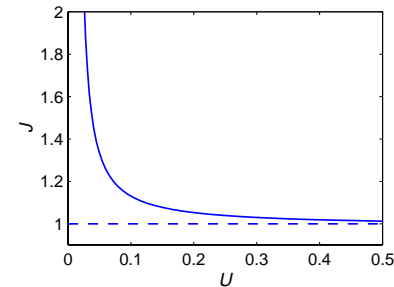
for the control component.

This performance index takes the implementation into account



Example of Performance Index

PID controller controlling an inverted pendulum:

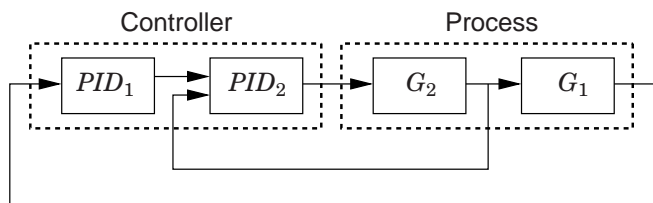


Direct trade-off between control performance and CPU use

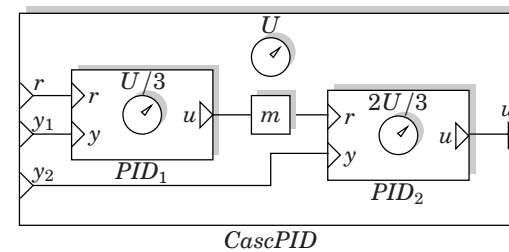
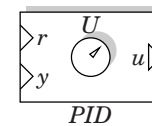


Communicating Control Tasks

Example: Cascade controller:



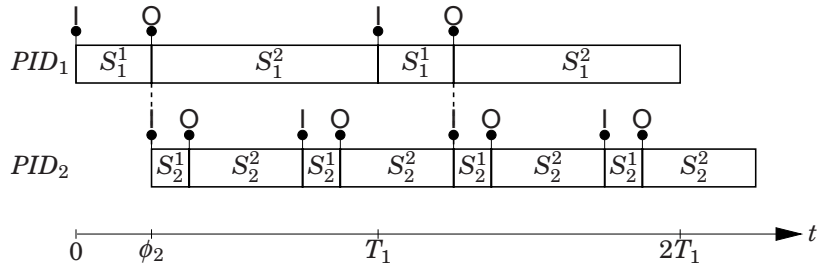
Composite Components





Synchronization

Segment layout:



Offset ϕ_2 assigned to minimize the end-to-end latency

17



The Composite Controller

- can be viewed as a new component with $U = U_1 + U_2$
- has predictable control performance
- has predictable real-time properties

18



Conclusion

This talk has presented the Control Server approach:

- Combination of static and dynamic scheduling
 - Static scheduling of I/O
 - Dynamic scheduling of computations
- A fraction of the CPU is reserved for each task
- Both schedulability and control performance depend on the utilization factor only
- The low-level scheduling details are eliminated from the design process
- Has been implemented and tested in the control lab

19



Further Reading

- A. Cervin and J. Eker: "The Control Server: A Computational Model for Real-Time Control Tasks." In *Proc. 15th Euromicro Conference on Real-Time Systems*, July 2003.
- A. Cervin: *Integrated Control and Real-Time Scheduling*. PhD Thesis TFRT-1065, Department of Automatic Control, Lund Institute of Technology, Sweden, April 2003.

(Both available at <http://www.control.lth.se/~anton>)

20