Embedded Control Systems

Samarjit Chakraborty

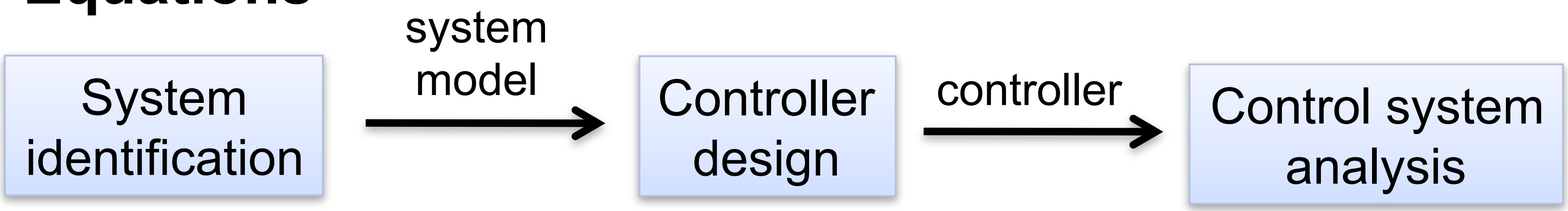
www.rcs.ei.tum.de

TU Munich, Germany

Joint work with Dip Goswami (now at TU/e), Reinhard Schneider (now at Audi), Wanli Chang (now at Singapore Institute of Technology), Anuradha Annaswamy (MIT), Arne Hamann (Bosch), and many others ...

Control Systems Design

Equations



System identification

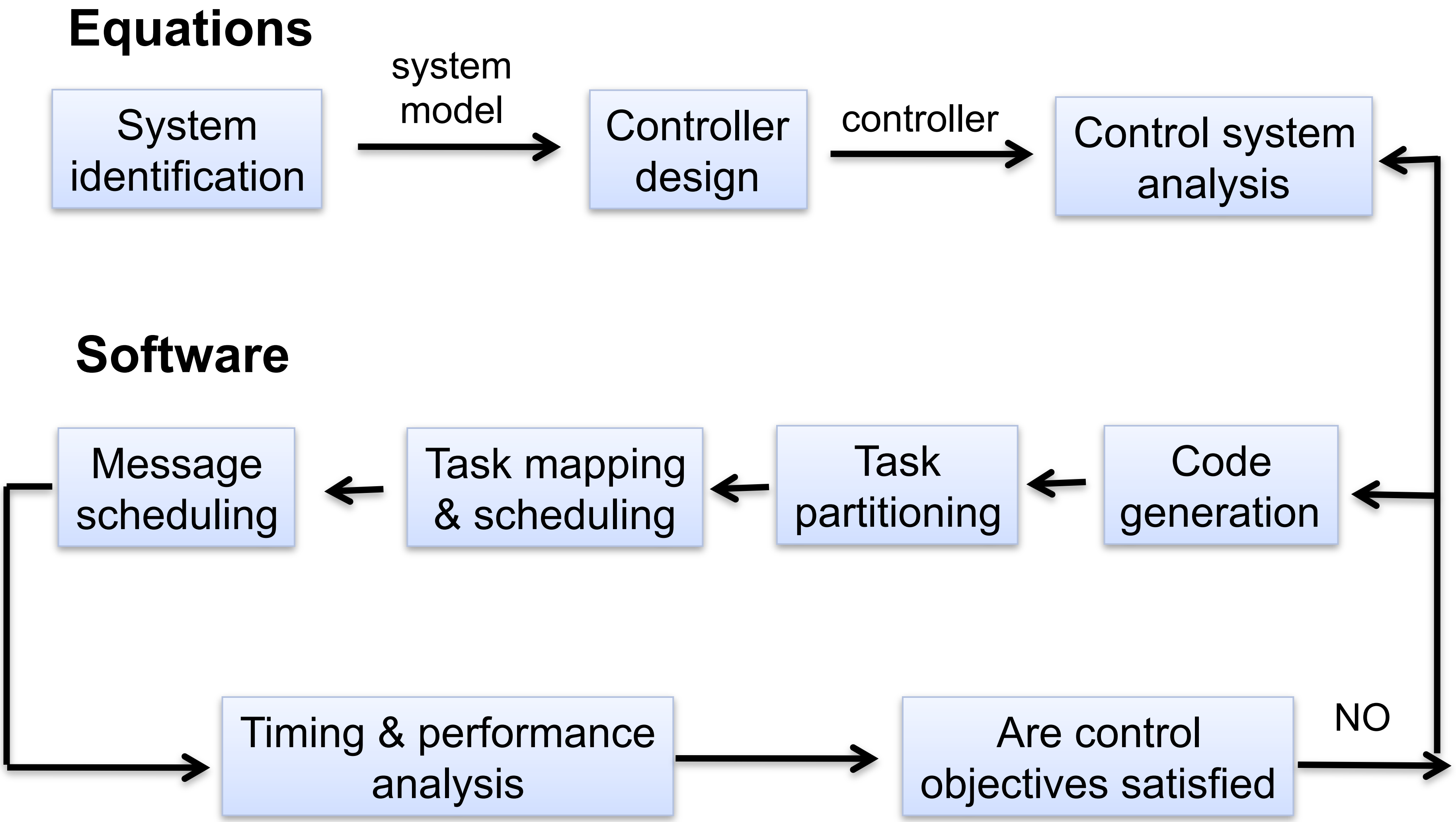
system model

Controller design

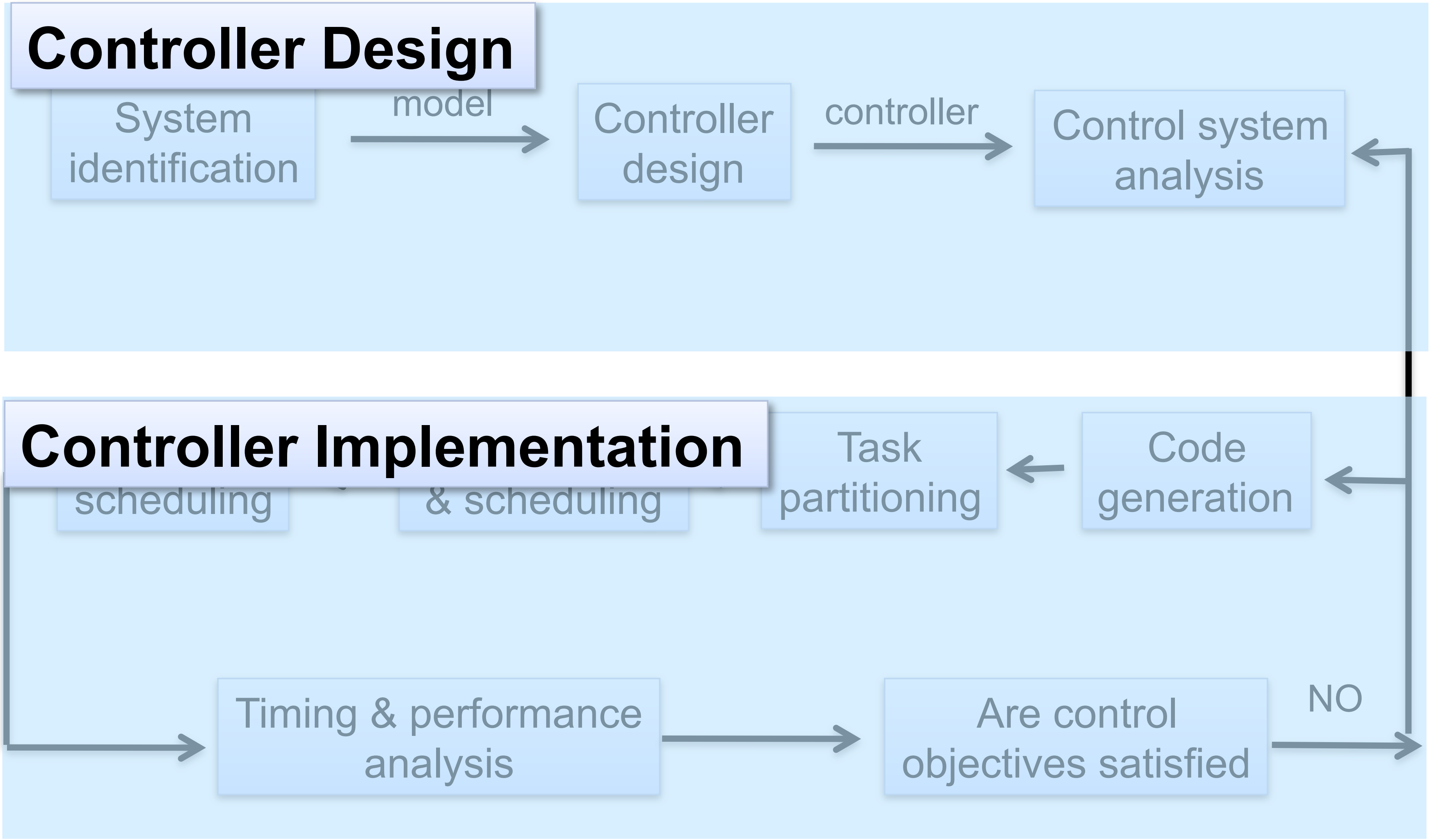
controller

Control system analysis

Control Systems Implementation



The Design Flow



Controller Design

System identification

model

Controller design

controller

Control system analysis

Controller Implementation

scheduling

& scheduling

Task partitioning

Code generation

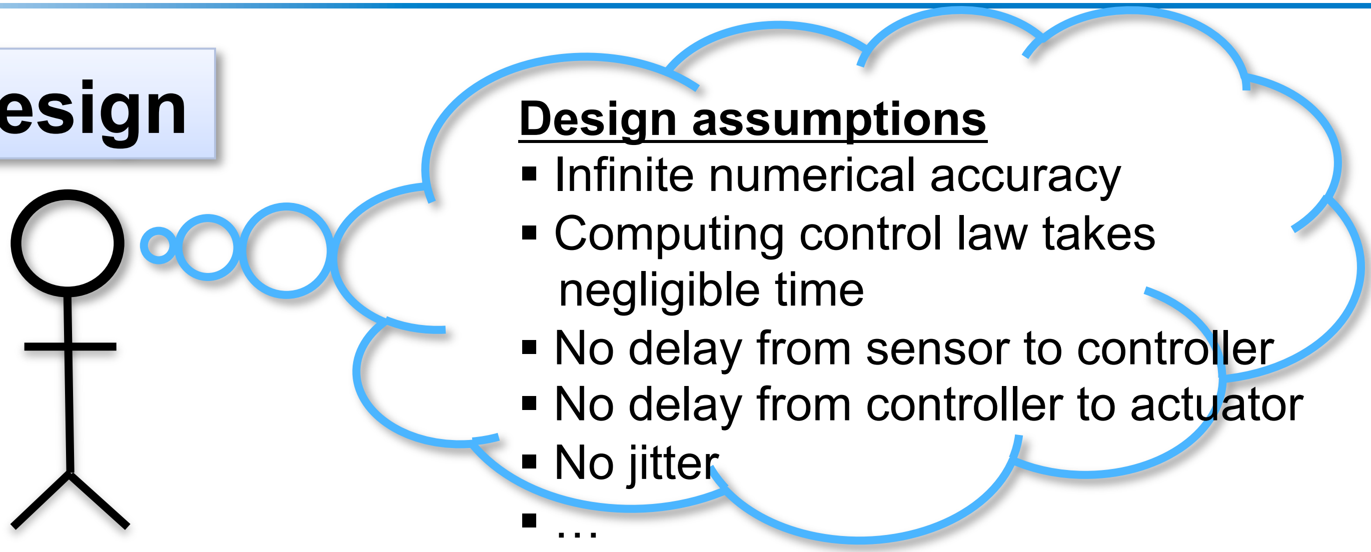
Timing & performance analysis

Are control objectives satisfied

NO

The Design Flow

Controller Design

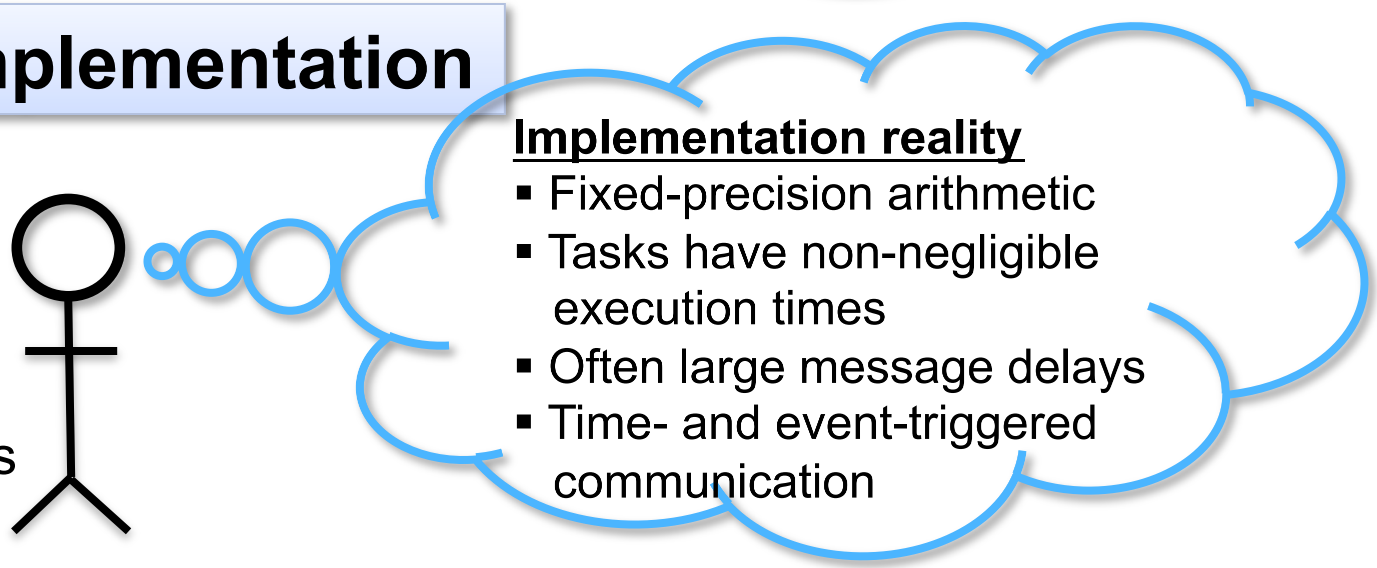


Control theorist

Design assumptions

- Infinite numerical accuracy
- Computing control law takes negligible time
- No delay from sensor to controller
- No delay from controller to actuator
- No jitter
- …

Controller Implementation



Embedded systems engineer

Implementation reality

- Fixed-precision arithmetic
- Tasks have non-negligible execution times
- Often large message delays
- Time- and event-triggered communication

The Design Flow

Controller Design

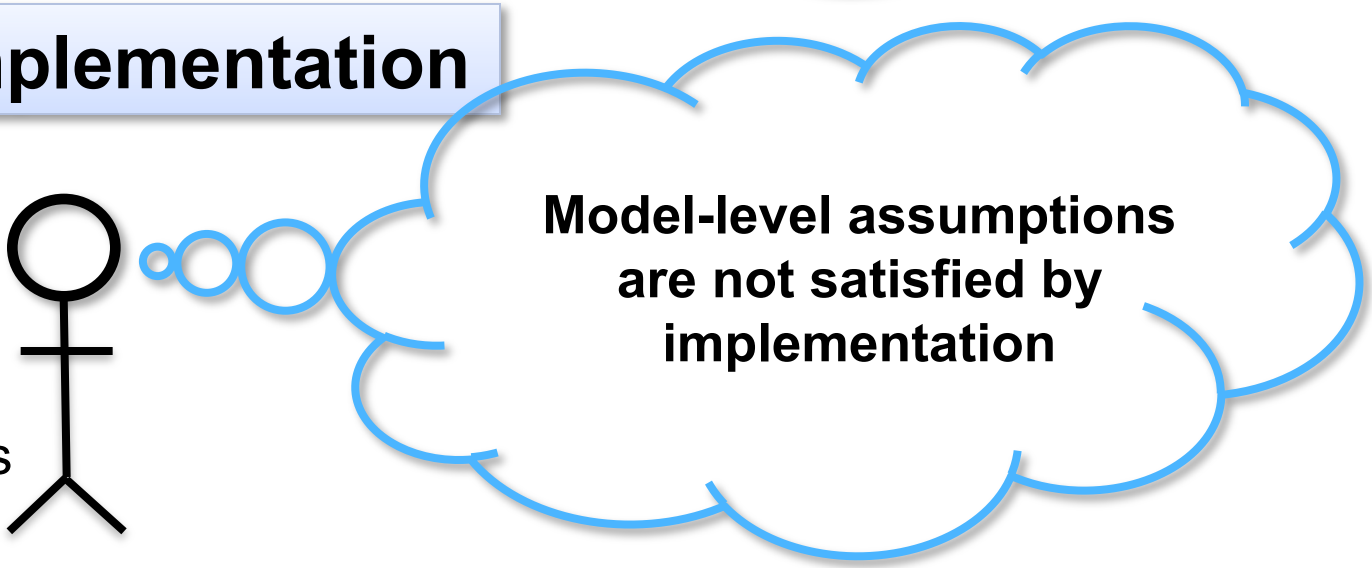
Control theorist

These are implementation details

Not my problem!

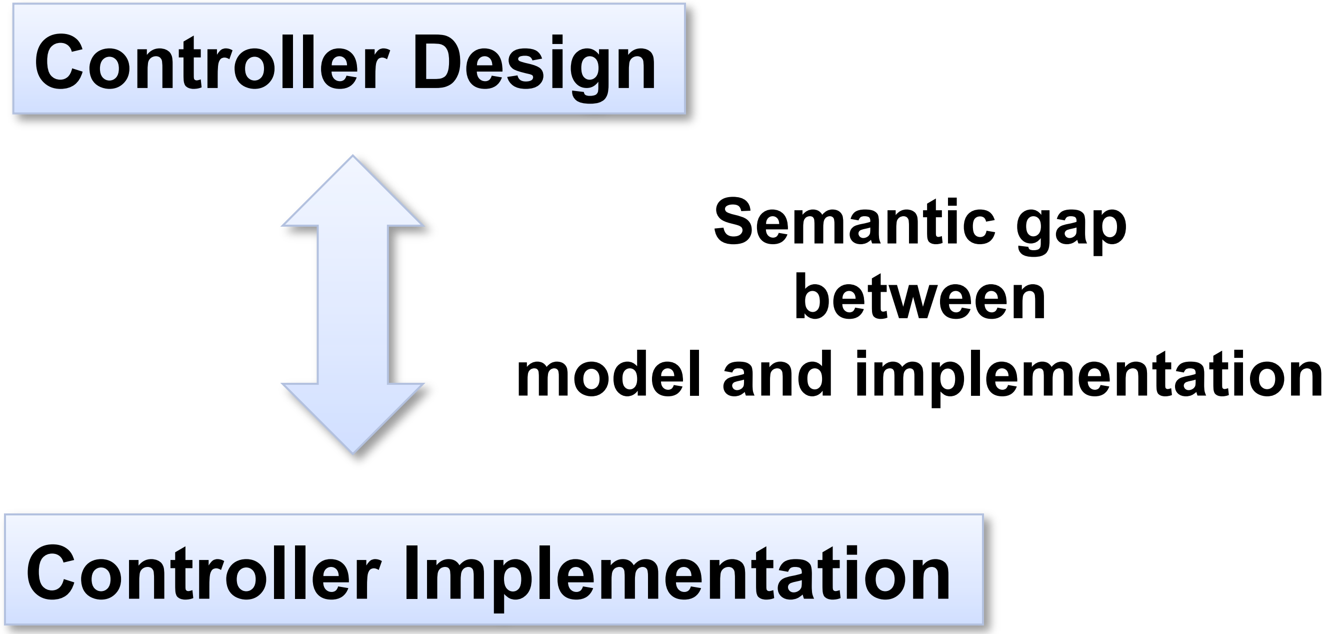
Controller Implementation

Embedded systems engineer



Model-level assumptions are not satisfied by implementation

Semantic Gap



Controller Design

Semantic gap
between
model and implementation

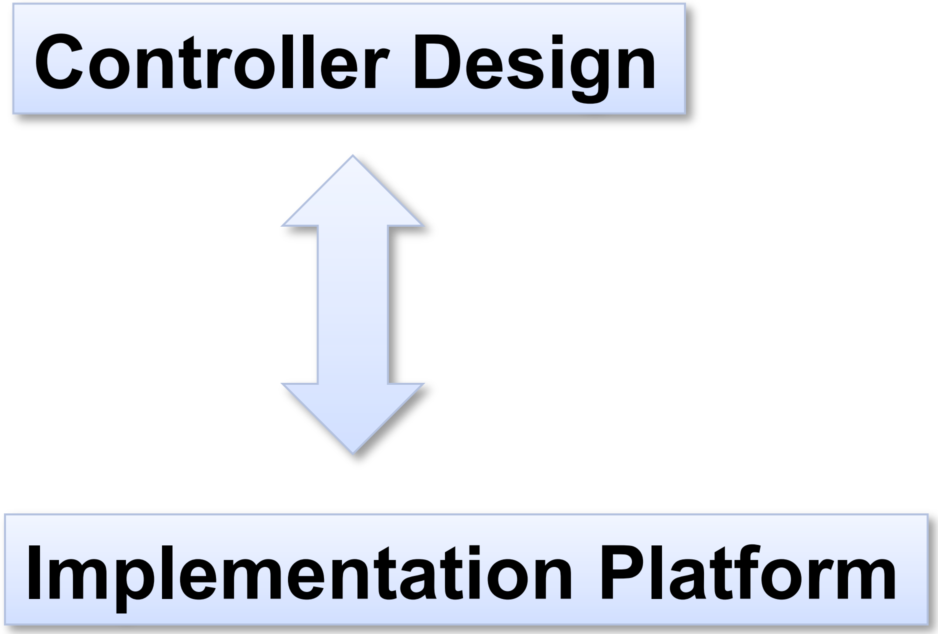
Controller Implementation

Research Questions?

- How should we quantify this gap?
- How should we close this gap?

Solution: Controller/Architecture Co-design

Resource-aware Controller Design



Controller Design

Implementation Platform

stability, settling time,
peak overshoot, ...

computation, communication
memory, power, ...

- Traditionally, Computer Science has been concerned with *efficient* implementation of algorithms
- What are notions of efficiency? Computation, communication, memory, energy, ...
- Metrics for control algorithms have been different ...

Control Tasks - Characteristics

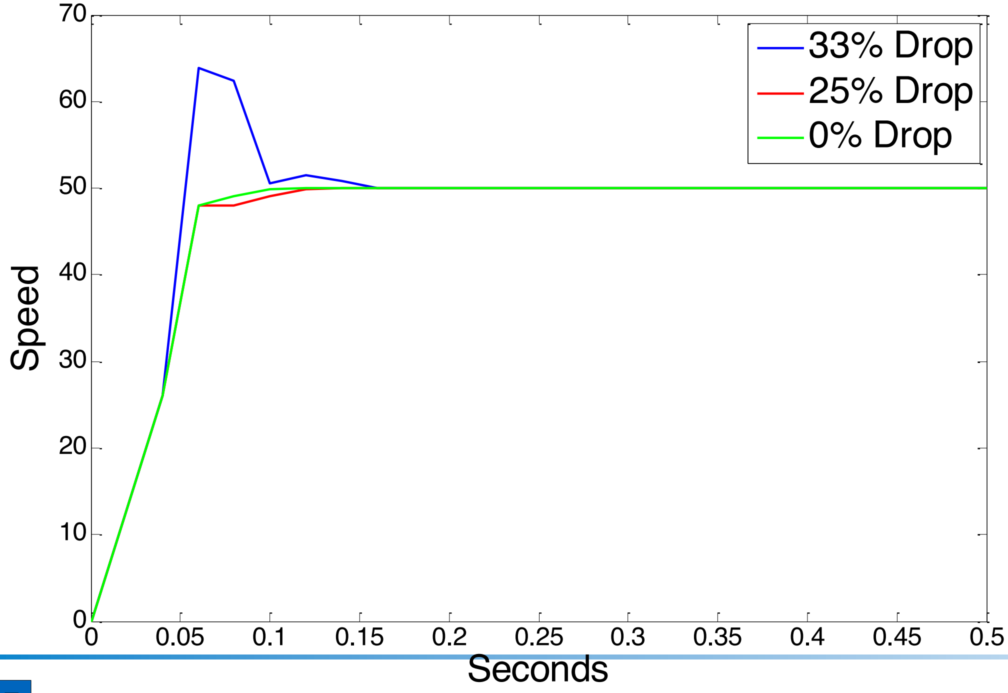
The deadlines are usually not ***hard*** for control-related messages

DC motor: $$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}=\begin{bmatrix}-\frac{b}{J} & -\frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix}+\begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}V$$

$$\rightarrow \dot{x}(t) = Ax(t) + Bu(t)$$

Objective: $\dot{\theta} \rightarrow 50$

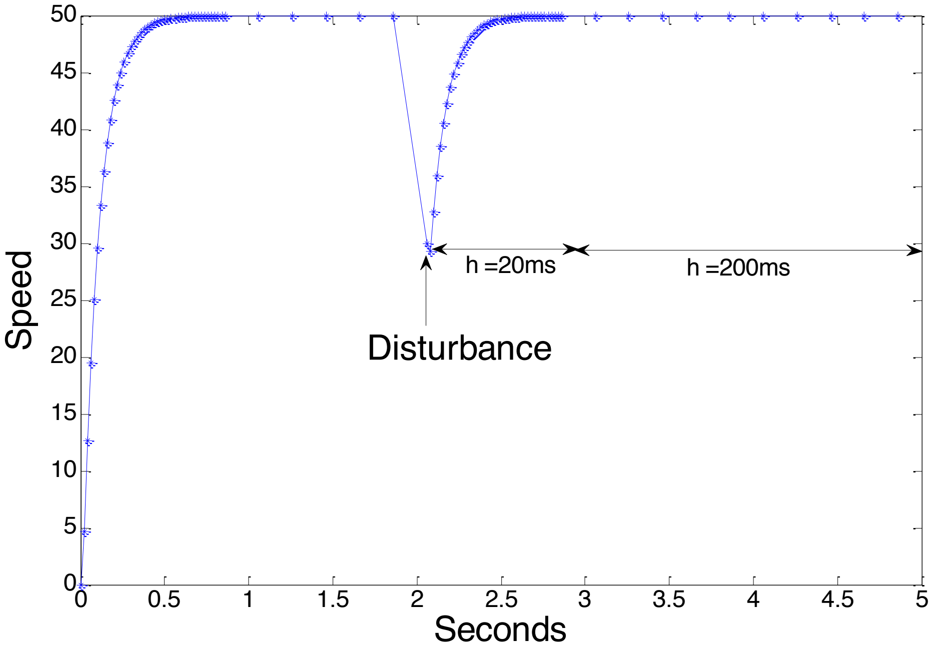
A fraction of feedback signals being dropped



Controllers can be designed to be robust to drops and deadline-misses

Control Tasks - Characteristics

Sensitivity of control performance depends on the ***state*** of the controlled plant



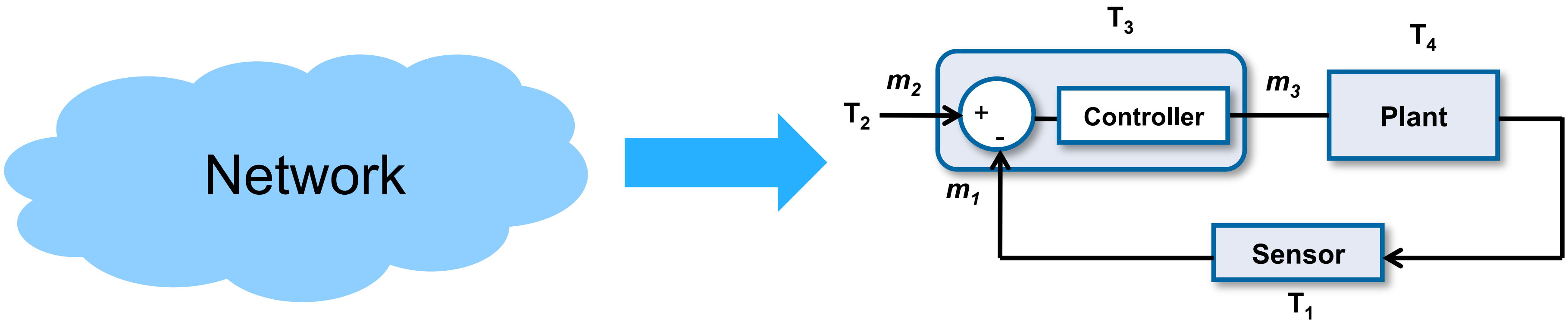
(1) The computation requirement at the steady state is less, i.e., sampling frequency can be reduced (e.g., event-triggered sampling)

(2) The communication requirements are less at the steady state, (e.g., lower priority can be assigned to the feedback signals)

Bottomline

- Embedded and Real-time Systems
  - Meeting deadlines is the center of attraction
- Co-design
  - Deadline takes the back seat
  - As a result, the design space becomes bigger
  - Resulting design is better, robust, cost-effective …
- Design objectives shift from “lower level” metrics like deadlines to metrics governing system dynamics (like stability)

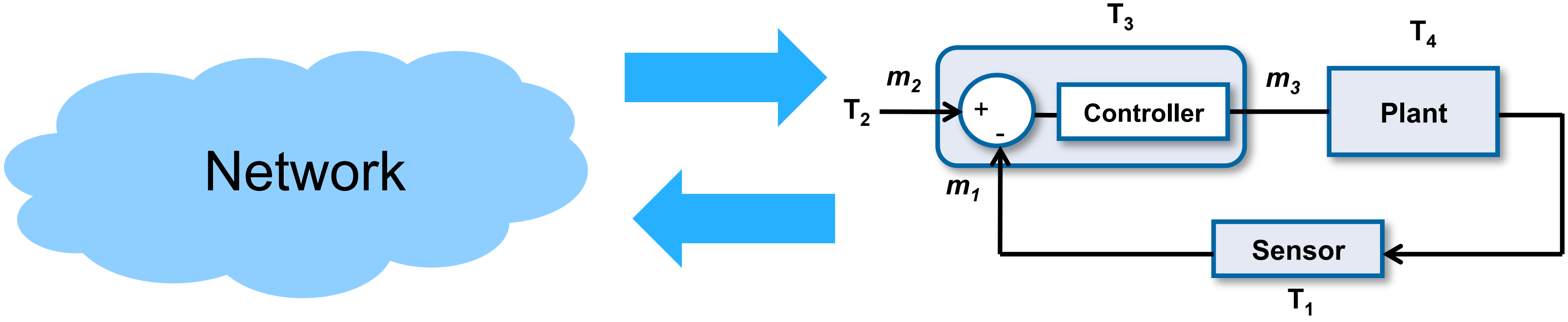
What about NCS?



Networked Control Systems

- Take network characteristics into account when designing the control laws
  - Packet drops, delays, jitter …

What about NCS? Answer: ANCS



Arbitrated Networked Control Systems

- ANCS – We can design the network
  - By taking into account control performance constraints
- Problem: How to design the network?
- Given a network, how to design the controller?
  - NCS problem
- Co-design Problem: **How to design the network and the controller together?**

A Simple Case

Controller Design: Continuous Model

- We have a linear system given by the state-space model

$$\dot{x} = A\ x\ + B\mathrm{u}$$
$$y = Cx$$

- For *n-dimensional* Single-Input-Single-Output (SISO) systems

$$x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}'$$

$$A \in R^n \times R^n, B \in R^n \times 1, C \in 1 \times R^n$$

- Objective

$$y \to r \text{ as } time \to \infty$$

- u = ?

Controller Design: Continuous Model

- Control law

$$u = Kx + Fr$$

r = reference

K = feedback gain

F = static feedforward gain

- How to design K?
- How to design F?

Computing Feedback Gain

- Choose the desired closed-loop poles at

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}$$

- Pole placement is a constrained optimization problem (poles: decision variables, objective: control performance, constraints: saturation, stability)
- Using Ackermann's formula we get

$$K = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \gamma^{-1} H(A)$$

where

$$\gamma = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$

$$H(A) = (A-\alpha_1 I)(A-\alpha_2 I)(A-\alpha_3 I)\cdots(A-\alpha_n I)$$

- Poles of (A+BK) are at $\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}$

Static Feedforward Gain

$u = Kx + Fr$

K →pole placement

F → static feedforward gains are calculated as follows

Closed-loop system

$$\dot{x} = (A + BK)x + BFr$$
$$y = Cx$$

Taking Laplace transform

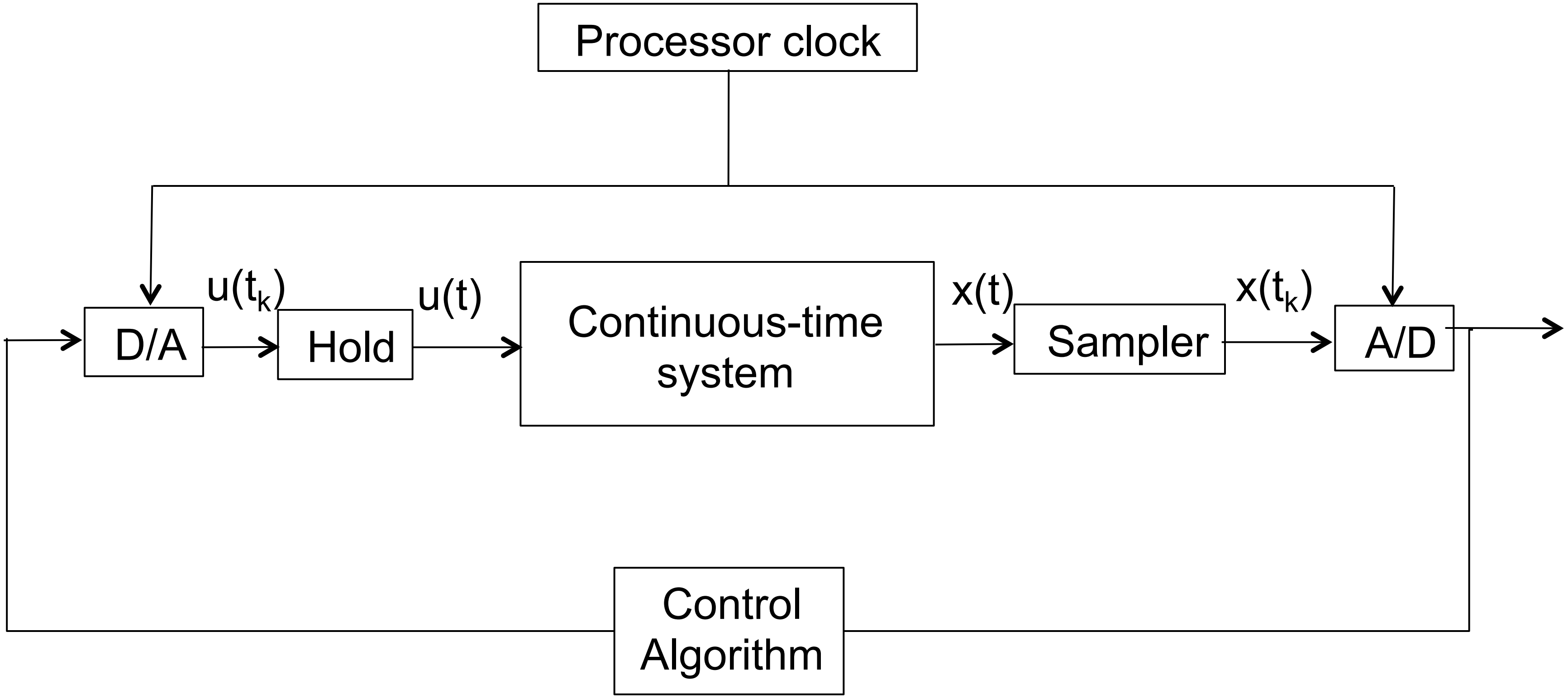
$$\rightarrow X(s) = (sI - A - BK)^{-1}BFR(S)$$
$$\rightarrow Y(s) = CX(s) = C(sI - A - BK)^{-1}BFR(S)$$
$$\rightarrow G_{cl}(s) = \frac{Y(s)}{R(s)} = C(sI - A - BK)^{-1}BF$$

F should be chosen such that y(t) → r (constant) as t → ∞ i.e.,

Using final value theorem $\Longrightarrow$ $\lim_{s\to 0} sY(s) = r$

$$\Longrightarrow \quad F = \frac{1}{C(-A-BK)^{-1}B}$$

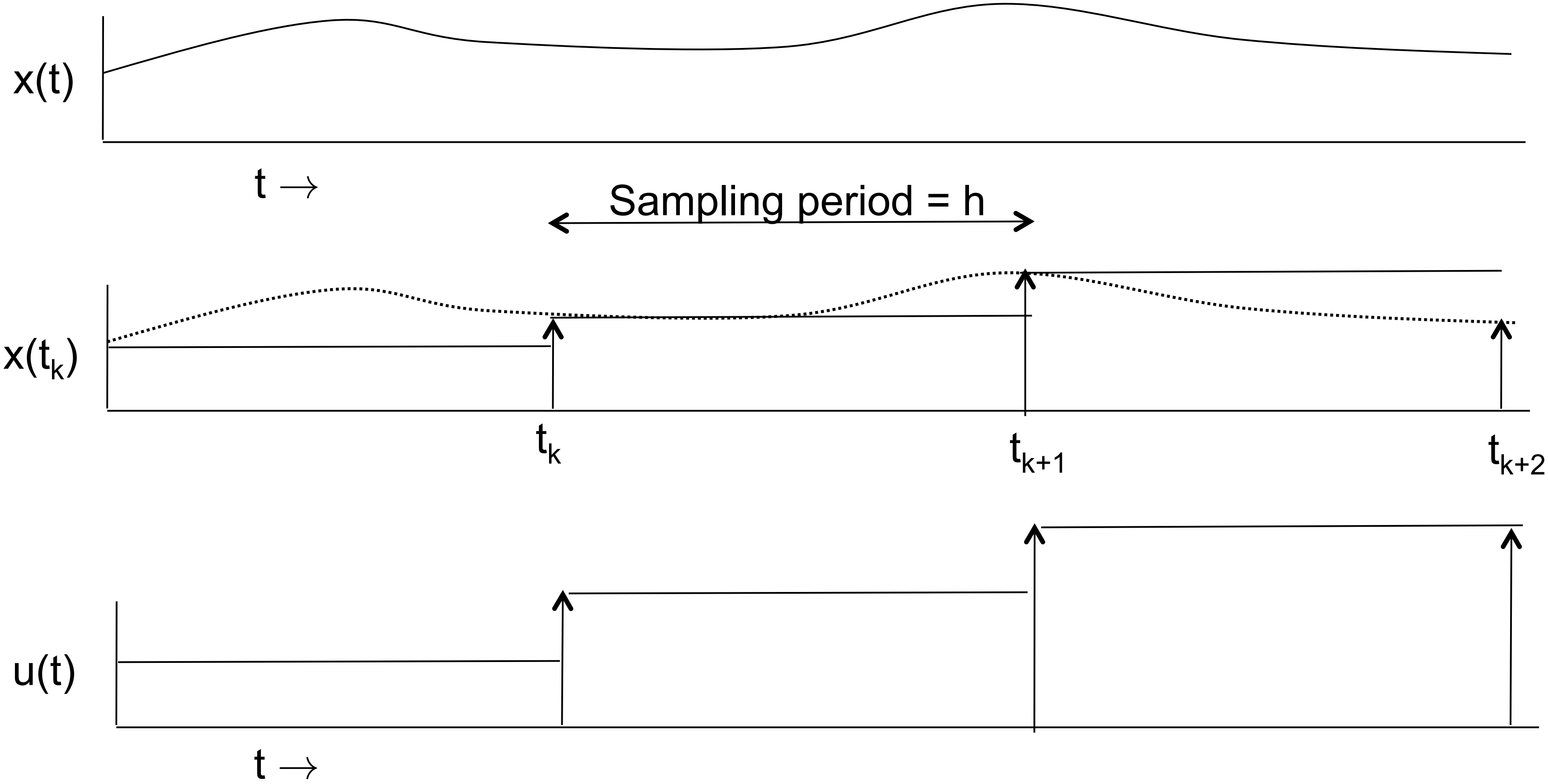
Digital Platform: Sample and Hold



D/A → digital-to-analog converter
A/D → analog-to-digital converter

- Input u(t) is piecewise constant
- Look at the sampling points

ZOH Sampling



piecewise contant

$\rightarrow u(t) = u(t_k)$ for $t_k \leq t \leq t_{k+1}$

Design: Step 1 (Discretization)

$$\dot{x} = A\ x + B\mathrm{u}$$
$$y = Cx$$

$\Downarrow$ ZOH periodic sampling with period = h

$$x[k+1] = \phi x[k] + \Gamma u[k]$$
$$y[k] = Cx[k]$$

where

$$\begin{aligned} \phi &= e^{Ah} \\ \Gamma &= \int_0^h e^{As} B ds \end{aligned}$$

$$e^{Ah} = I + Ah + \frac{A^2h^2}{2!} + \frac{A^3h^3}{3!} + \ldots$$

Design: Step 2 (Controller Design)

- Given system: $x[k+1] = \phi x[k] + \Gamma u[k]$
  $y[k] = Cx[k]$
- Control law: $u[k] = Kx[k] + Fr$

Objectives

(i) Place system poles

(ii) Achieve y → r as t → ∞

(iii) Design K and F

$$\Downarrow$$

1. Check controllability of (ϕ,Γ) → must be controllable. γ must be invertible.

$$\gamma = \left[\begin{array}{ccccc} \Gamma & \phi\Gamma & \phi^2\Gamma & \cdots & \phi^{n-1}\Gamma \end{array} \right]$$

2. Apply Ackermann's formula $K = -\left[\begin{array}{cccc} 0 & 0 & \cdots & 1 \end{array} \right] \gamma^{-1} H(\phi)$

3. Feedforward gain $F = \frac{1}{C(I-\phi-\Gamma K)^{-1}\Gamma}$

Step 2

- Given

$$x[k+1] = \phi x[k] + \Gamma u[k]$$
$$y[k] = Cx[k]$$

$$\phi \in R^n \times R^n, \Gamma \in R^n \times 1, C \in 1 \times R^n$$

- The control input $u[k] = Kx[k]$ such that closed-loop poles are at

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \cdots & \alpha_n \end{bmatrix}$$

- Using Ackermann's formula:

$$K = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \gamma^{-1} H(\phi)$$

where

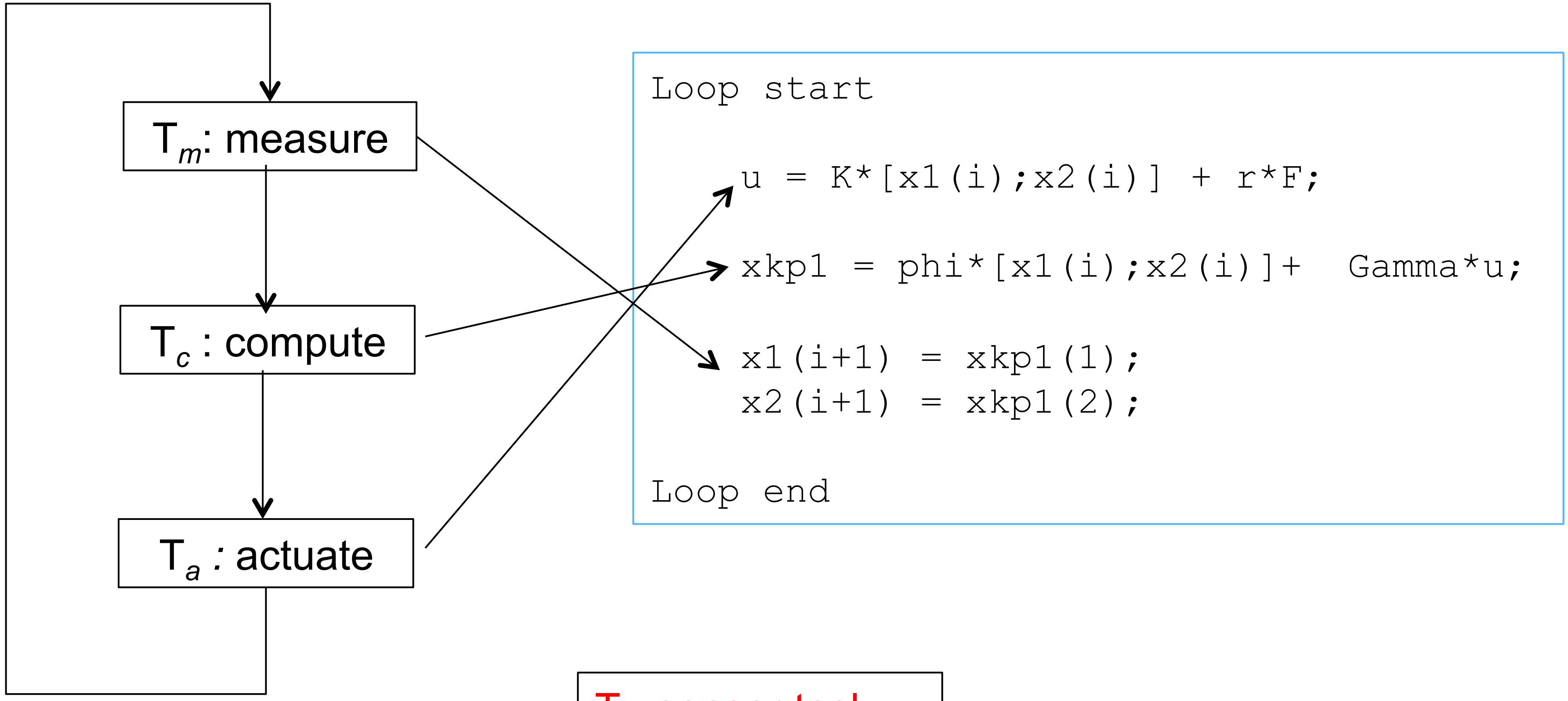
$$\gamma = \begin{bmatrix} \Gamma & \phi\Gamma & \phi^2\Gamma & \cdots & \phi^{n-1}\Gamma \end{bmatrix}$$
$$H(\phi) = (\phi - \alpha_1 I)(\phi - \alpha_2 I)(\phi - \alpha_3 I)\cdots(\phi - \alpha_n I)$$

Continuous Vs Discrete Time

Continuous-time	ZOH periodic sampled
$\dot{x}= A\ x + B\mathrm{u}$ $y = Cx$	$x[k+1] = \phi x[k] + \Gamma u[k]$ $y[k] = Cx[k]$
Input: $u = kx + Fr$	Input: $u[k] = Kx[k] + Fr$
Controllability matrix: $\gamma = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$	Controllability matrix: $\gamma = \begin{bmatrix} \Gamma & \phi\Gamma & \phi^2\Gamma & \cdots & \phi^{n-1}\Gamma \end{bmatrix}$
$K = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}\gamma^{-1}H(A)$	$K = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix}\gamma^{-1}H(\phi)$
$F = \frac{1}{C(-A-BK)^{-1}B}$	$F = \frac{1}{C(I-\phi-\Gamma K)^{-1}\Gamma}$

The Real Case

Feedback loop



T_m: measure

T_c : compute

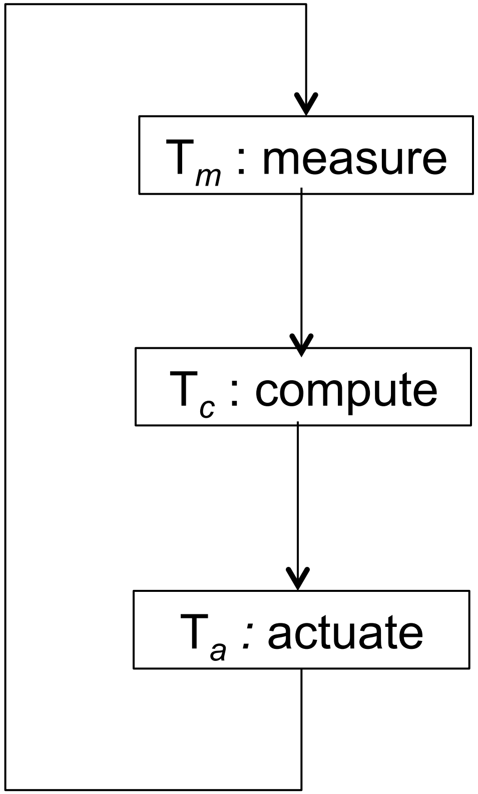
T_a : actuate

```
Loop start

        u = K*[x1(i);x2(i)] + r*F;

        xkp1 = phi*[x1(i);x2(i)]+  Gamma*u;

        x1(i+1) = xkp1(1);
        x2(i+1) = xkp1(2);

Loop end
```

T_m sensor task
T_c controller task
T_a actuator task

Control Loop

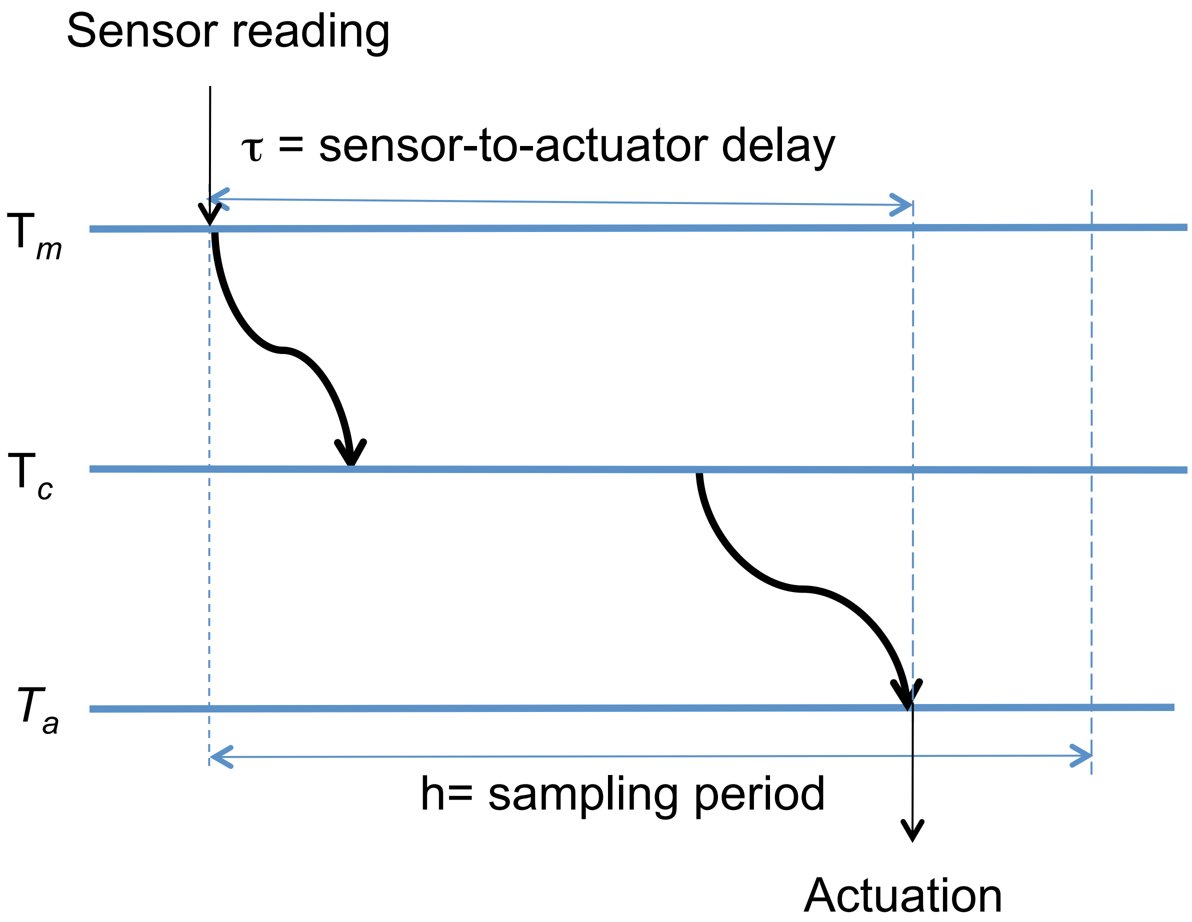
Feedback loop



T_m : measure

T_c : compute

T_a : actuate



Sensor reading

τ = sensor-to-actuator delay

T_m

T_c

T_a

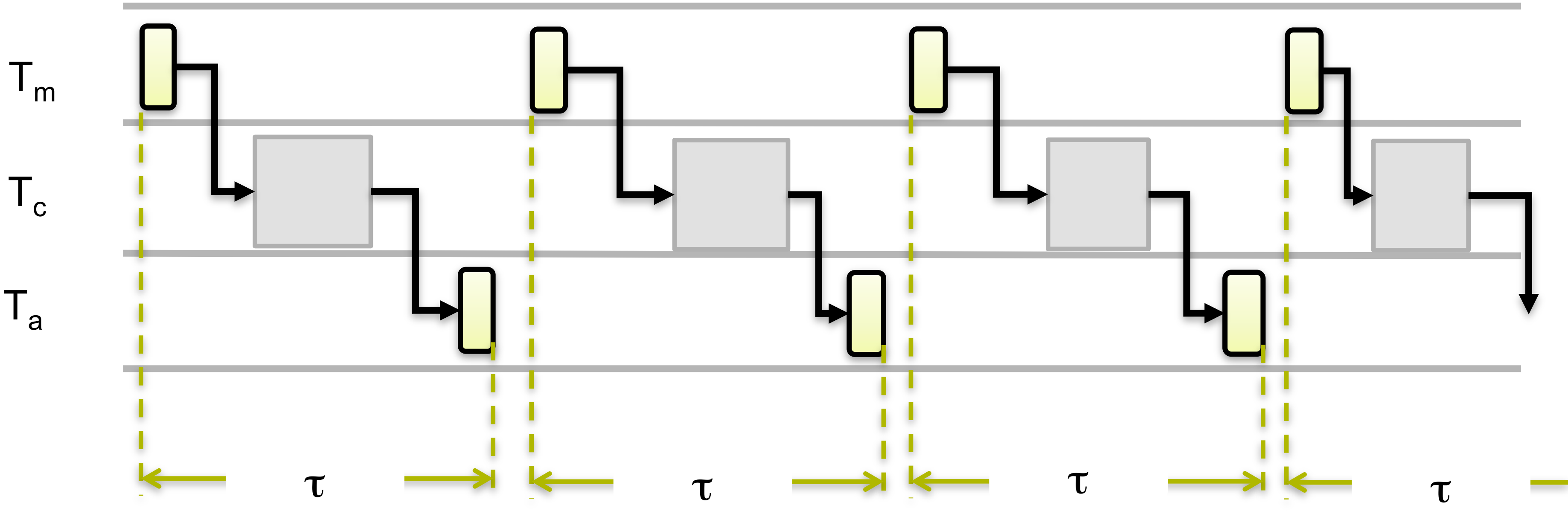
h= sampling period

Actuation

Ideal design assumes: $\tau = 0$ or $\tau << h$

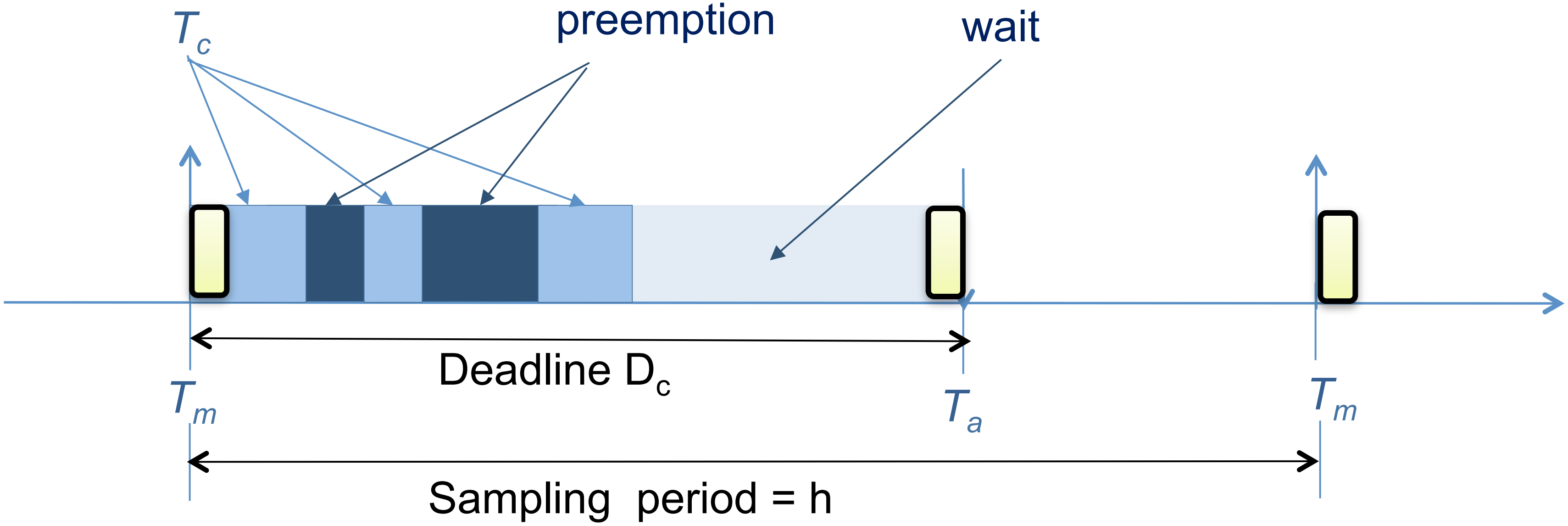
Control Task Triggering

- In general, T_m and T_a tasks consume negligible computational time and are time-triggered
- T_c needs finite computation time and is preemptive
- When multiple tasks are running on a processor, T_c can be preempted



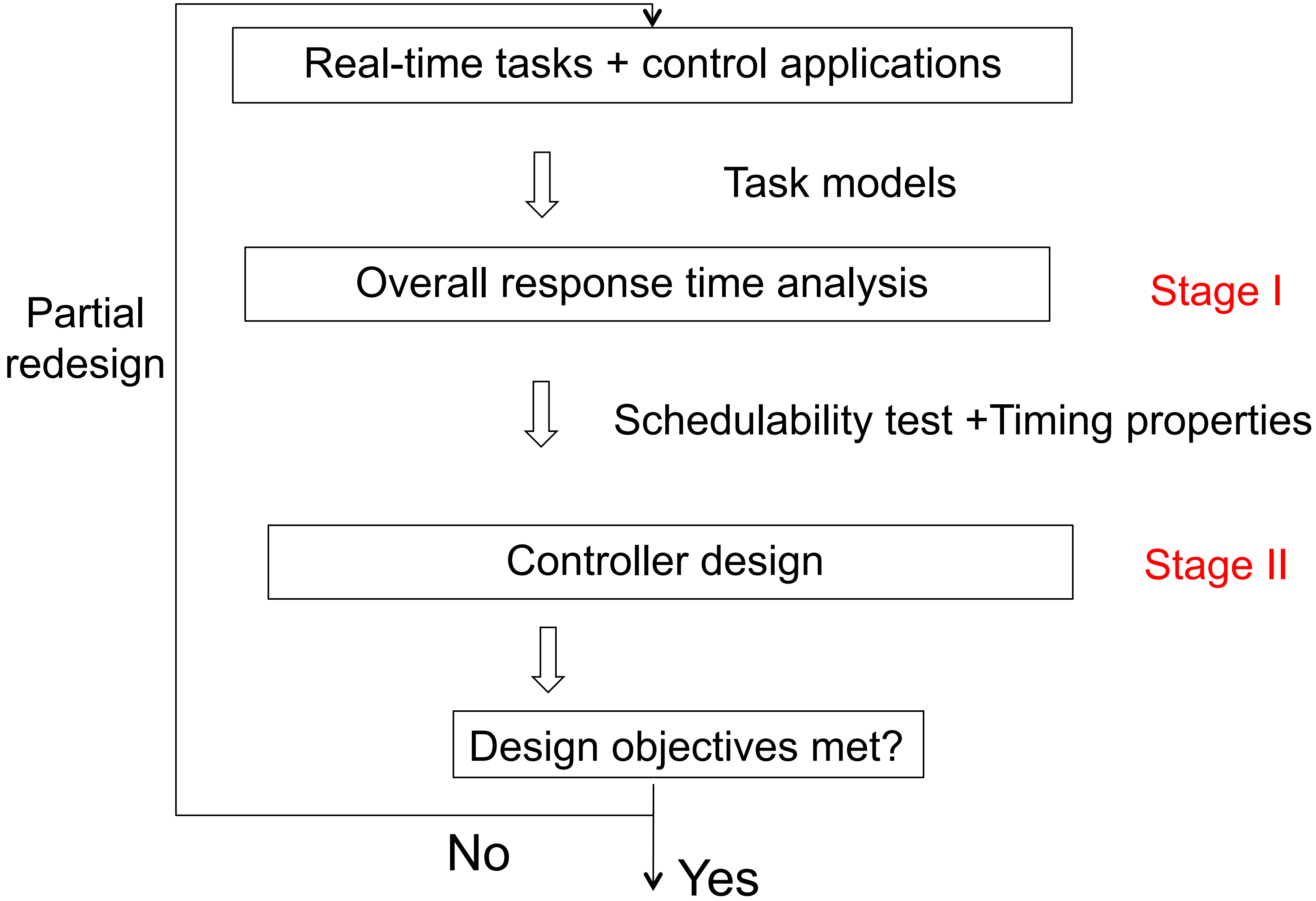
Sensor-to-actuator delay: τ

Control Task Model: Constant Delay

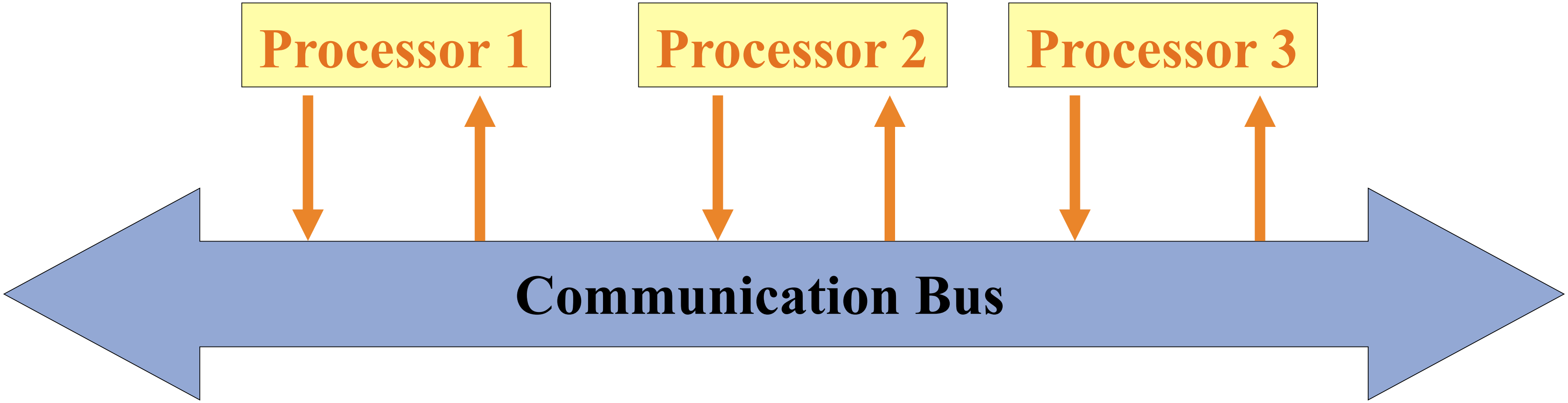


$$\text{sensor-to-actuator delay } \tau = D_c$$

Design Steps


Real-time tasks + control applications
Task models
Overall response time analysis
Stage I
Partial redesign
Schedulability test +Timing properties
Controller design
Stage II
Design objectives met?
No
Yes

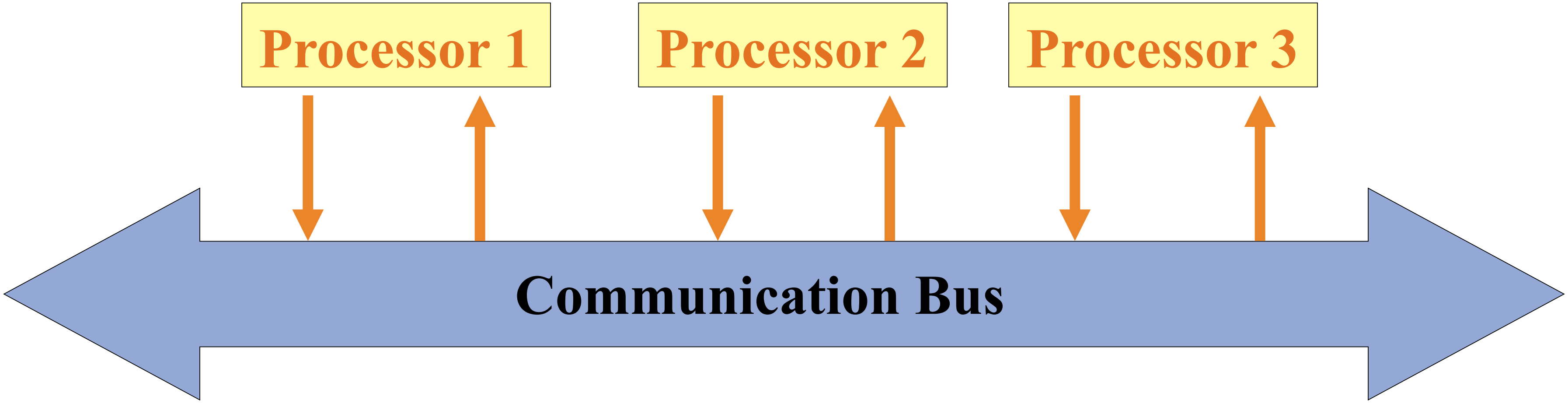
Bus Arbitration Policies



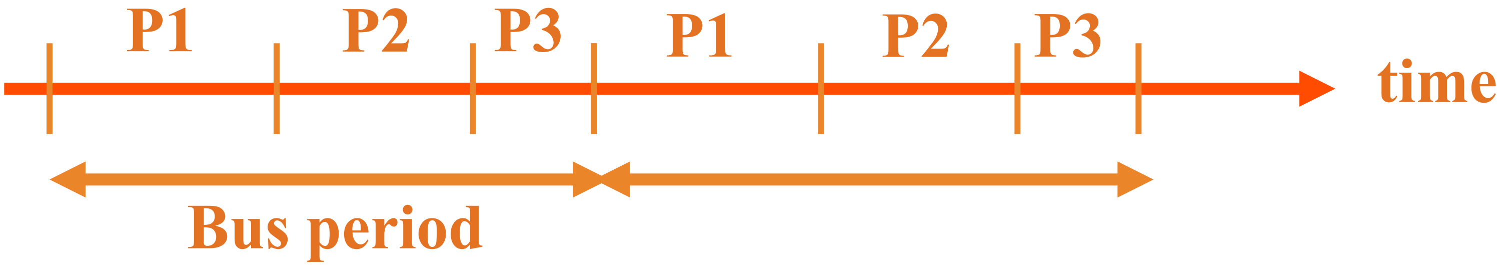
When multiple processors want to transmit data at the same time, how is the contention resolved?

- Using a bus arbitration policy, i.e., determine who gets priority
- Examples of arbitration policies
  - Time Division Multiple Access (TDMA)
  - Round Robin (RR)
  - Fixed Priority (FP)
  - Earliest Deadline First (EDF), …

Time Vs Event-Triggered Arbitration

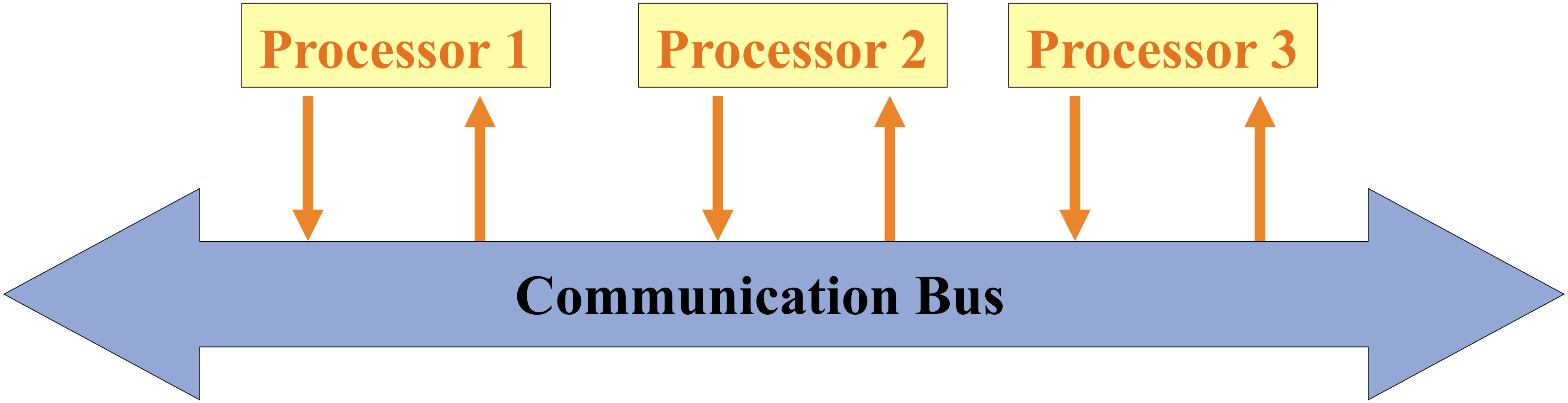


Time-triggered arbitration policy:

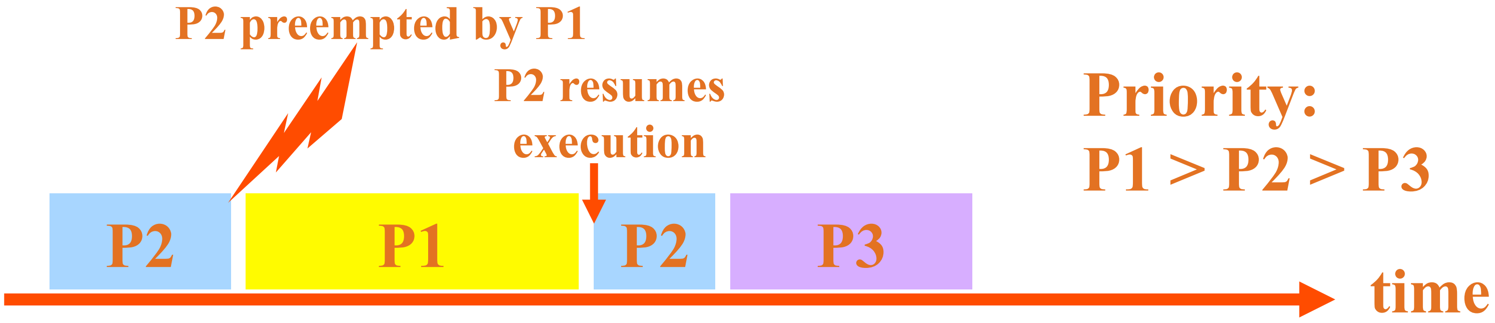


All components have a priory knowledge of the message send/ receive time instants (global time)

Time Vs Event-Triggered Arbitration

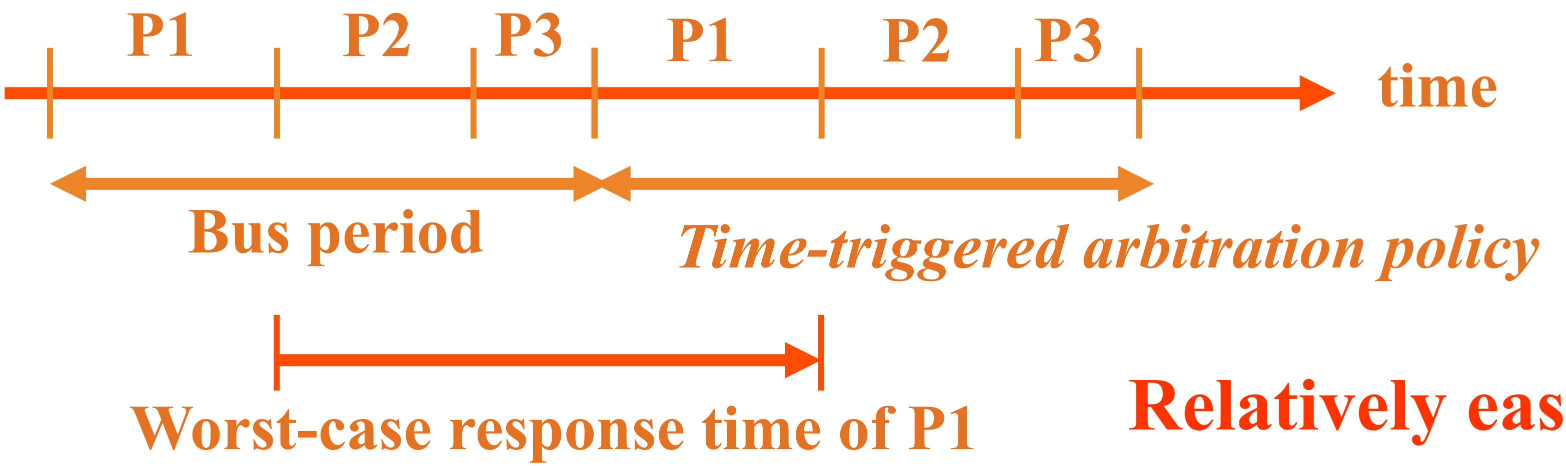


Event-triggered arbitration policy:



Priority:
P1 > P2 > P3

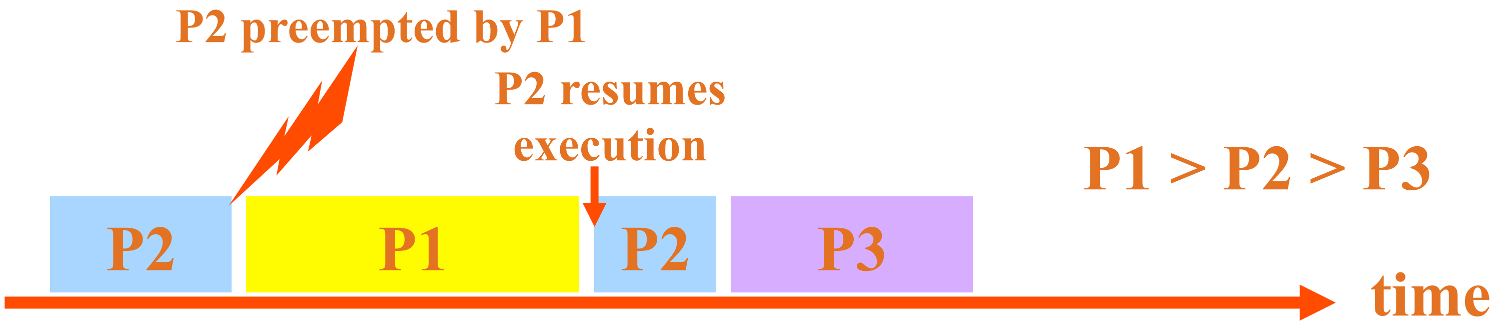
Computing Response Times



Time-triggered arbitration policy

Worst-case response time of P1

Relatively easy!



P1 > P2 > P3

Event-triggered arbitration policy

Response time of ith task $r_i = e_i + \sum_{j \in hp(i)} (\lceil r_i/T_j \rceil \times e_j)$

Response Time in Event-Triggered

Response time of i^{th} task: $r_i = e_i + \sum_{j \in hp(i)} (\lceil r_i/T_j \rceil \times e_j)$

- **hp(i)** – set of all tasks having priority higher than i
- T_j – period of task j
- $\lceil r_i/T_j \rceil$ – number of times task i is preempted by task j
- e_i – execution time of task i
- Response time of task i is made up of:
  - Execution time of task i and
  - the time during which i is preempted and higher priority tasks are running

Example: Compute WCRT for task 3

Prio	e_i	T_i
1	1	6
2	2	8
3	**4**	**10**
4	2	20

hp(3) = {1,2}

Fixed point computation:

$r_3^0 = e_3$ *(initial value)*

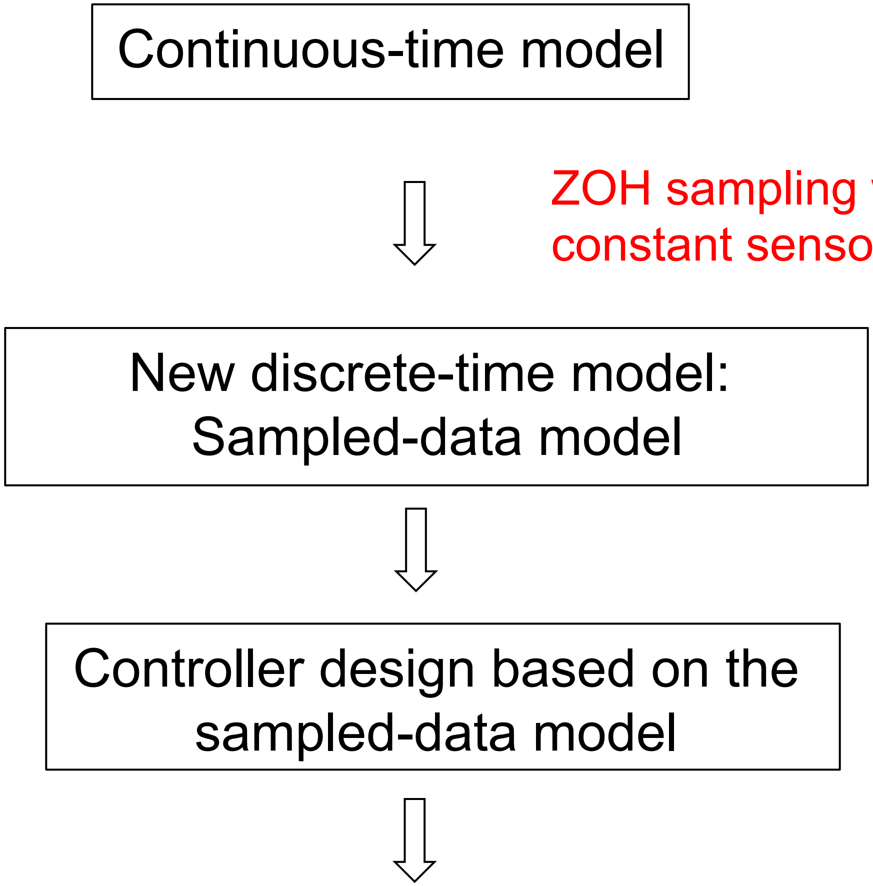
$$r_3^1 = e_3 + \sum_{j \in \{1,2\}} (\lceil r_3^0/T_j \rceil \times e_j) = 4 + \lceil 4/6 \rceil 1 + \lceil 4/8 \rceil 2 = 7$$

$$r_3^2 = e_3 + \sum_{j \in \{1,2\}} (\lceil r_3^1/T_j \rceil \times e_j) = 4 + \lceil 7/6 \rceil 1 + \lceil 7/8 \rceil 2 = 8$$

$$r_3^3 = e_3 + \sum_{j \in \{1,2\}} (\lceil r_3^2/T_j \rceil \times e_j) = 4 + \lceil 8/6 \rceil 1 + \lceil 8/8 \rceil 2 = 8$$

$$r_3^3 = r_3^2$$

Controller design steps for $D_c < h$



Continuous-time model

$$\dot{x} = A\,x + Bu$$
$$y = Cx$$

ZOH sampling with period h and
constant sensor-to-actuator delay D_c

Step I

New discrete-time model:
Sampled-data model

$$x[k+1] = f_1(x[k], u[k])$$
$$y[k] = f_2(x[k])$$

Step II

Controller design based on the sampled-data model

$$u[k] = f(x[.])$$

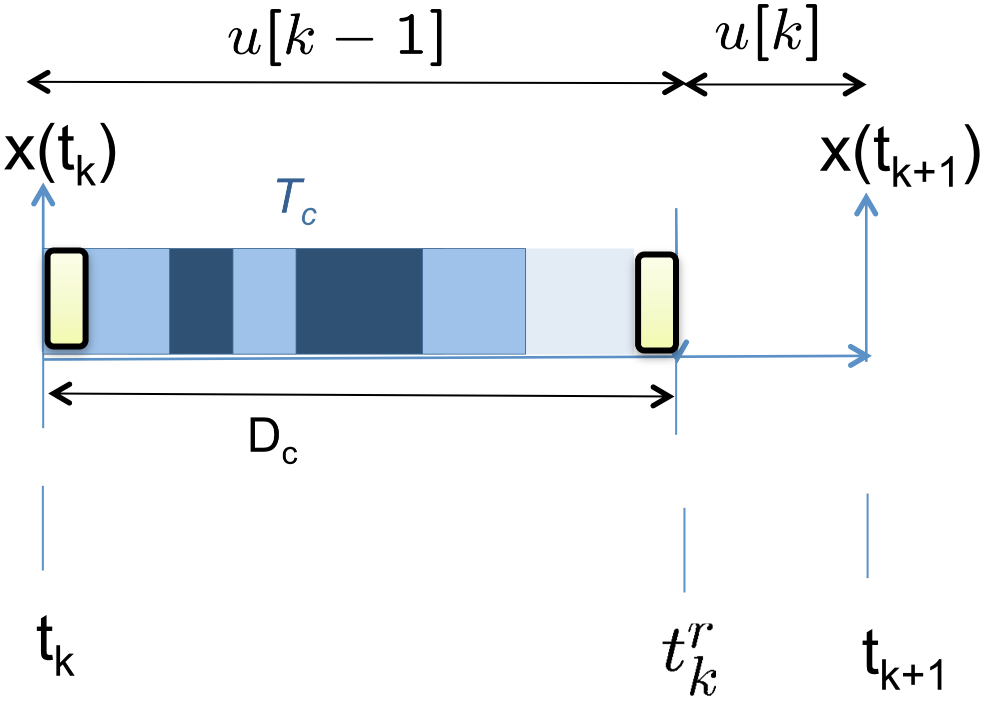
Objectives

(i) Place system poles

(ii) Achieve y → r as t → ∞

Snapshot of One Sampling Period

What happens within one sampling period?



$$\begin{aligned} \dot{x} &= A\ x\ + B\mathsf{u} \\ y &= Cx \end{aligned} \quad \Rightarrow \quad \begin{aligned} x(t) &= e^{At}x(\mathsf{0}) + \textstyle\int_{\mathsf{0}}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \\ y(t) &= Cx(t) \end{aligned}$$

$$x(t_{k+1}) = e^{A(t_{k+1}-t_k)}x(t_k)+\int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bu(\tau)d\tau$$

$$\Downarrow$$

$$\begin{aligned}
&u(\tau) = u[k-1] \text{ for } t_k \leq t \leq t_k^r \\
&u(\tau) = u[k] \text{ for } t_k^r < t \leq t_{k+1} \\
&t_{k+1} - t_k = h \\
&x(t_{k+1}) = x[k+1] \\
&x(t_k) = x[k]
\end{aligned}$$

$$\Downarrow$$

$$\begin{aligned}
x[k+1] = e^{Ah}x[k] + \int_{t_k}^{t_k^r} e^{A(t_{k+1}-\tau)}Bd\tau.\underbrace{u[k-1]}_{} +\\
+ \int_{t_k^r}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bd\tau.\underbrace{u[k]}_{}
\end{aligned}$$

$$x[k+1] = e^{Ah}x[k] + \int_{t_k}^{t_k^r} e^{A(t_{k+1}-\tau)}Bd\tau.u[k-1] +$$
$$+ \int_{t_k^r}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bd\tau.u[k]$$

$$\Downarrow$$

$$x[k+1] = e^{Ah}x[k] + \int_{h-D_c}^{h} e^{As}Bds.u[k-1] +$$
$$\int_0^{h-D_c} e^{As}Bds.u[k]$$

where $s = t_{k+1} - \tau$

$$\Downarrow$$

$$x[k+1] = \phi x[k] + \Gamma_1(D_c)u[k-1] + \Gamma_0(D_c)u[k]$$

$$\phi = e^{Ah}$$
$$\Gamma_1(D_c) = \int_{h-D_c}^{h} e^{As}Bds$$
$$\Gamma_0(D_c) = \int_0^{h-D_c} e^{As}Bds.$$

Sampled-data Model

$$\dot{x} = A\ x\ + B\mathrm{u}$$
$$y = Cx$$

Continuous-time model

⇩

ZOH sampling with period h and constant sensor-to-actuator delay D_c

$$x[k+1] = \phi x[k] + \Gamma_1(D_c)u[k-1] + \Gamma_0(D_c)u[k]$$
$$y[k] = Cx[k]$$

$$\phi = e^{Ah}$$
$$\Gamma_1(D_c) = \int_{h-D_c}^{h} e^{As} B ds$$
$$\Gamma_0(D_c) = \int_{0}^{h-D_c} e^{As} B ds.$$

Sampled-data model

End of Step 1

Augmented System

- We define new system states:

$$z[k] = \begin{bmatrix} x[k] \\ u[k-1] \end{bmatrix}$$

- With the new definition of states, the state-space becomes

$$z[k+1] = \phi_{aug} z[k] + \Gamma_{aug} u[k]$$
$$y[k] = C_{aug} z[k]$$

where the augmented matrices are defined as follows

$$\phi_{aug} = \begin{bmatrix} \phi & \Gamma_1(D_c) \\ 0 & 0 \end{bmatrix}, \ \Gamma_{aug} = \begin{bmatrix} \Gamma_0(D_c) \\ I \end{bmatrix}$$
$$C_{aug} = \begin{bmatrix} C & 0 \end{bmatrix}$$

Controller Design for $D_c < h$

- Given system: $$z[k+1] = \phi_{aug} z[k] + \Gamma_{aug} u[k]$$ $$y[k] = C_{aug} z[k]$$
- Control law: $$u[k] = Kz[k] + Fr$$

Objectives

(i) Place system poles

(ii) Achieve y $\rightarrow$ r as t $\rightarrow \infty$

(iii) Design K and F

$$\Downarrow$$

1. Check controllability of $(\phi_{aug}, \Gamma_{aug})$ → must be controllable. γ must be invertible where γ is defined as follows
$$\gamma_{aug} = \left[\begin{array}{ccccc} \Gamma_{aug} & \phi_{aug}\Gamma_{aug} & \phi_{aug}^2\Gamma_{aug} & \cdots & \phi_{aug}^{n-1}\Gamma_{aug} \end{array}\right]$$
2. Apply Ackermann's formula $K = -\left[\begin{array}{cccc} 0 & 0 & \cdots & 1 \end{array}\right] \gamma_{aug}^{-1} H(\phi_{aug})$
3. Feedforward gain $F = \dfrac{1}{C_{aug}(I - \phi_{aug} - \Gamma_{aug}K)^{-1}\Gamma_{aug}}$

End of Step II

Summary: Design for D_c < h

Continuous-time model

$$\dot{x} = A\ x + Bu$$
$$y = Cx$$

$$\Downarrow$$

Sampled-data model

$$x[k+1] = \phi x[k] + \Gamma_1(D_c)u[k-1] + \Gamma_0(D_c)u[k]$$
$$y[k] = Cx[k]$$

$$\Downarrow$$

Augmented system

$$z[k+1] = \phi_{aug} z[k] + \Gamma_{aug} u[k]$$
$$y[k] = C_{aug} z[k]$$

$$\Downarrow$$

Controller gains

$$u[k] = Kz[k] + Fr$$

$$K = -\begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \gamma_{aug}^{-1} H(\phi_{aug})$$

$$F = \frac{1}{C_{aug}(I - \phi_{aug} - \Gamma_{aug} K)^{-1} \Gamma_{aug}}$$

Computation, Communication and Memory-aware Controller Design

Automotive Communication Buses

- Time-Triggered Bus Protocols
  - **Time-Triggered Protocol (TTP)** – mostly used for reliable/guaranteed communication. Also used in avionics (airplanes)
  - Based on Time Division Multiple Access (TDMA) policy
  - Has two variants TTP/A and TTP/C
    - "A" refers to "Automotive Class A" for soft real-time applications. It is a scaled down version of TTP and is cheaper
    - "C" refers to "Automotive Class C" for hard real-time applications. It is the full version of TTP and offers fault tolerance

- Event-Triggered Bus Protocols
  - **Controller Area Network (CAN)** – widely used for chassis control systems and power train communication
  - Based on fixed priority scheduling policy
  - Does not provide hard real-time guarantees

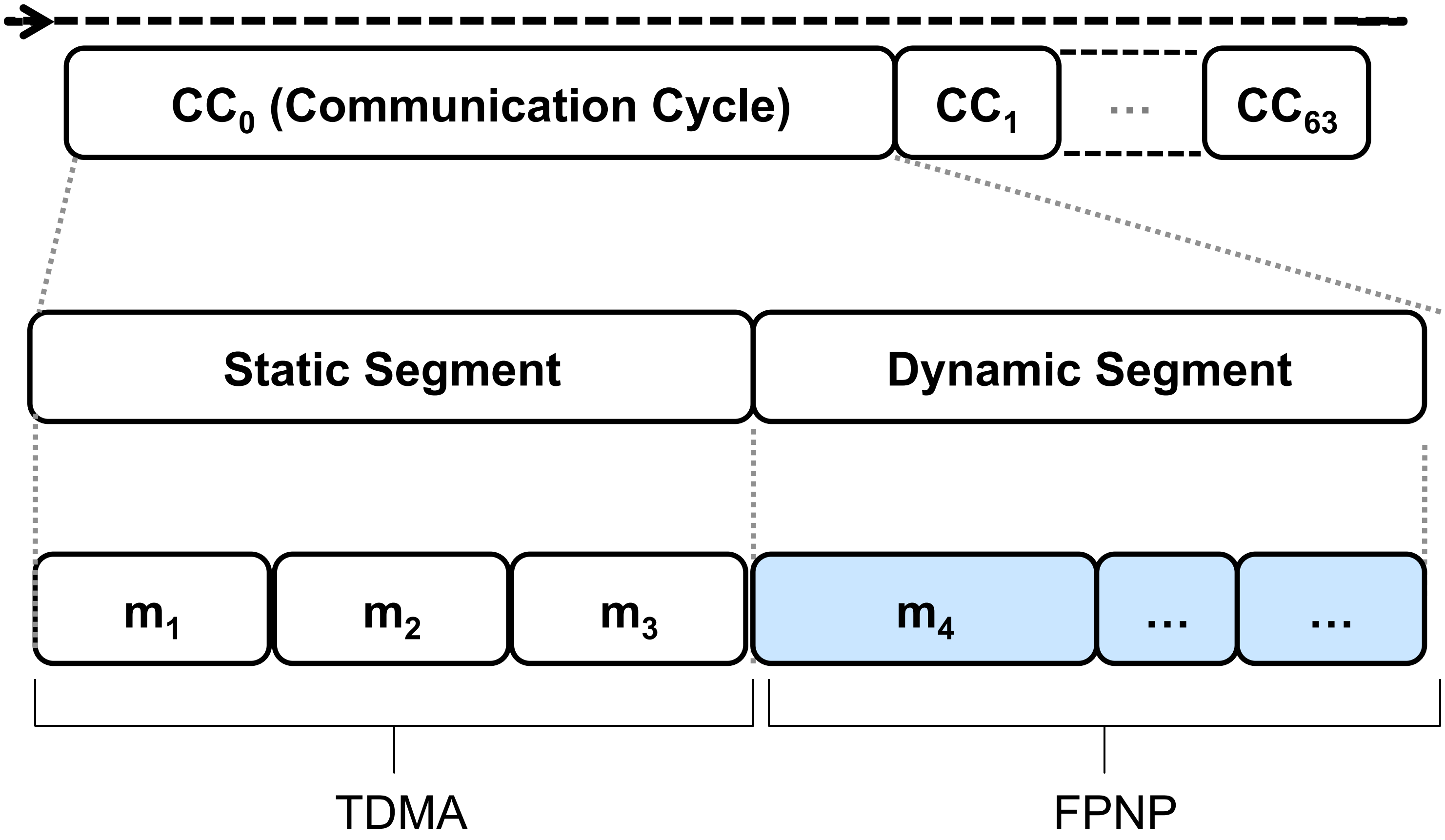
Time-Triggered or Event-Triggered?

	Time-Triggered	Event-Triggered
Timing Guarantees	Deterministic behavior, higher dependability	Difficult to provide hard real-time guarantees
Target Applications	Regular/Periodic	Good performance for asynchronous events
Bus Utilization	Low if applications are not periodic	High
Flexibility	Small change might require full redesign	Flexible and scalable
Composability	Different components can be easily composed	Difficult to provide timing guarantees

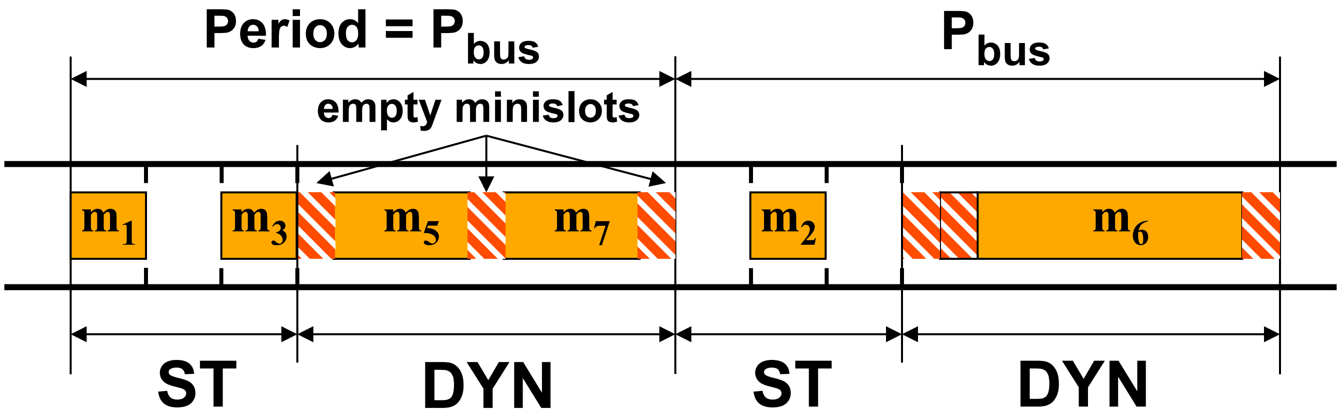
Mix of Time- and Event-Triggered

- The question of Time-Triggered or Event-Triggered is a subject of debate. Each has its own advantages and disadvantages
- This has led to the development of mixed or hybrid protocols which combine the features of both time- and event-triggered paradigms
- Examples
  - **TTCAN** – Time-Triggered CAN, built on top of CAN
  - **FlexRay** – started by DaimlerChrysler and BMW. It is widely believed that this will become the most popular bus protocol in the future

Hybrid Communication (FlexRay)



FlexRay – Brief Overview



- Tasks T_1, …, T_8 send messages over a FlexRay bus
- T_1, T_2, T_3 over the ST segment and T_4, …, T_8 over the DYN segment
- In the first cycle, T_5, T_6 and T_7 have messages to send, but not T_4 and T_8. Message from T_6 did not fit into the DYN segment
- In the second cycle, T_4, T_5 and T_8 had nothing to send. Message from T_7 did not fit into the DYN segment

Communication Schedules

Time-triggered (TT)

- The temporal behavior is predictable
- The bandwidth utilization is poor
- Availability is limited

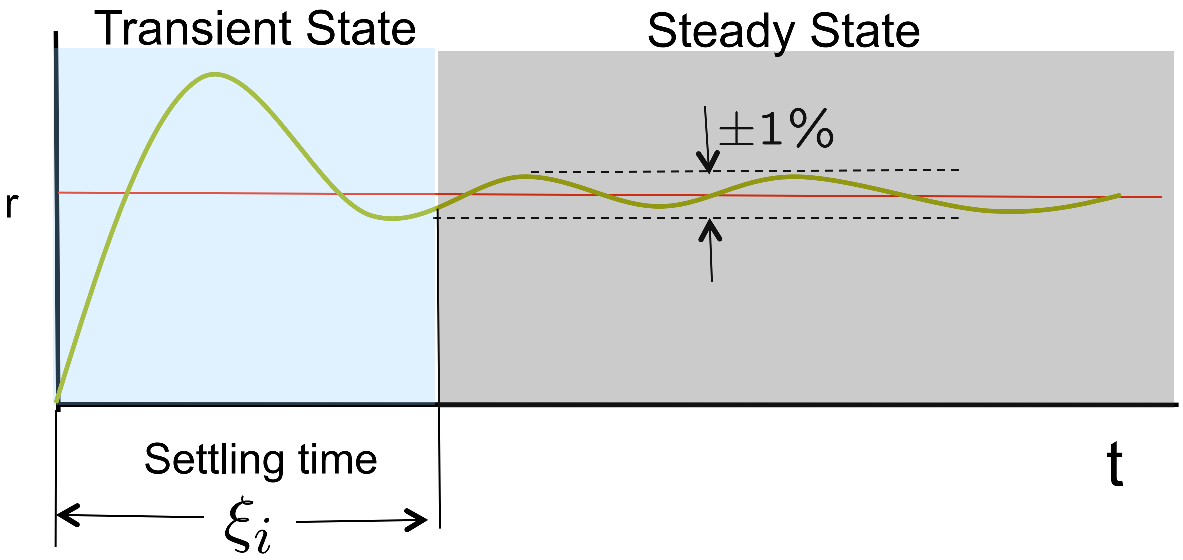
Event-triggered (ET)

- The temporal behavior is unpredictable
- The bandwidth utilization is better
- Availability is higher

Conventional design: Use TT for control-messages

Challenge:
Can we design controllers that use fewer TT slots but still have good control performance?

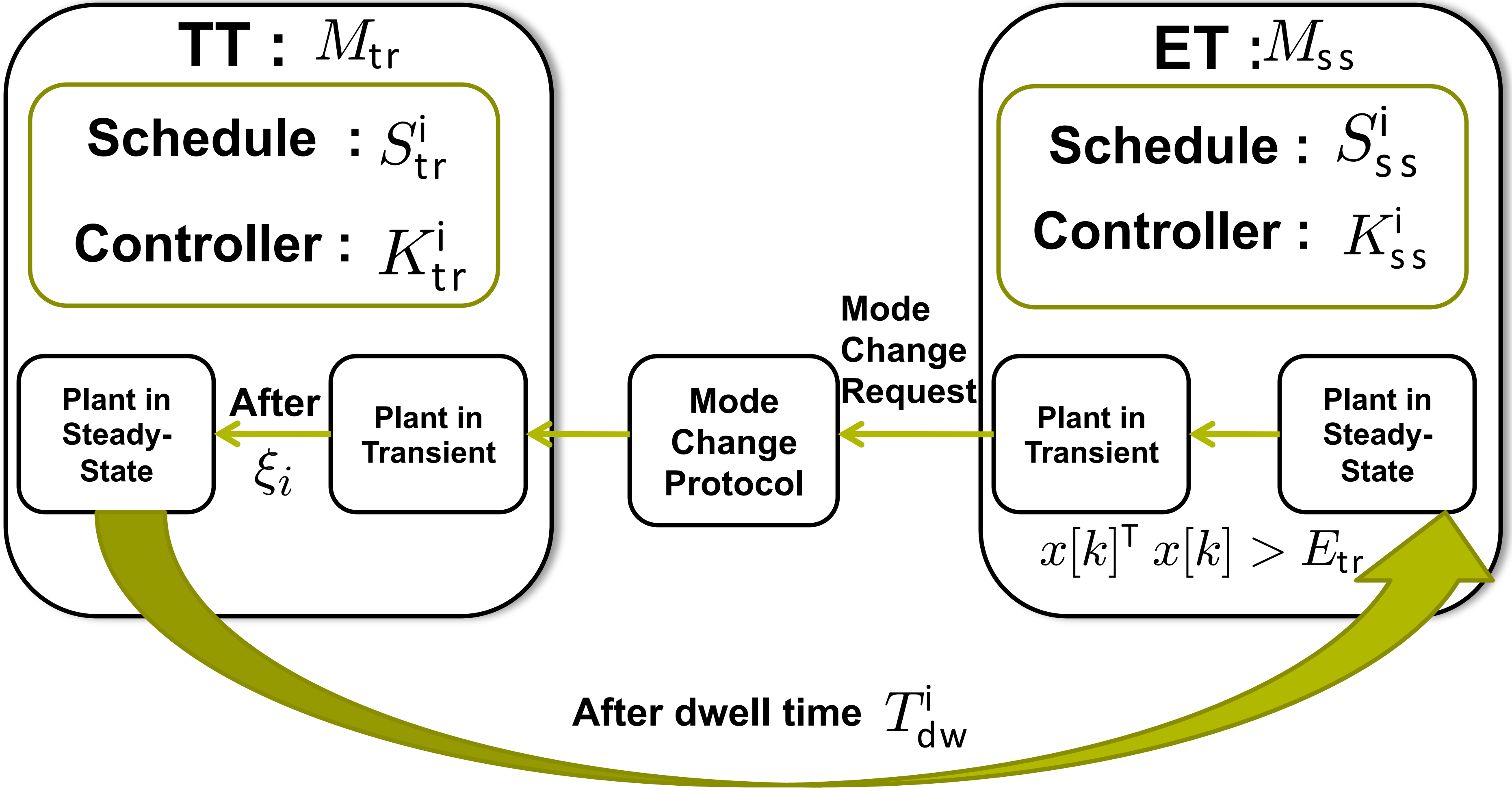
Quality of Control vs. System State



Observations

- The performance of a control application is more sensitive to the applied control input in transient state compared to that in steady-state
- ET communication for the control signals is good enough in the steady-state
- TT communication is better suited for transient state

Mode Switching Scheme


TT : M_{tr}
Schedule : $S^{\mathsf{i}}_{\mathsf{tr}}$
Controller : $K^{\mathsf{i}}_{\mathsf{tr}}$
Plant in Steady-State
After ξ_i
Plant in Transient
Mode Change Protocol
Mode Change Request
ET : M_{ss}
Schedule : $S^{\mathsf{i}}_{\mathsf{ss}}$
Controller : $K^{\mathsf{i}}_{\mathsf{ss}}$
Plant in Transient
Plant in Steady-State
$x[k]^{\mathsf{T}}\, x[k] > E_{\mathsf{tr}}$
After dwell time $T^{\mathsf{i}}_{\mathsf{dw}}$

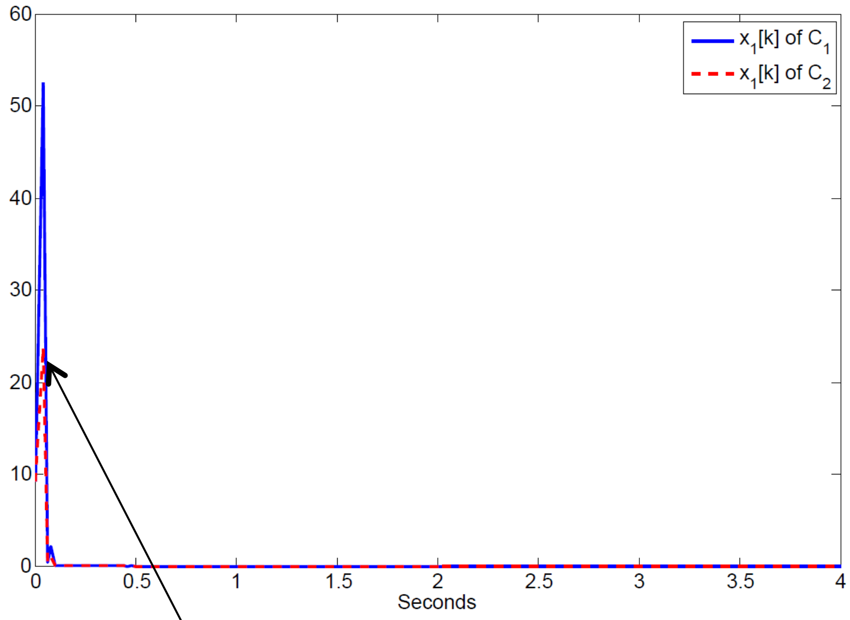
Example

- We consider two distributed control applications communicating via a hybrid communication bus

$$C_1 : x[k+1] = A_1 x[k] + B_1 u[k]$$
$$C_2 : x[k+1] = A_2 x[k] + B_2 u[k]$$
$$A_1 = \begin{bmatrix} 0.4 & 1.0 \\ -1.56 & -0.9 \end{bmatrix}, B_1 = \begin{bmatrix} 0.3 \\ 0.1 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 1.2 & 0.2 \\ -1.8 & -2.1 \end{bmatrix}, B_2 = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}.$$

- We apply state-feedback controller for both, i.e., u[k] =Fx[k]

Performance with TT Communication



Converges very fast without any oscillation

Control Gains

$$F^1_{tr} = \begin{bmatrix} 7.4394 & 2.6819 \end{bmatrix}$$
$$F^2_{tr} = \begin{bmatrix} 0.0417 & 2.9722 \end{bmatrix}$$

Quality of control

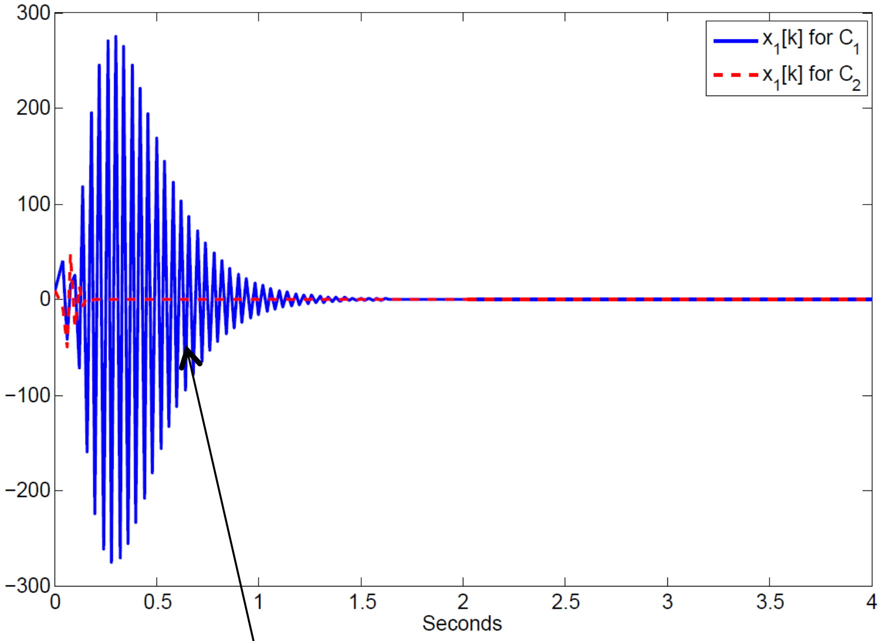
C$_1$

$$\xi_1 = 0.14 \text{ Sec.}$$
$$\sum_k x[k]^T x[k] = 8.6 \times 10^3$$
$$\sum_k u[k]^2 = 4.7276 \times 10^4.$$

C$_2$

$$\xi_2 = 0.14 \text{ Sec}$$
$$\sum_k x[k]^T x[k] = 1.8136 \times 10^3$$
$$\sum_k u[k]^2 = 1.2857 \times 10^4.$$

Performance with ET Communication



Large oscillations and long settling time

Control Gains

$$F_{ss}^1 = \begin{bmatrix} 3.4674 & 2.7978 \end{bmatrix}$$

$$F_{ss}^2 = \begin{bmatrix} -6.1031 & -4.0312 \end{bmatrix}$$

Quality of control

C_1

$$\xi_1 = 2.42 \text{ Sec.}$$

$$\sum_k x[k]^T x[k] = 6.9648 \times 10^6$$

$$\sum_k u[k]^2 = 9.1703 \times 10^6.$$

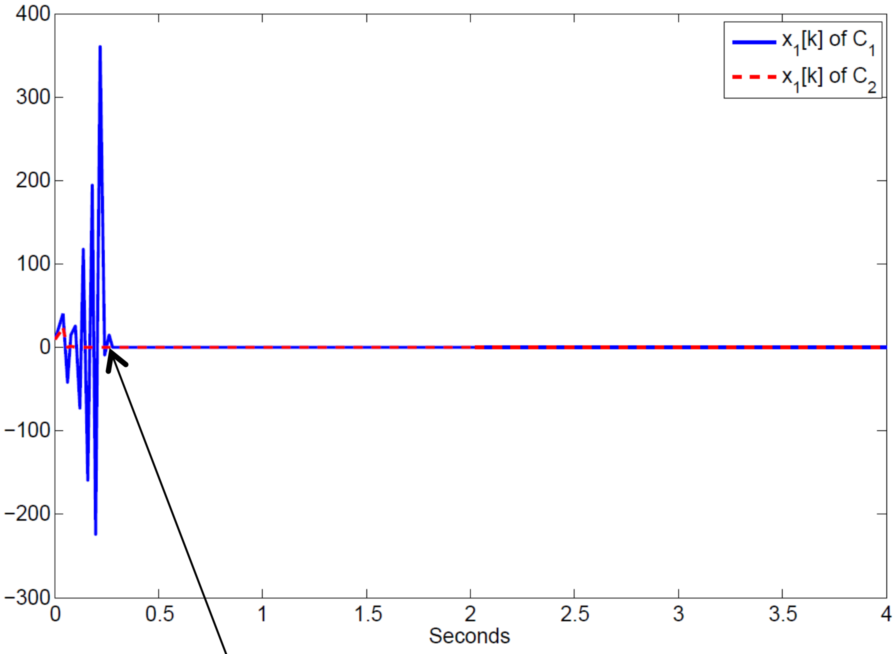
C_2

$$\xi_2 = 0.28 \text{ Sec}$$

$$\sum_k x[k]^T x[k] = 5.4479 \times 10^4$$

$$\sum_k u[k]^2 = 3.0933 \times 10^5.$$

Performance with Switching



We have one shared TT communication slot. The control messages are transmitted via ET communication when they are in steady state and switches to TT communication when transient state occurs due to some disturbance

Quality of control

C1

$$\xi_1 = 0.36 \text{ Sec}$$

$$\sum_k x[k]^T x[k] = 1.3651 \times 10^6$$

$$\sum_k u[k]^2 = 2.3607 \times 10^6.$$

C2

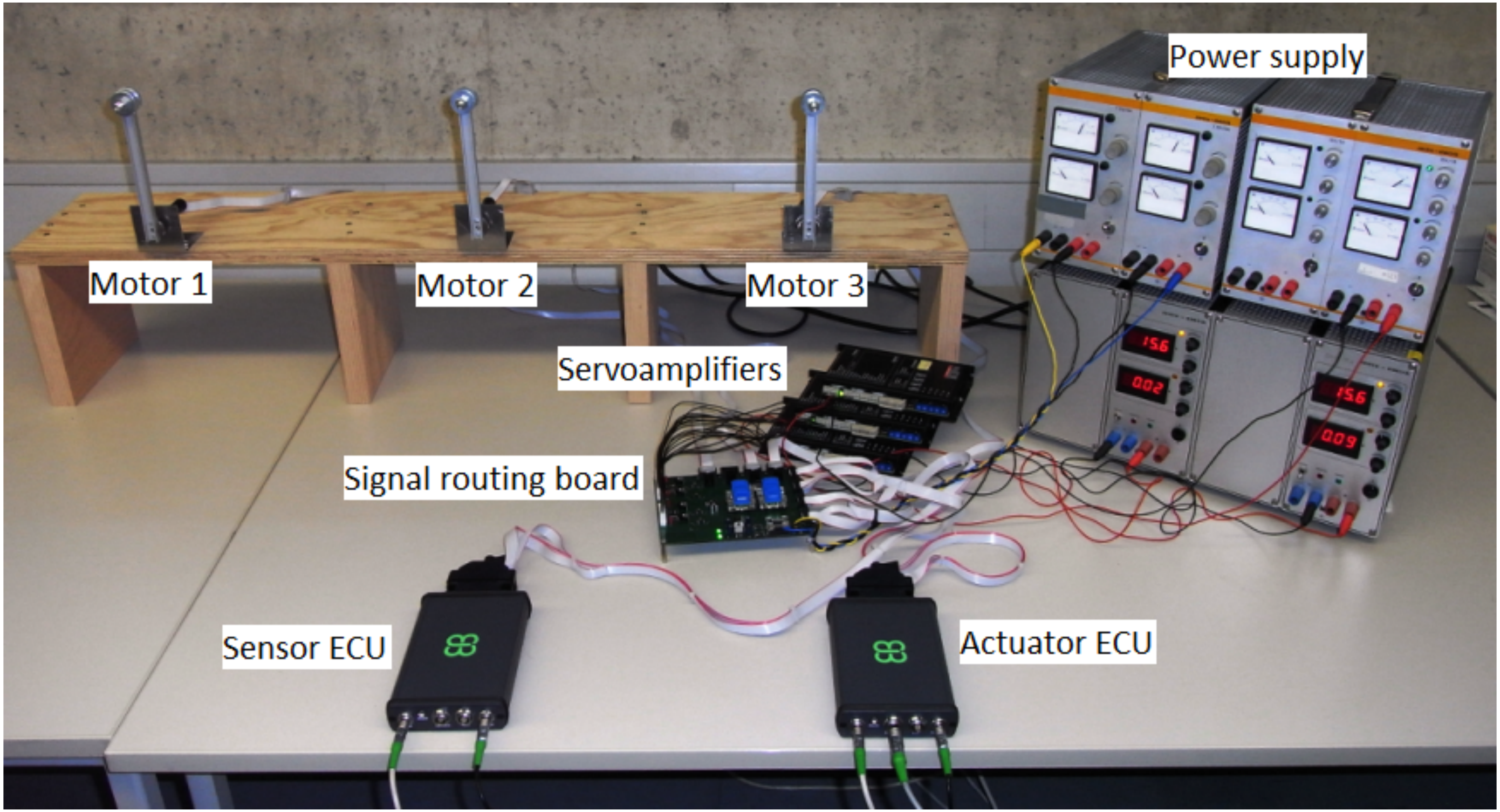
$$\xi_2 = 0.14 \text{ Sec}$$

$$\sum_k x[k]^T x[k] = 1.8136 \times 10^3$$

$$\sum_k u[k]^2 = 1.2857 \times 10^4.$$

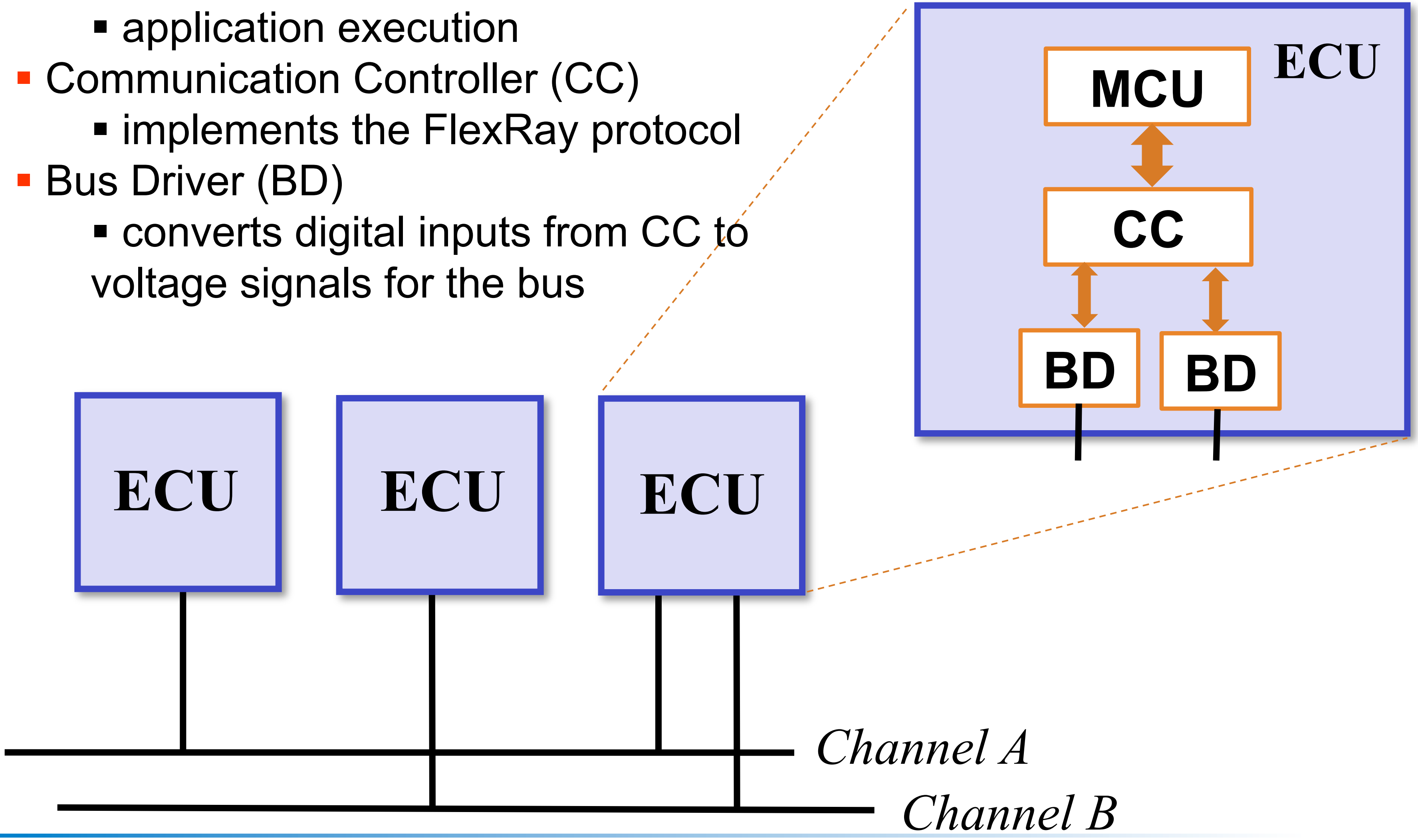
Performance is better than that with ET communication but we consume less TT communication slots

Experimental Setup

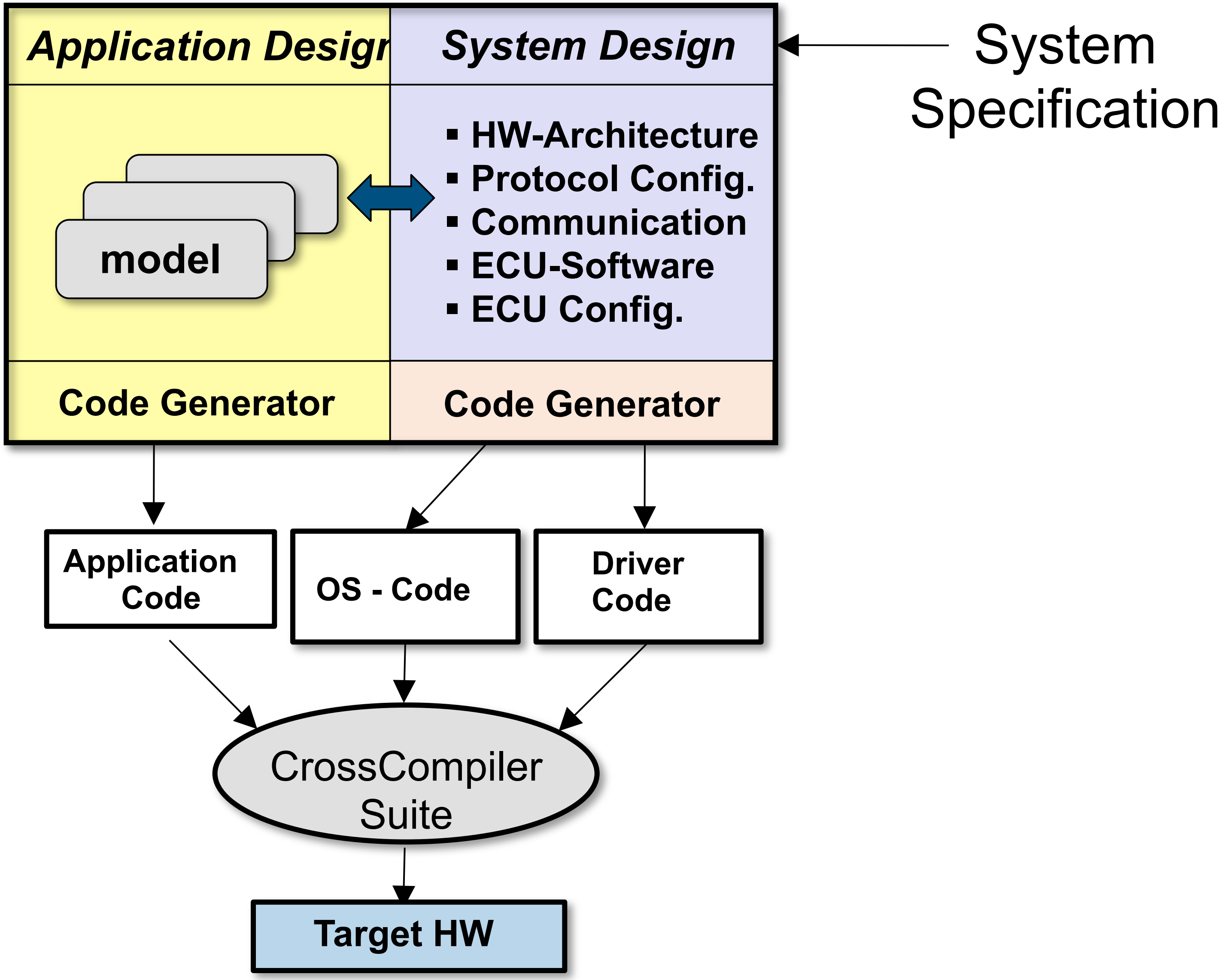


Design Flow

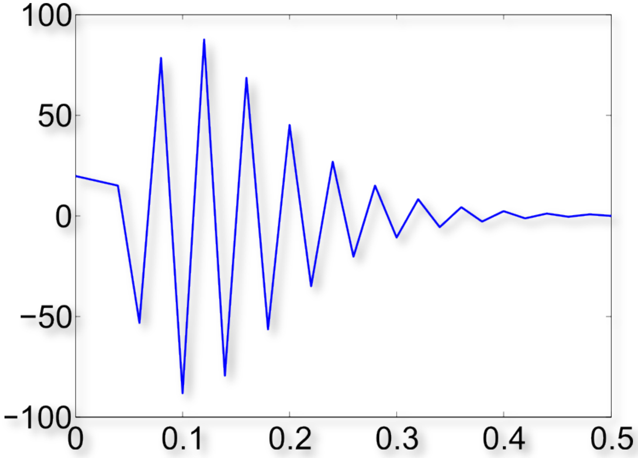
- Microcontroller Unit (MCU)
  - application execution
- Communication Controller (CC)
  - implements the FlexRay protocol
- Bus Driver (BD)
  - converts digital inputs from CC to voltage signals for the bus



ECU Software Development

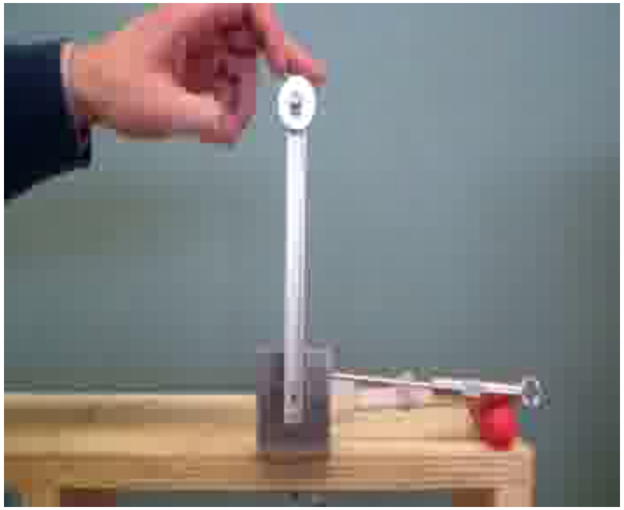


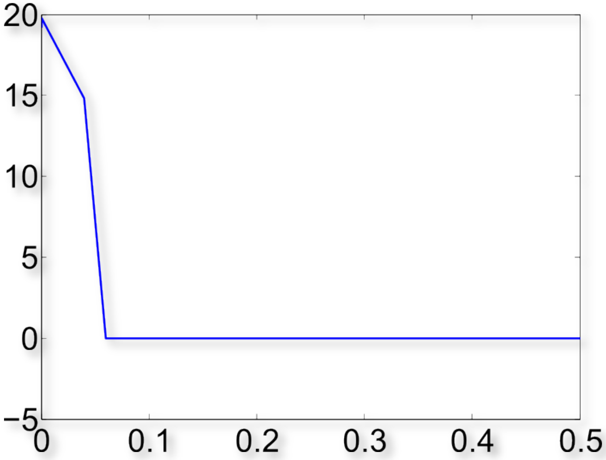
Experimental Results



Purely event-triggered

Mixed time-/event-triggered

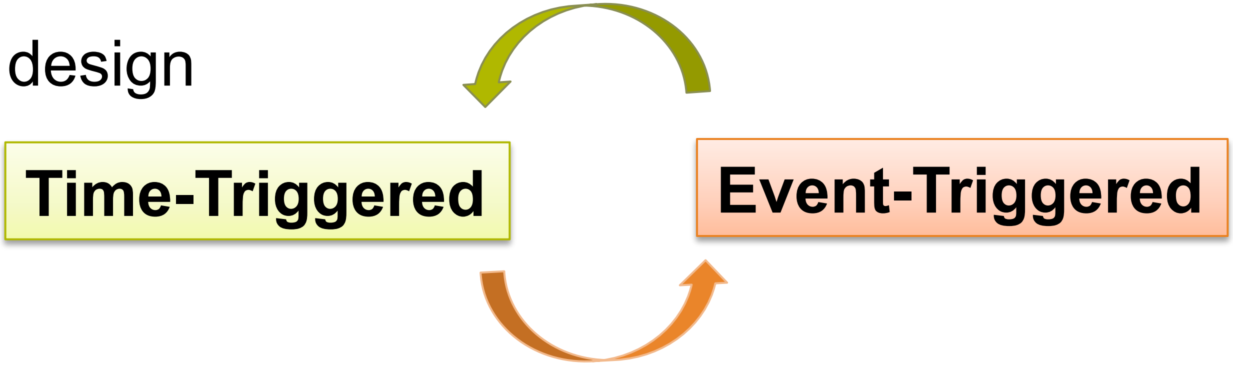




Purely time-triggered

Issues …

- What is the disturbance model?
- How many time-triggered slots?
- How many switches?
- Controller design



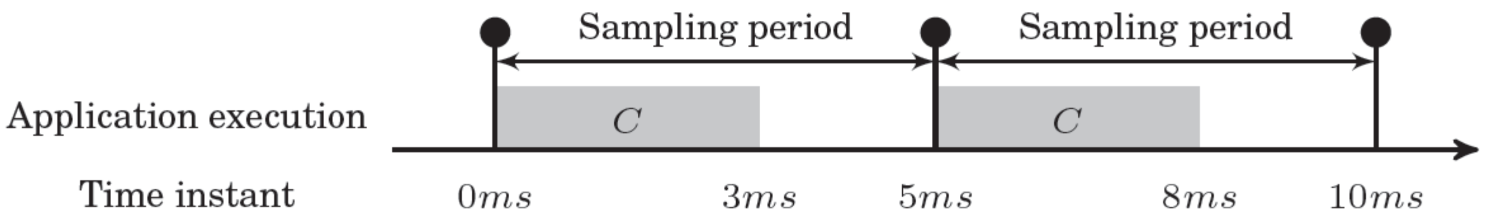
Time-Triggered

Event-Triggered

- Engineering issues: protocol constraints

Computation-aware Controller Design

Example



- Consider a control task that has a sampling period of 5 ms and execution time of 3 ms
- This implies that only one such task can be implemented on a processor

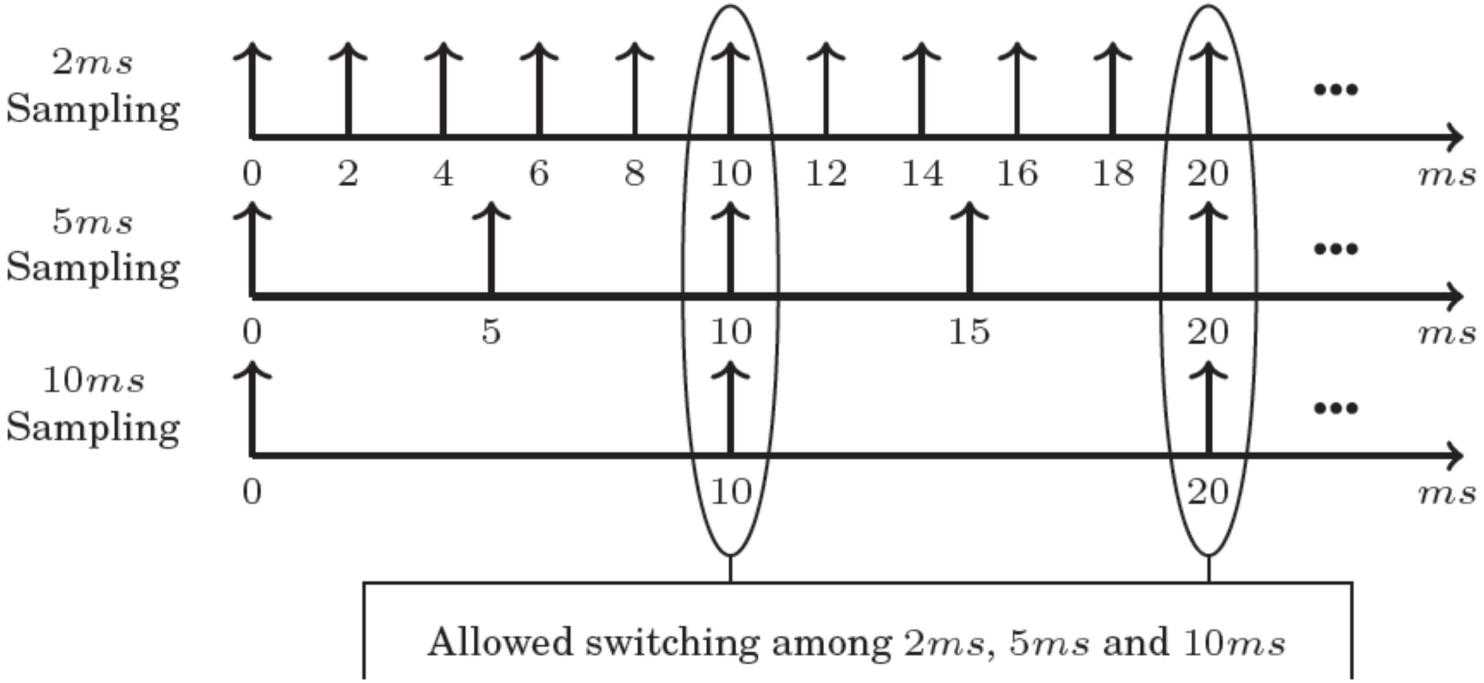
Example - OSEK/VDX Operating System

- Often the operating system is configured to support only a fixed set of sampling periods
- For a control application, if the required sampling period is not offered by the operating system then a smaller sampling period has to be used
- But this leads to poor utilization of the processor

Example

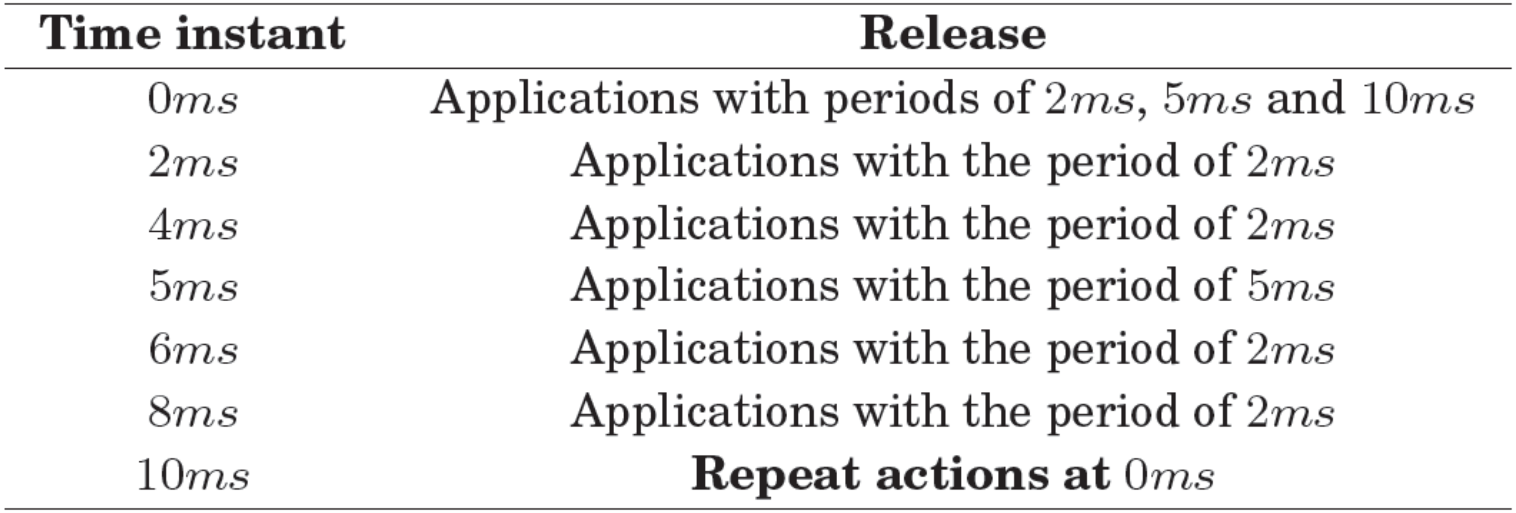
- Again consider the control task that was previously sampled at 5ms
- Instead, with the schedule **{5ms, 5ms, 10ms, repeat}** the average sampling period is 6.67ms and this might be an acceptable sampling period, while 10ms might not be acceptable
- Now with such a **non-uniform sampling schedule**, two control tasks can be implemented on the same processor, whereas with a sampling period of 5ms only one task can be implemented
- Questions: (i) How to design controllers that use such non-uniform sampling? (ii) How to design such schedules?

Switching between multiple sampling periods



- The switching between different sampling period are only allowed at intervals of 10 ms
- Schedule design is an optimization problem

OSEK/VDX



Time instant	**Release**
$0ms$	Applications with periods of $2ms$, $5ms$ and $10ms$
$2ms$	Applications with the period of $2ms$
$4ms$	Applications with the period of $2ms$
$5ms$	Applications with the period of $5ms$
$6ms$	Applications with the period of $2ms$
$8ms$	Applications with the period of $2ms$
$10ms$	**Repeat actions at** $0ms$

Release times of different applications with different sampling periods

Designing controllers with non-uniform sampling periods

System Model

- The plant dynamics is given by:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t),$$
$$y(t) = \mathbf{C}\mathbf{x}(t),$$

  where $\mathbf{x}(t) \in \mathbb{R}^n$ is the system state $y(t)$ is the system output, and $u(t)$ is the control input applied to the system

- Assuming a sampling period of h, the sampled system states are $\mathbf{x}[k] = \mathbf{x}(t_k), \;\; t_k = kh, \;\; k = 0, 1, 2, 3, \cdots$
- The sampled outputs are $y[k] = y(t_k)$

System Model (contd.)

- The discrete values of the control input are similarly denoted by $u[k]$

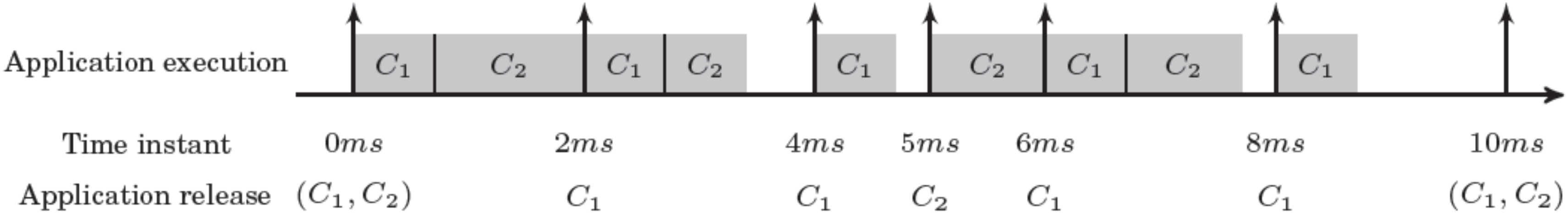
- Using zero-order hold (ZOH), the input applied to the plant is $u(t) = u[k], \quad t_k \leq t < t_{k+1}$
- Hence, the discrete dynamics of the system are given by

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}_d \mathbf{x}[k] + \mathbf{B}_d u[k], \\ y[k] &= \mathbf{C} x[k], \end{aligned}$$

where $\mathbf{A}_d = e^{\mathbf{A}h}, \; \mathbf{B}_d = \int_0^h (e^{\mathbf{A}\tau'} d\tau')\mathbf{B}$

Example

- Consider two applications with C_1 and C_2 that are sharing a single ECU
- C_1 has a period of 2 *ms* and an execution time of 0.7 *ms*
- C_2 has a period of 5 *ms* and an execution time of 2 *ms*
- Assume that they are scheduled using a preemptive fixed priority scheduling policy with rate monotonic priority assignments



New system model

- To cope with the variations in task completion times, we assume that the actuation is done at the end of the sampling period
- Hence, the resulting system model is:

$$\mathbf{x}[k+1] = \mathbf{A}_d \mathbf{x}[k] + \mathbf{B}_d u[k-1]$$

Controller with non-uniform sampling

- Let the operating system offer a set of sampling periods ϕ
- A control application uses a sequence of sampling period given by $S = \{T_1, T_2, T_3, \ldots, T_N\}$

  where $\forall j \in \{1, 2, \ldots, N\},\ T_j \in \phi$

- Hence, the schedule of sampling periods used by the controller is given by

$$T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_N \rightarrow T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_N \rightarrow repeat$$

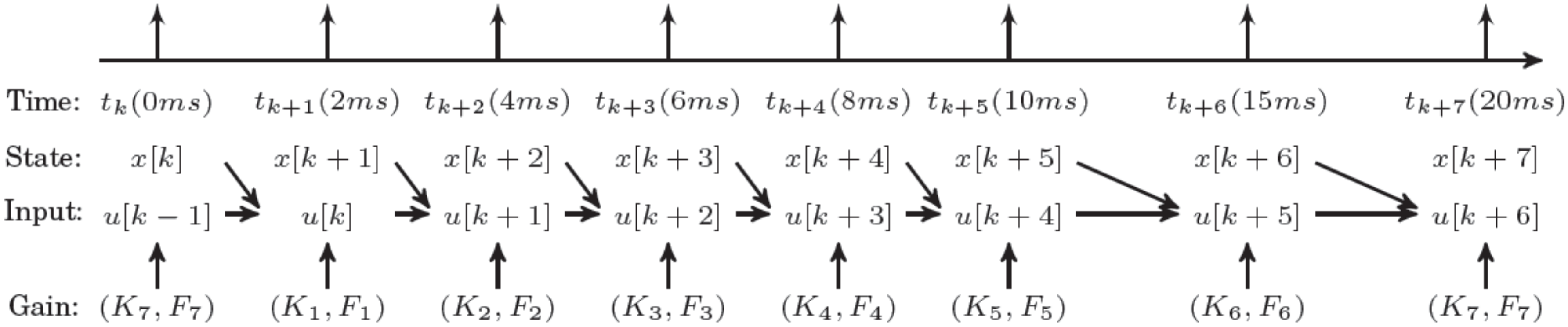
- The resulting load on the processor is $L_i = \frac{N e_i}{\sum_{j=1}^{N} T_j}.$

  where e_i is the execution time of the controller

Resulting system dynamics

- Dynamics of the resulting system within one cycle of *S* is given by:

$$
\begin{aligned}
\mathbf{x}[k+1] &= \mathbf{A}_d(T_1)\mathbf{x}[k] + \mathbf{B}_d(T_1)u[k-1],\\
\mathbf{x}[k+2] &= \mathbf{A}_d(T_2)\mathbf{x}[k+1] + \mathbf{B}_d(T_2)u[k],\\
&\vdots\\
\mathbf{x}[k+N] &= \mathbf{A}_d(T_N)\mathbf{x}[k+N-1] + \mathbf{B}_d(T_N)u[k+N-2].
\end{aligned}
$$



Time:	$t_k(0ms)$	$t_{k+1}(2ms)$	$t_{k+2}(4ms)$	$t_{k+3}(6ms)$	$t_{k+4}(8ms)$	$t_{k+5}(10ms)$	$t_{k+6}(15ms)$	$t_{k+7}(20ms)$
State:	$x[k]$	$x[k+1]$	$x[k+2]$	$x[k+3]$	$x[k+4]$	$x[k+5]$	$x[k+6]$	$x[k+7]$
Input:	$u[k-1]$	$u[k]$	$u[k+1]$	$u[k+2]$	$u[k+3]$	$u[k+4]$	$u[k+5]$	$u[k+6]$
Gain:	(K_7, F_7)	(K_1, F_1)	(K_2, F_2)	(K_3, F_3)	(K_4, F_4)	(K_5, F_5)	(K_6, F_6)	(K_7, F_7)

$$S^0 = \{2ms, 2ms, 2ms, 2ms, 2ms, 5ms, 5ms\}$$

Resulting system dynamics

- Let us introduce a new augmented system state

$$\mathbf{z}[k] = [\, \mathbf{x}[k] \;\; u[k-1] \,]^T$$

- Then for $\forall j \in \{1, 2, \ldots, N\}$ we have

$$\mathbf{z}[k+j] = \begin{bmatrix} \mathbf{A}_d(T_j) & \mathbf{B}_d(T_j) \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{z}[k+j-1] + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} u[k+j-1]$$

  where $\mathbf{0}$ is a zero vector

- The system and input matrices for the augmented state are

$$\mathbf{A}_{aug}(T_j) = \begin{bmatrix} \mathbf{A}_d(T_j) & \mathbf{B}_d(T_j) \\ \mathbf{0} & 0 \end{bmatrix}, \; \mathbf{B}_{aug}(T_j) = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$$

Resulting system dynamics

- The system output is

$$y[k+j-1] = \mathbf{C}_{aug}\mathbf{z}[k+j-1]$$

where $\mathbf{C}_{aug} = [\,\mathbf{C} \;\; 0\,]$

- The control input is designed as

$$u[k+j-1] = \mathbf{K}_j\mathbf{z}[k+j-1] + F_j r$$

Resulting system dynamics

- Hence, the closed loop dynamics of the system is given by

$$\begin{aligned}\mathbf{z}[k+j] &= \mathbf{A}_{aug}(T_j)\mathbf{z}[k+j-1] + \mathbf{B}_{aug}(T_j)u[k+j-1] \\ &= (\mathbf{A}_{aug}(T_j) + \mathbf{B}_{aug}(T_j)\mathbf{K}_j)\mathbf{z}[k+j-1] + \mathbf{B}_{aug}(T_j)F_j r\end{aligned}$$

- The closed loop system matrix may be denoted as

$$\mathbf{A}_{cl,j} = \mathbf{A}_{aug}(T_j) + \mathbf{B}_{aug}(T_j)\mathbf{K}_j$$

Resulting system dynamics

- Hence, the overall system dynamics in one cycle for a schedule $S^0 = \{2ms, 2ms, 2ms, 2ms, 2ms, 5ms, 5ms\}$ is given by

$$
\begin{aligned}
\mathbf{z}[k+7] &= \mathbf{A}_{cl,7}\mathbf{z}[k+6] + \mathbf{B}_{aug}(T_7 = 5ms)F_7 r \\
&= \mathbf{A}_{cl,7}(\mathbf{A}_{cl,6}\mathbf{z}[k+5] + \mathbf{B}_{aug}(T_6 = 5ms)F_6 r) + \mathbf{B}_{aug}(T_7 = 5ms)F_7 r \\
&= \mathbf{A}_{cl,7}\mathbf{A}_{cl,6}\mathbf{z}[k+5] + \mathbf{A}_{cl,7}\mathbf{B}_{aug}(T_6 = 5ms)F_6 r + \mathbf{B}_{aug}(T_7 = 5ms)F_7 r \\
&= \mathbf{A}_{cl,7}\mathbf{A}_{cl,6}(\mathbf{A}_{cl,5}\mathbf{z}[k+4] + \mathbf{B}_{aug}(T_5 = 2ms)F_5 r) \\
&+ \mathbf{A}_{cl,7}\mathbf{B}_{aug}(T_6 = 5ms)F_6 r + \mathbf{B}_{aug}(T_7 = 5ms)F_7 r \\
&= \mathbf{A}_{cl,7}\mathbf{A}_{cl,6}\mathbf{A}_{cl,5}\mathbf{z}[k+4] + \mathbf{A}_{cl,7}\mathbf{A}_{cl,6}\mathbf{B}_{aug}(T_5 = 2ms)F_5 r \\
&+ \mathbf{A}_{cl,7}\mathbf{B}_{aug}(T_6 = 5ms)F_6 r + \mathbf{B}_{aug}(T_7 = 5ms)F_7 r \\
&\vdots
\end{aligned}
$$

Resulting system dynamics

$$
\begin{aligned}
&\mathbf{z}[k+7] \\
&= \prod_{j=1}^{7} \mathbf{A}_{cl,j}\mathbf{z}[k] + \prod_{j=2}^{7} \mathbf{A}_{cl,j}\mathbf{B}_{aug}(2ms)F_1 r + \prod_{j=3}^{7} \mathbf{A}_{cl,j}\mathbf{B}_{aug}(2ms)F_2 r \\
&+ \prod_{j=4}^{7} \mathbf{A}_{cl,j}\mathbf{B}_{aug}(2ms)F_3 r + \prod_{j=5}^{7} \mathbf{A}_{cl,j}\mathbf{B}_{aug}(2ms)F_4 r \\
&+ \prod_{j=6}^{7} \mathbf{A}_{cl,j}\mathbf{B}_{aug}(2ms)F_5 r + \mathbf{A}_{cl,7}\mathbf{B}_{aug}(5ms)F_6 r + \mathbf{B}_{aug}(5ms)F_7 r.
\end{aligned}
$$

Controller design

- The poles to place are the eigenvalues of $\mathbf{A}_{cl,j}$
- The number of poles are (n+1)N
- To ensure stability, the eigenvalues of the overall closed-loop system matrix

$$\prod_{j=1}^{7} \mathbf{A}_{cl,j}$$

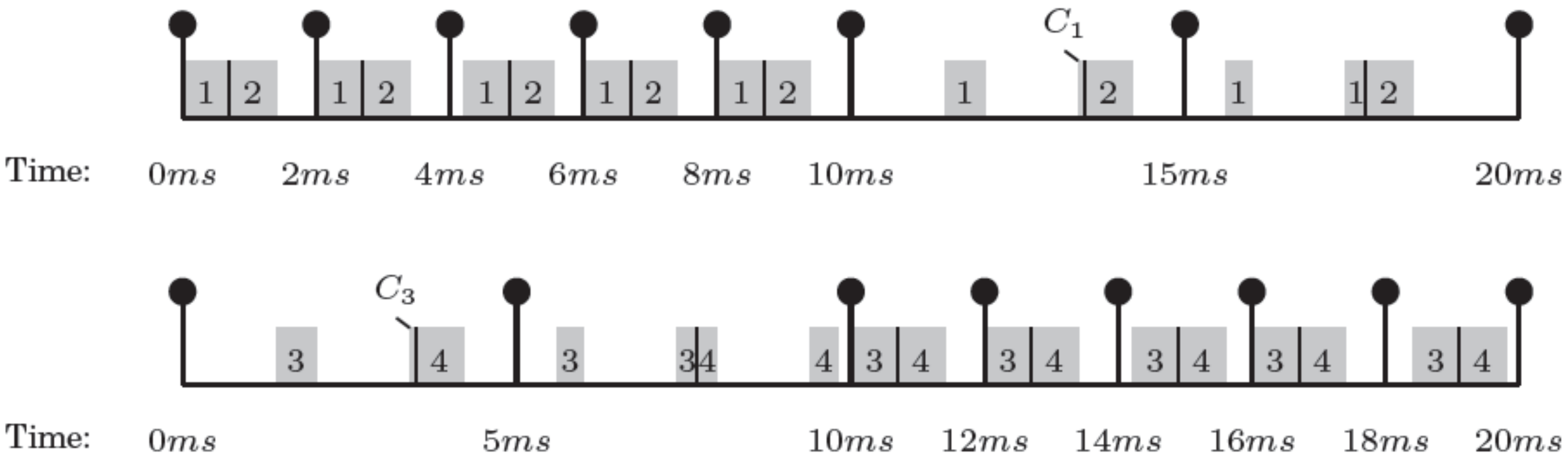
must have absolute values of less than unity

- Once the poles are chosen, the feedback and feedforward gains can be determined in the usual way (as discussed for the earlier problems)

Pole placement

- Choosing the poles involves solving a complex optimization problem, taking into account constraints like input saturation and settling time

Example



- Execution time of each application is 0.7ms
- Schedule for C_1 and C_2 is

$$\{2ms, 2ms, 2ms, 2ms, 2ms, 5ms, 5ms\}$$

- Schedule for C_3 and C_4 is

$$\{5ms, 5ms, 2ms, 2ms, 2ms, 2ms, 2ms\}$$

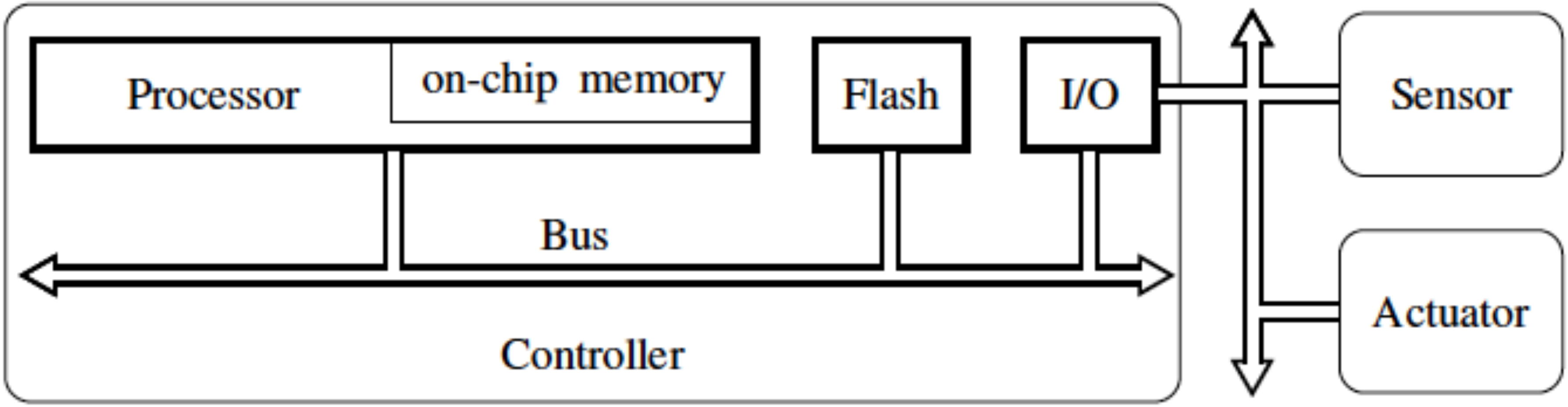
Schedule/controller co-synthesis

- Given a set of plants, how to synthesize the controllers and a schedule such that control objectives are satisfied and the maximum number of controllers can be packed into a single processor
- Since there are non-convex and non-linear optimization problems, heuristic optimization techniques are needed
- While they may perform well in practice, there are no optimality guarantees

Memory Aware Controller Design

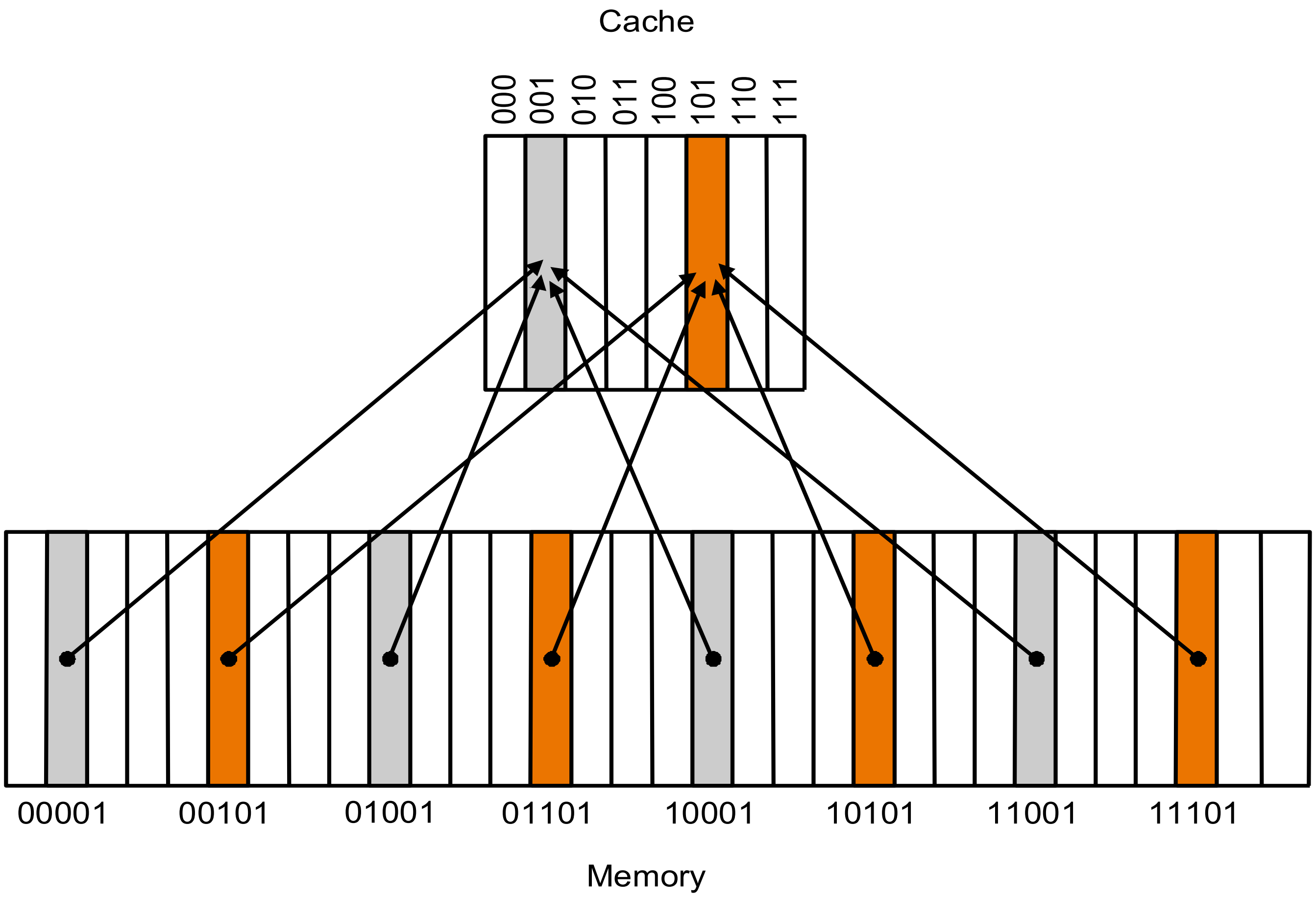
- System setup:
  - Processor executing multiple control applications
  - These applications are on a flash memory and are fetched by the processor one after the other

- Schedule is given as: $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \cdots)$



How does a Cache Work? Direct Mapped Cache

- Mapping: address is modulo the number of blocks in the cache



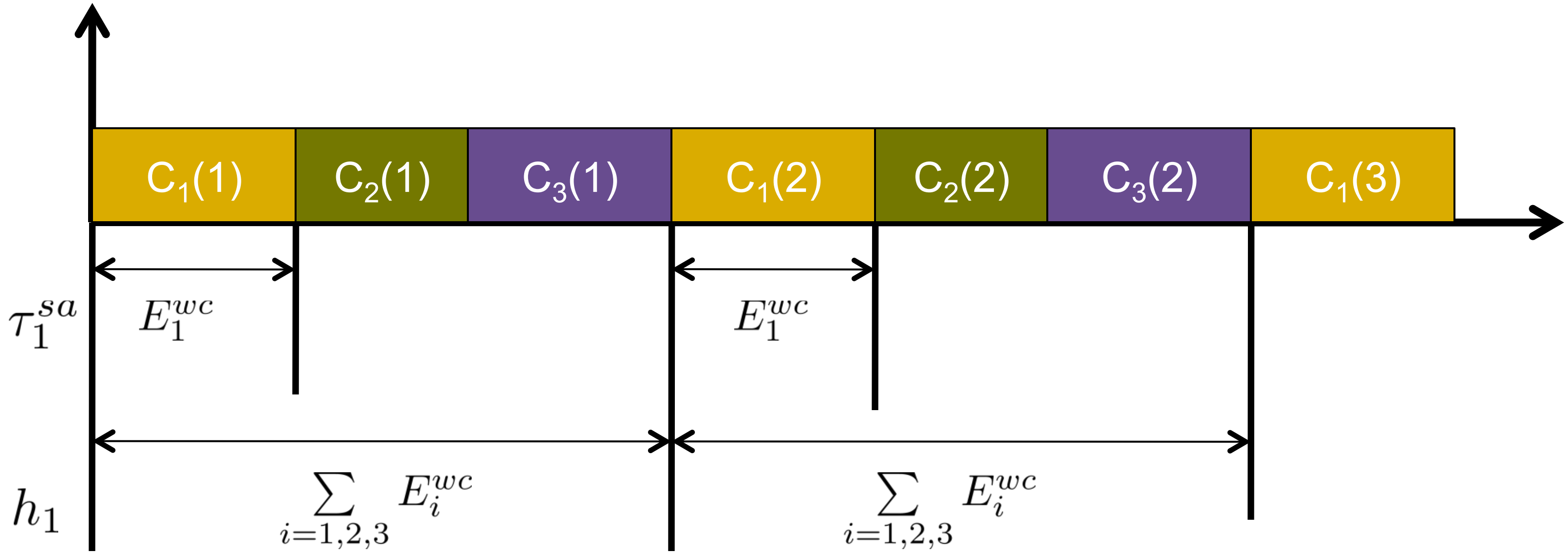
Cache Misses

- This results in each control application evicting the code of the previous application from the on-chip memory (cache)

- Hence, each application experiences a larger execution time (resulting from the code having to be fetched from the flash memory)

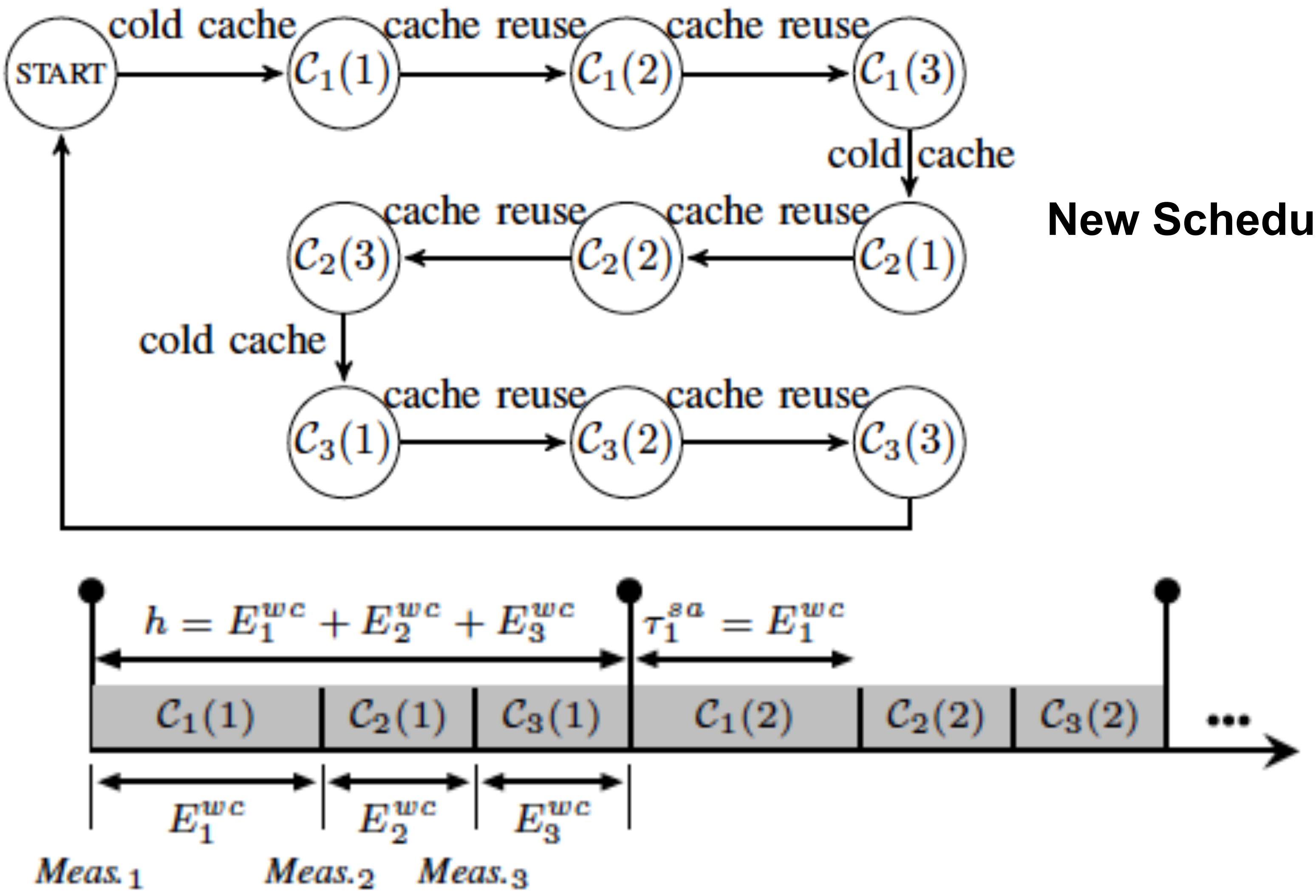
- This increases the sampling period of each application

Controller design for memory oblivious schedule



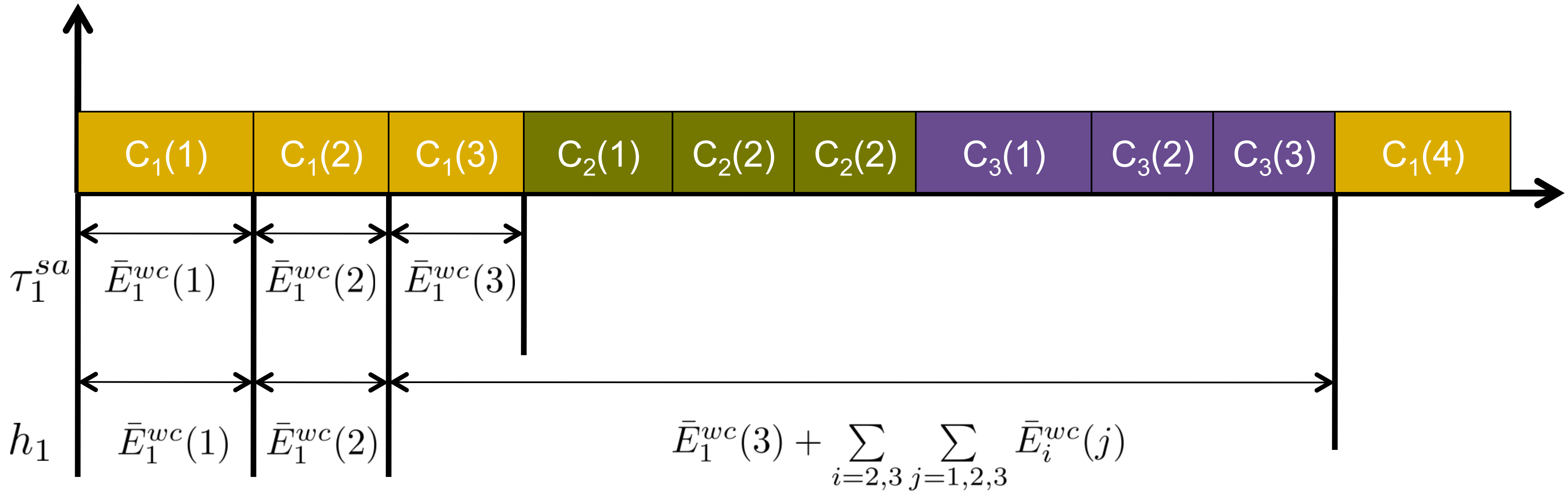
- Average Sampling period $h_{avg} = h_1 = h_2 = h_3 = \sum_{i=1,2,3} E_i^{wc}$
- Sensor-to-actuator delay $\tau_i^{sa} < h_i$
- Discrete-time Controller Design for D_c<h case

Memory-aware Controller Design



New Schedule

Controller design for memory aware schedule



- Sensor-to-actuator delay reduces for second and third instances.

$\tau_i^{sa}(1) = \bar{E}_i^{wc}(1) = E_i^{wc}$, $\tau_i^{sa}(2) = \tau_i^{sa}(3) = \bar{E}_i^{wc}(2) = \bar{E}_i^{wc}(3) = E_i^{wc} - \bar{E}_i^{g}$,

$\bar{E}_i^{g} =$ is the WCET reduction for memory aware schedule.

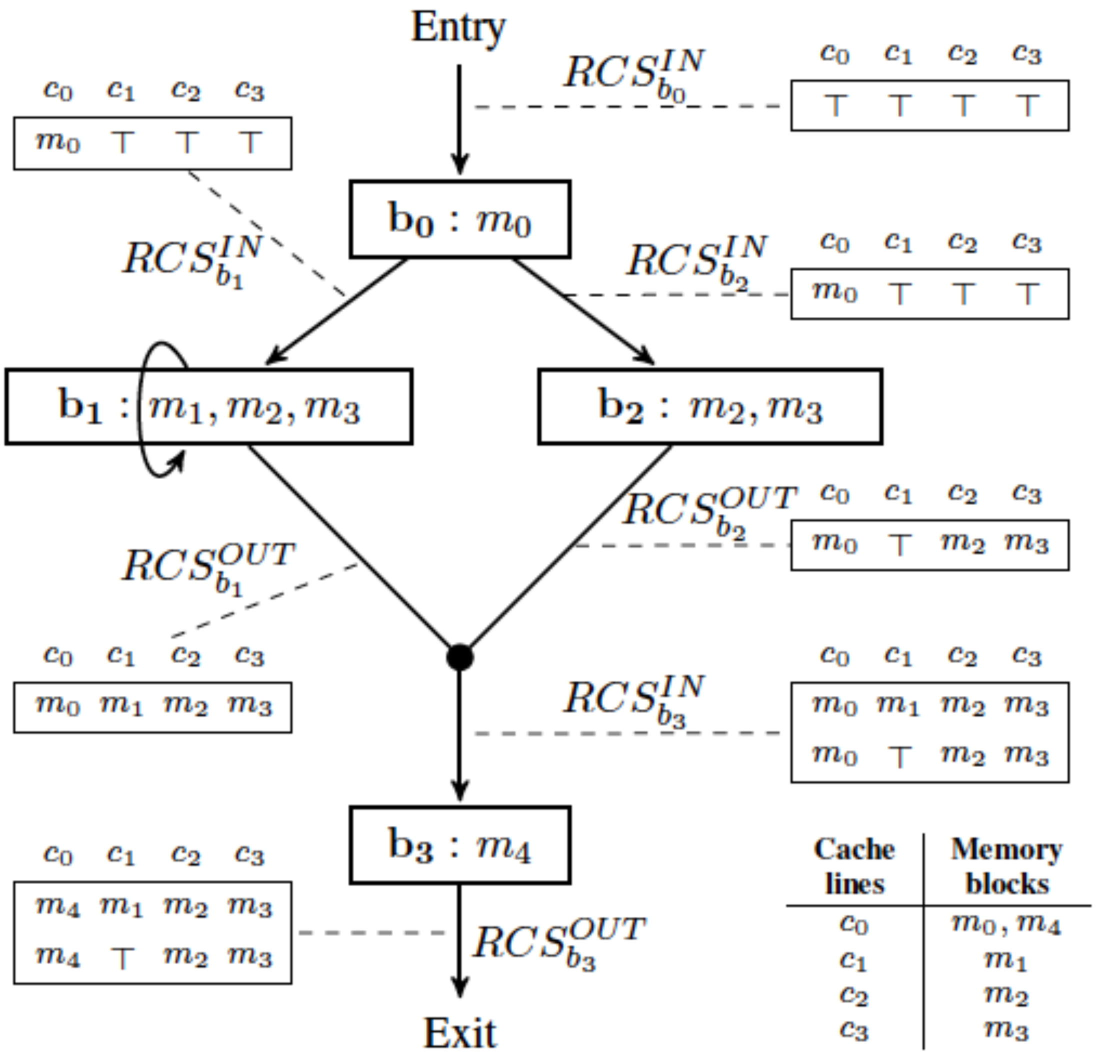
- Average sampling period reduces.

$$\bar{h}_{avg} = \frac{h_i(1)+h_i(2)+h_i(3)}{3} = \frac{\sum\limits_{i=1}^{3}\sum\limits_{j=1}^{3} \bar{E}_i^{wc}(j)}{3} < \frac{3\cdot\sum\limits_{i=1}^{3} E_i^{wc}}{3} < h_{avg}$$

Design Problem

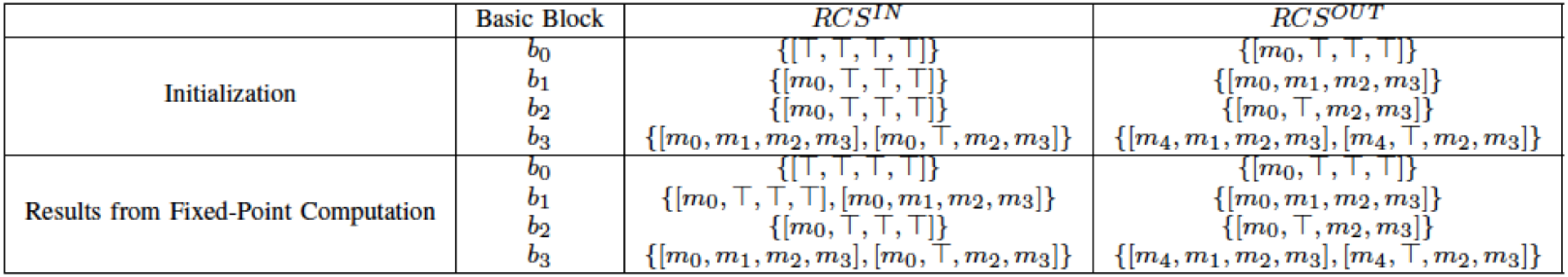
- Consists of two problems
  - How to estimate the guaranteed reduction in worst case execution time?
    - Needs program analysis techniques
  - How to do controller design for non-uniformly sampled systems?

Program Analysis Technique

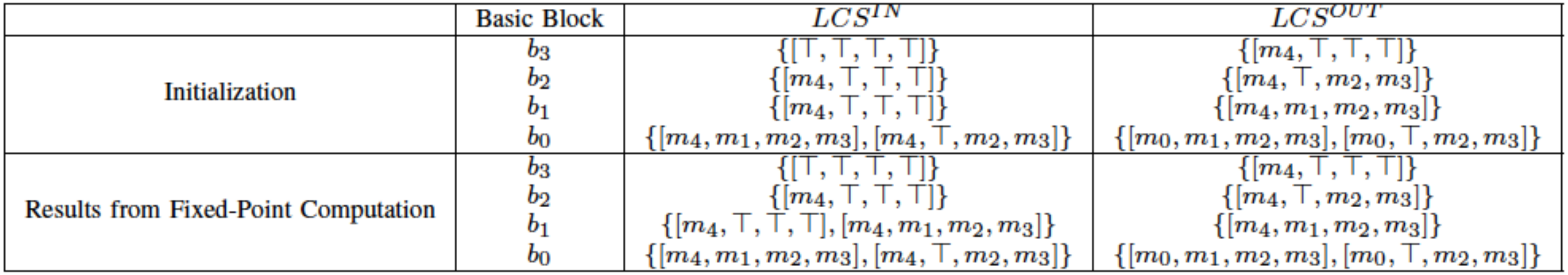


Cache lines	Memory blocks
c_0	m_0, m_4
c_1	m_1
c_2	m_2
c_3	m_3

Program Analysis Technique



	Basic Block	RCS^{IN}	RCS^{OUT}
Initialization	b_0	$\{[\top,\top,\top,\top]\}$	$\{[m_0,\top,\top,\top]\}$
	b_1	$\{[m_0,\top,\top,\top]\}$	$\{[m_0,m_1,m_2,m_3]\}$
	b_2	$\{[m_0,\top,\top,\top]\}$	$\{[m_0,\top,m_2,m_3]\}$
	b_3	$\{[m_0,m_1,m_2,m_3],[m_0,\top,m_2,m_3]\}$	$\{[m_4,m_1,m_2,m_3],[m_4,\top,m_2,m_3]\}$
Results from Fixed-Point Computation	b_0	$\{[\top,\top,\top,\top]\}$	$\{[m_0,\top,\top,\top]\}$
	b_1	$\{[m_0,\top,\top,\top],[m_0,m_1,m_2,m_3]\}$	$\{[m_0,m_1,m_2,m_3]\}$
	b_2	$\{[m_0,\top,\top,\top]\}$	$\{[m_0,\top,m_2,m_3]\}$
	b_3	$\{[m_0,m_1,m_2,m_3],[m_0,\top,m_2,m_3]\}$	$\{[m_4,m_1,m_2,m_3],[m_4,\top,m_2,m_3]\}$



	Basic Block	LCS^{IN}	LCS^{OUT}
Initialization	b_3	$\{[\top,\top,\top,\top]\}$	$\{[m_4,\top,\top,\top]\}$
	b_2	$\{[m_4,\top,\top,\top]\}$	$\{[m_4,\top,m_2,m_3]\}$
	b_1	$\{[m_4,\top,\top,\top]\}$	$\{[m_4,m_1,m_2,m_3]\}$
	b_0	$\{[m_4,m_1,m_2,m_3],[m_4,\top,m_2,m_3]\}$	$\{[m_0,m_1,m_2,m_3],[m_0,\top,m_2,m_3]\}$
Results from Fixed-Point Computation	b_3	$\{[\top,\top,\top,\top]\}$	$\{[m_4,\top,\top,\top]\}$
	b_2	$\{[m_4,\top,\top,\top]\}$	$\{[m_4,\top,m_2,m_3]\}$
	b_1	$\{[m_4,\top,\top,\top],[m_4,m_1,m_2,m_3]\}$	$\{[m_4,m_1,m_2,m_3]\}$
	b_0	$\{[m_4,m_1,m_2,m_3],[m_4,\top,m_2,m_3]\}$	$\{[m_0,m_1,m_2,m_3],[m_0,\top,m_2,m_3]\}$

Schedule/controller co-synthesis

- Again, similar to the previous problem
  - What should be the sampling schedule and the controller design?

- For various different memory architectures, the problem changes
  - For example, cache + scratchpad memory
- Similar problems for multicore processors (e.g., with shared cache)

Cross-layer Design

- What are the “layers” in a cross-layer design?
  - Model
  - Code

    side-effects (e.g., all control inputs applied simultaneously?)

    numerical precision
  - Implementation of the code on a distributed architecture

    timing
  - Hardware/device level characteristics

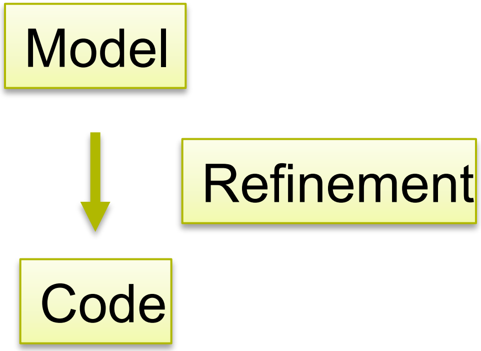
    incorrect computations

    need to reboot - timing

Cross-layer Design

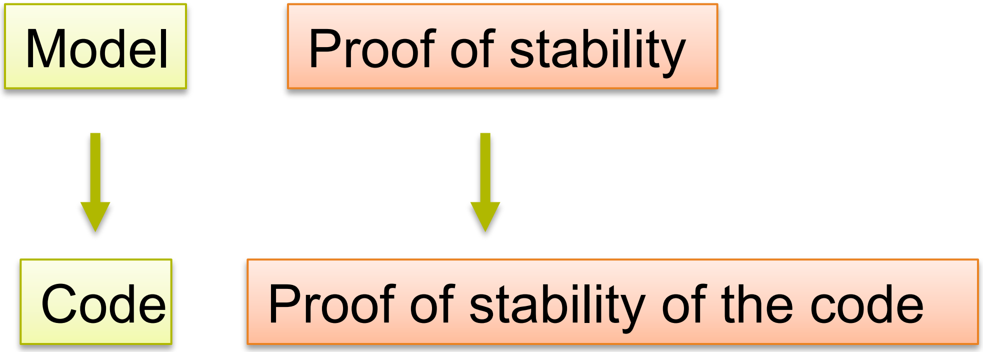
- Model to code
  - How to verify that the model-level semantics are preserved in the code?
  - Simulink code generator offers different optimization options. But what impact do they have on preserving model semantics?

  - In the case of mismatch, should we change the model, the refinement, of both? How?



Cross-layer Design

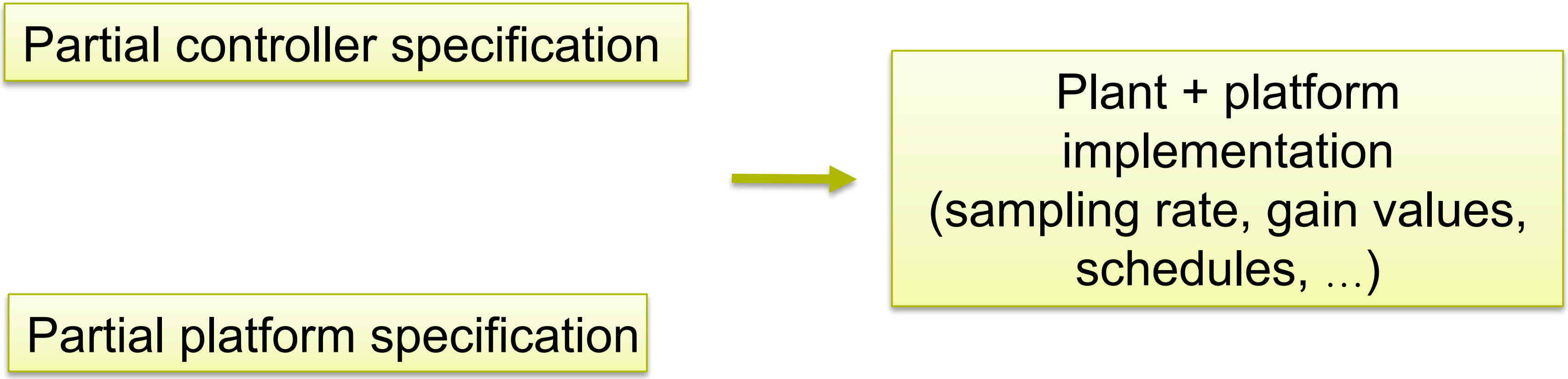
- Model to code
  - How to carry over proofs from the model level to an implementation?



  - Which refinements are “proof preserving”?

Cross-layer Design

- Code to platform
  - Co-synthesis
    - Given plant + control objectives + platform constraints
    - Synthesize controller + its implementation



Partial controller specification

Partial platform specification

Plant + platform implementation (sampling rate, gain values, schedules, …)

  - What kind of optimization techniques are needed?

Recurring open issues

- Some open (control theoretic) issues
  - Dealing with occasional loss of feedback signal
    - Work in NCS: only over infinite horizons, deals only with stability
    - Needed: finite length characterizations of allowed loss patterns, beyond stability
  - Tighter analysis of switched systems with known switching behavior
    - Known results: stability analysis under arbitrary switching patterns, very conservative results
    - Needed: analysis for specified switching behaviors, synthesize switching patterns that guarantee stability
  - Control with non-uniform sampling periods
  - Control with state-specific communication delays

References

- Wanli Chang, Dip Goswami, Samarjit Chakraborty, Lei Ju, Chun Jason Xue, Sidharta Andalam: **Memory-Aware Embedded Control Systems Design**. IEEE Trans. on CAD of Integrated Circuits and Systems 36(4): 586-599 (2017)
- Samarjit Chakraborty, Mohammad Abdullah Al Faruque, Wanli Chang, Dip Goswami, Marilyn Wolf, Qi Zhu: **Automotive Cyber-Physical Systems: A Tutorial Introduction**. IEEE Design & Test 33(4): 92-108, 2016
- Wanli Chang, Samarjit Chakraborty: **Resource-aware Automotive Control Systems Design: A Cyber-Physical Systems Approach**. Foundations and Trends in Electronic Design Automation 10(4): 249-369 (2016)
- Dip Goswami, Reinhard Schneider, Samarjit Chakraborty: **Relaxing Signal Delay Constraints in Distributed Embedded Controllers**. IEEE Trans. Contr. Sys. Techn. 22(6): 2337-2345 (2014)
- Harald Voit, Anuradha M. Annaswamy, Reinhard Schneider, Dip Goswami, Samarjit Chakraborty: **Adaptive switching controllers for systems with hybrid communication protocols**. American Control Conference (ACC) 2012
- Harald Voit, Anuradha Annaswamy, Reinhard Schneider, Dip Goswami, Samarjit Chakraborty: **Adaptive switching controllers for tracking with hybrid communication protocols**. Conference on Decision and Control (CDC) 2012