Introduction to Embedded Systems & Control: What are Cyber-Physical Systems?

Samarjit Chakraborty
www.rcs.ei.tum.de
TU Munich, Germany

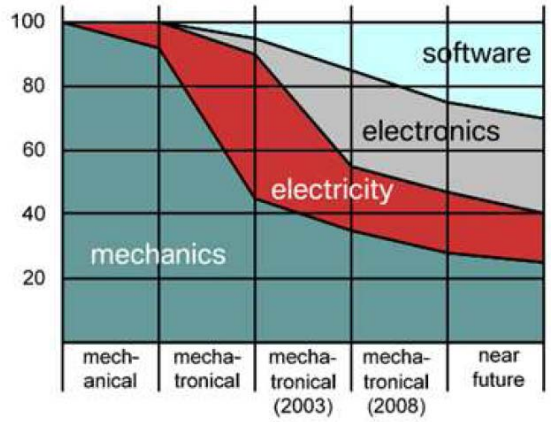
What are Cyber-Physical Systems?

- Networked Embedded Systems
- Sensor network based systems
- Internet of Things (IoT)
- Embedded Control Systems

… will use automotive as a running example

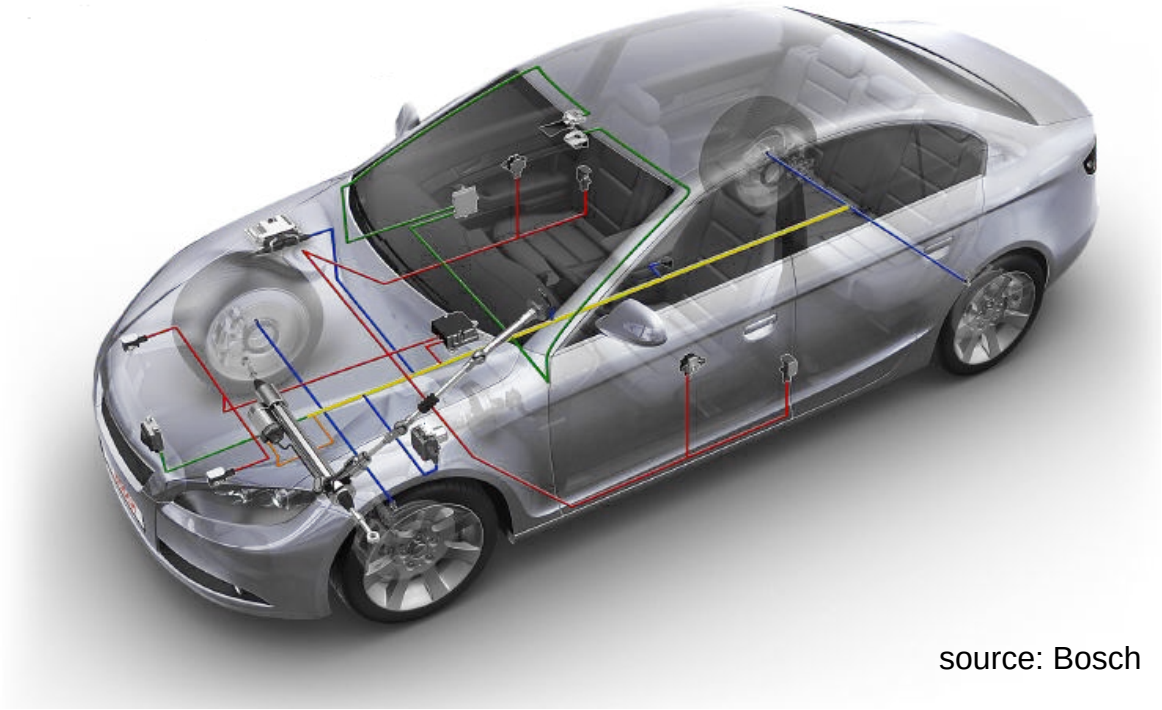
Automotive E/E Architecture Trends

Change in the value chain.
Value creation in cars is increasingly driven by electronics und software.



Source:
BMW Group, FTF 2010 Orlando
Energy Saving Strategies in Future Automotive E/E Architectures

The Modern Car



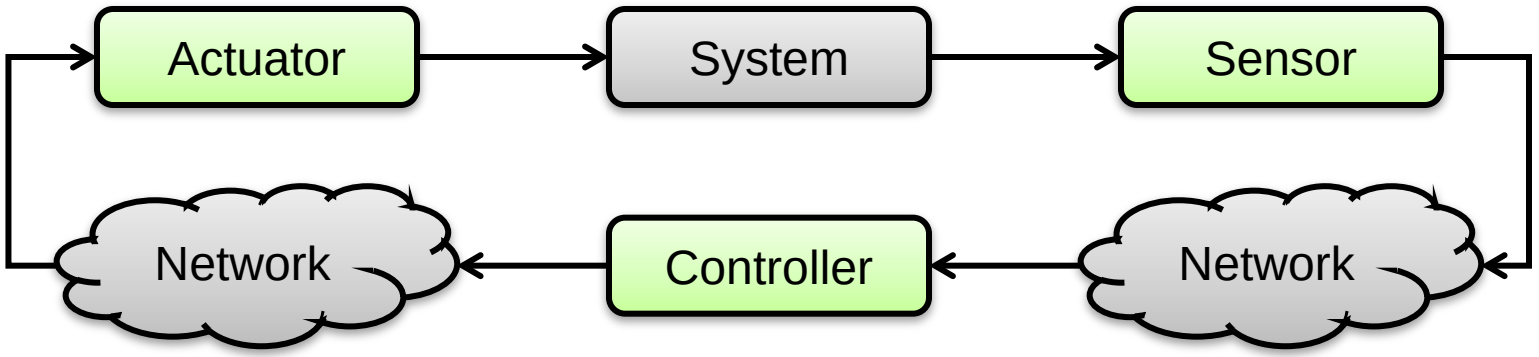
source: Bosch

- 50 – 100 Electronic Control Units (ECUs)
- Complex in-vehicle network (CAN, FlexRay, Ethernet)
- Complex systems software, operating systems
- Several distributed control applications (safety-critical & comfort)

Control Applications

Speed adjustment

Radar

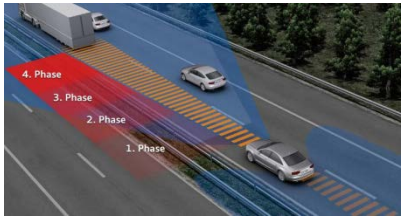


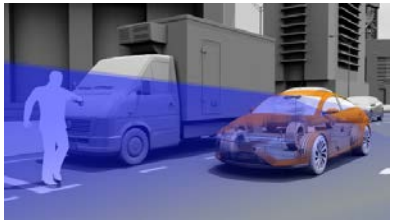
ACC ECU

Many of the safety-critical, driver assistance and comfort applications implement control functionality

Current Design Flow

Applications





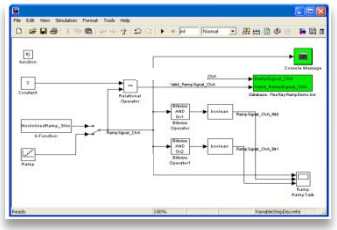
. . .

High-level requirements

Model

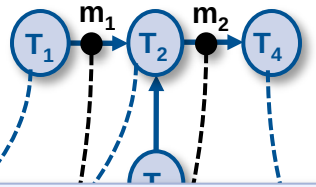
$$\ddot{y}(t) + \dot{y}(t) = K_P u(t)$$

$$u(t) = K_D \dot{e}(t)$$



code generation

algorithm design



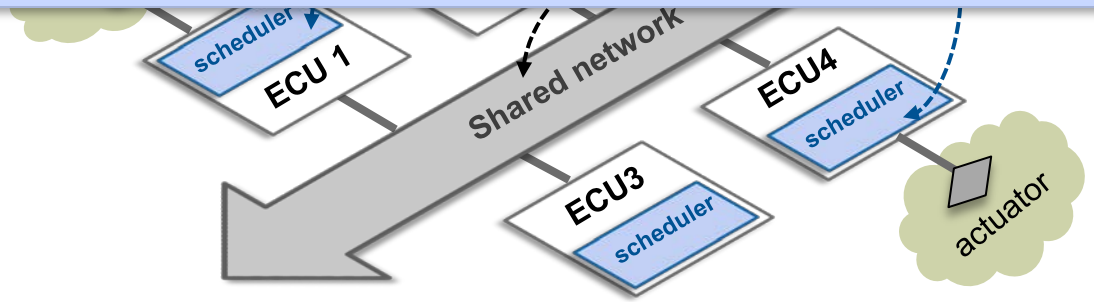
Task partitioning

HW/SW architecture

Multiple processor units (PUs)

co

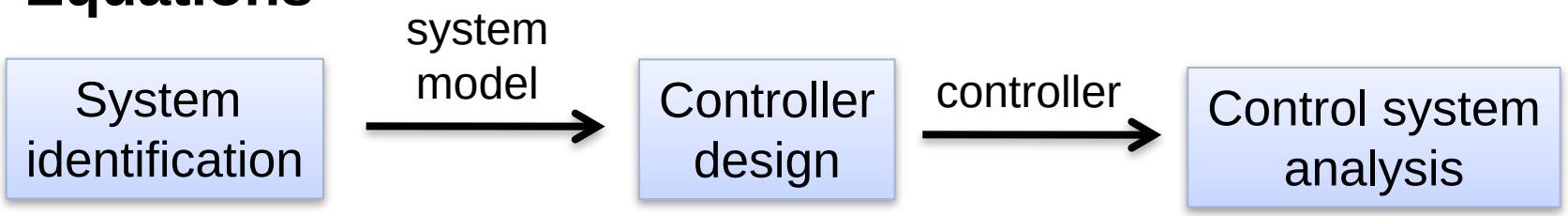
Algorithm & Architecture designed separately followed by integration, testing, debugging, …



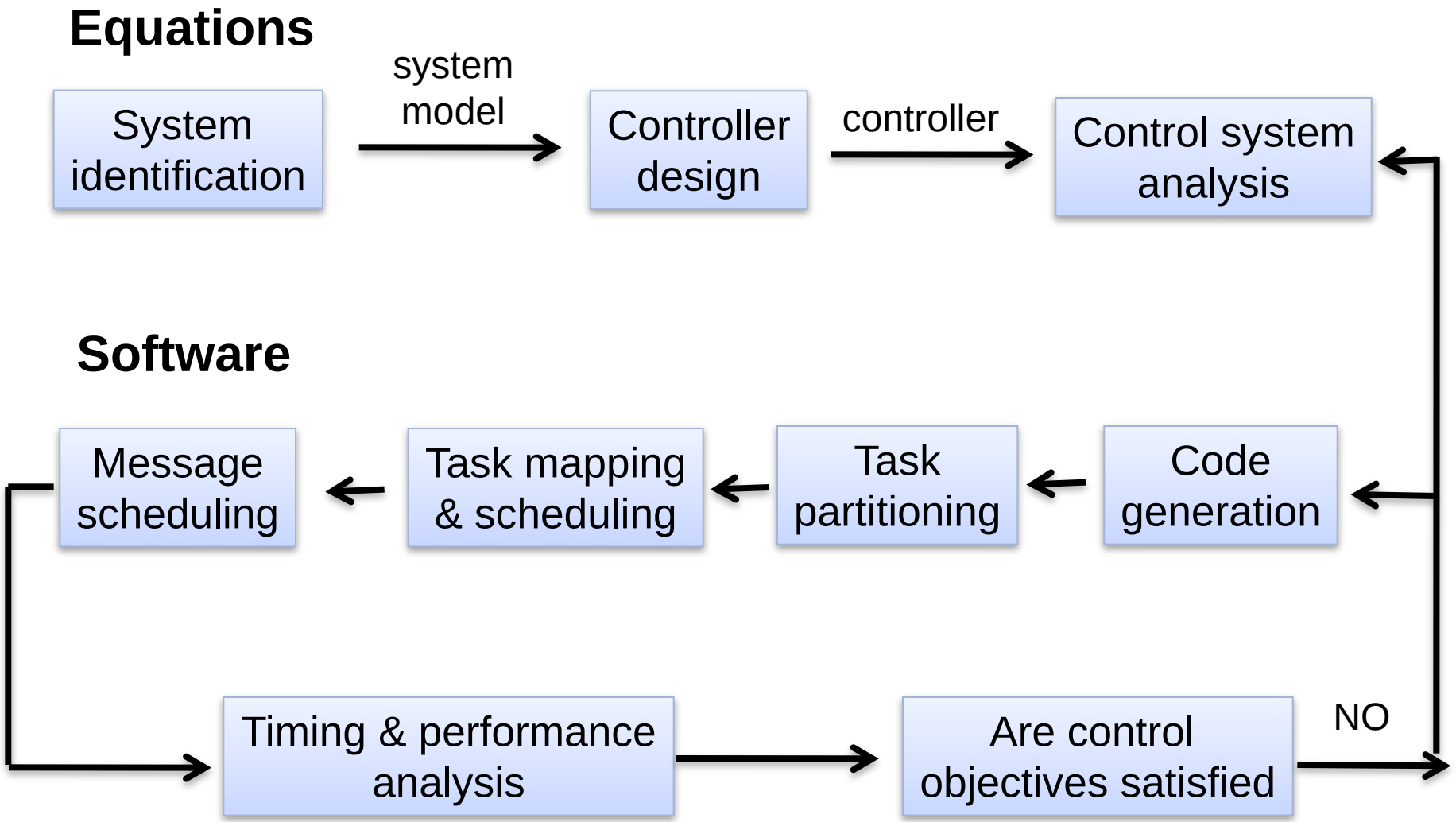
Task and Message scheduling

Control Systems Design

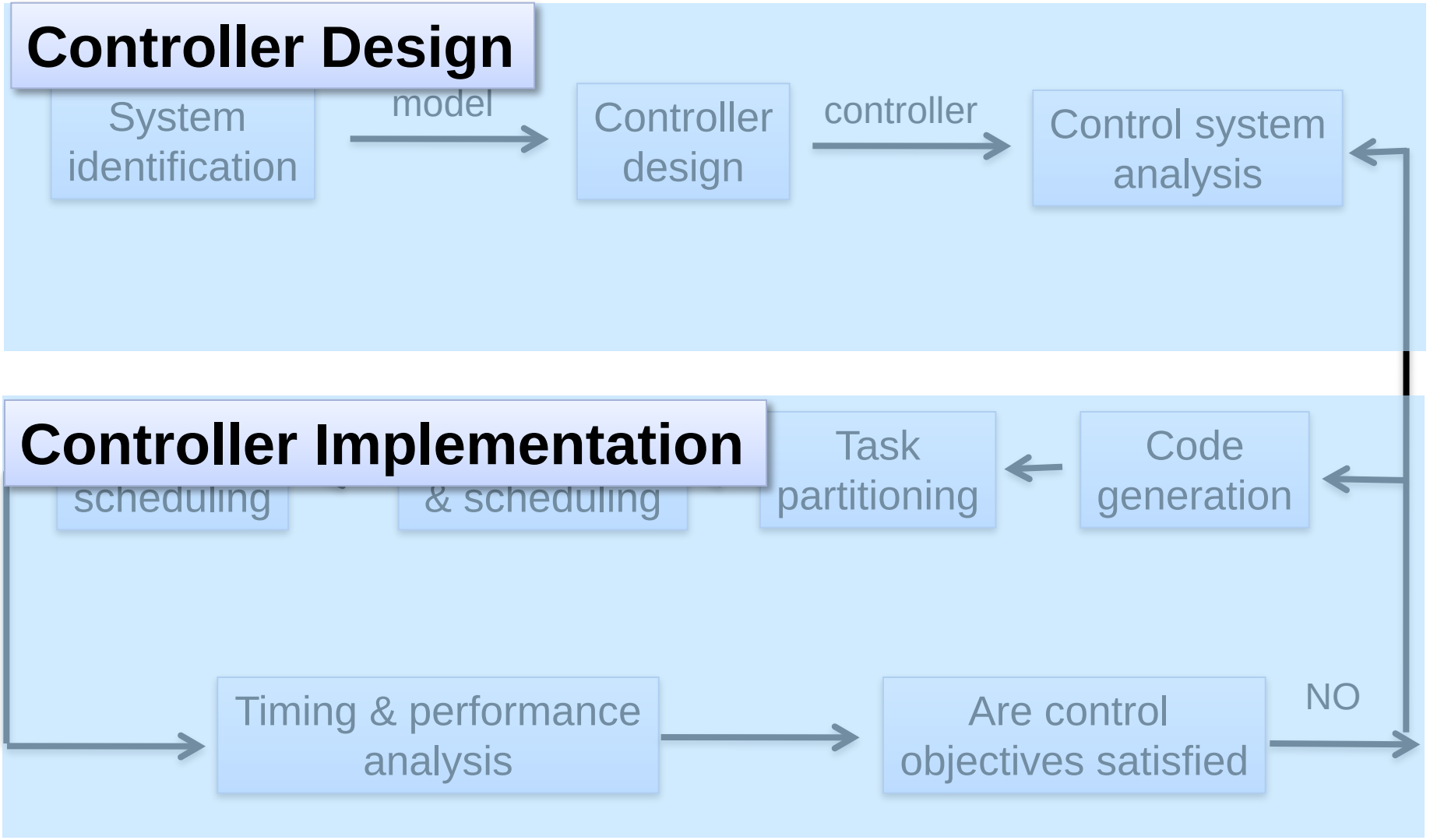
Equations



Control Systems Implementation

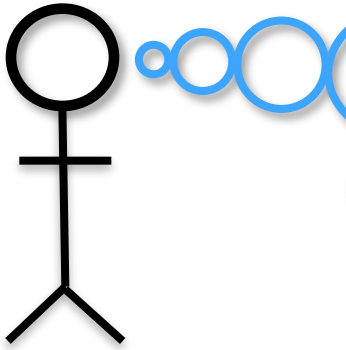


The Design Flow



The Design Flow

Controller Design

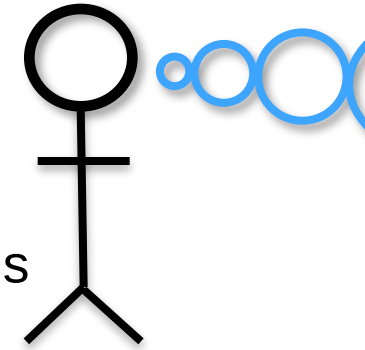


Control theorist

Design assumptions

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

Controller Implementation



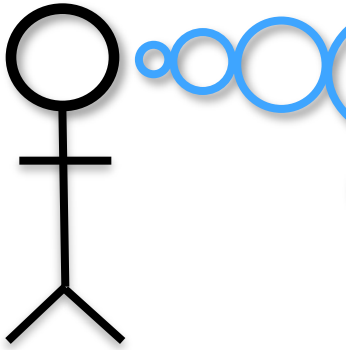
Embedded systems engineer

Implementation reality

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

The Design Flow

Controller Design

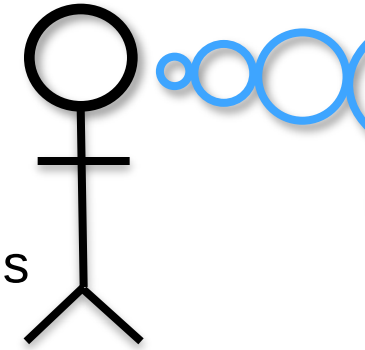


Control theorist

These are implementation details

Not my problem!

Controller Implementation

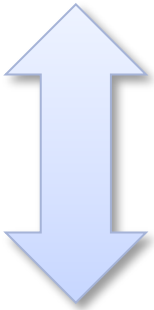


Embedded systems engineer

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?

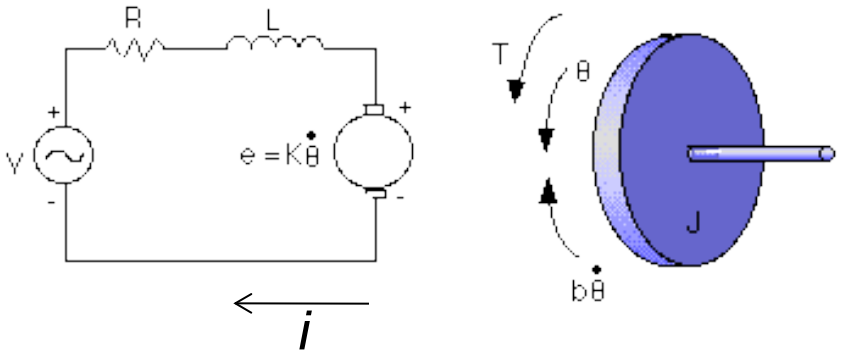
Solution: Controller/Architecture Co-design

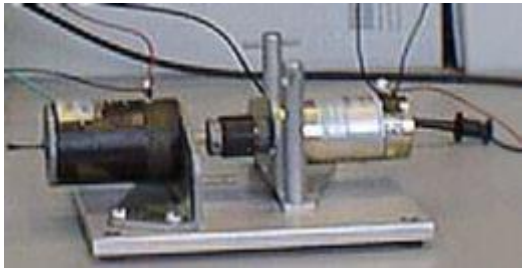
Cyber-Physical Systems

- Tight interaction between computational (cyber) and physical systems
- Isolated design of the two systems leads to:
  - Higher integration, testing and debugging costs
  - Poor resource utilization (e.g., unnecessarily high sampling rates)
- Question:
  - Can existing techniques from real-time & embedded systems be directly applied? We know hardware/software co-design
- Answer:
  - New techniques are required
  - Traditional embedded systems design have focused only on meeting *deadlines*
  - Here, deadlines take a back seat

Dynamical Systems: DC Motor

The time dependence of the various ***states*** of ***dynamical systems*** is described by a set of difference or differential equations.





J = moment of inertia of the rotor
b = system damping ratio,
K = EMF and Torque constant,
R = armature resistance,
L = armature inductance,
i = armature current,
V = input, DC terminal voltage
θ = output, position of shaft

$$J\ddot{\theta} + b\dot{\theta} = Ki$$
$$L\dot{i} + Ri = V - K\dot{\theta}$$

Modeling Approaches: State-space

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

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

System *states x(t)*: (1) motor shaft position, θ (2) motor shaft velocity, $\dot{\theta}$ and (3) armature current, i

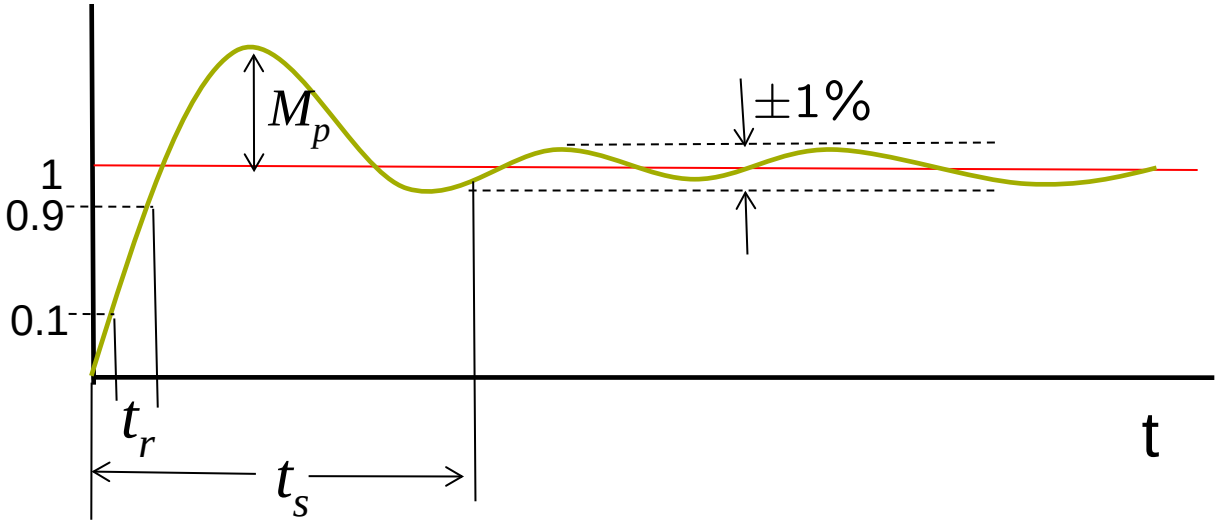
System *Input u(t)*: Terminal voltage, V

One might want to regulate one or more system *states*. The regulated states are known as *output y(t).* E.g., motor speed or position

Performance Metrics

Transient Behavior:

- The **rise time** t_r is the time it takes the system to reach the vicinity of its set point or reference command
- The **settling time** t_s is the time it takes the system transients to decay
- The **overshoot** M_p is the maximum amount the system overshoots (and often expressed as a percentage of reference command)



Performance Metrics

Steady-state Behavior:

- **Integral Input Cost:** $\int u(t)^2 dt$

  - indicator of how much effort is needed for the control

- **Integral Error Cost:** $\int e(t)^2 dt$

  - indicator of how accurate the control performance is

There are many possible metrics.

Moreover, one can also consider a weighted combination of the various cost such metrics

Control Tasks - Characteristics

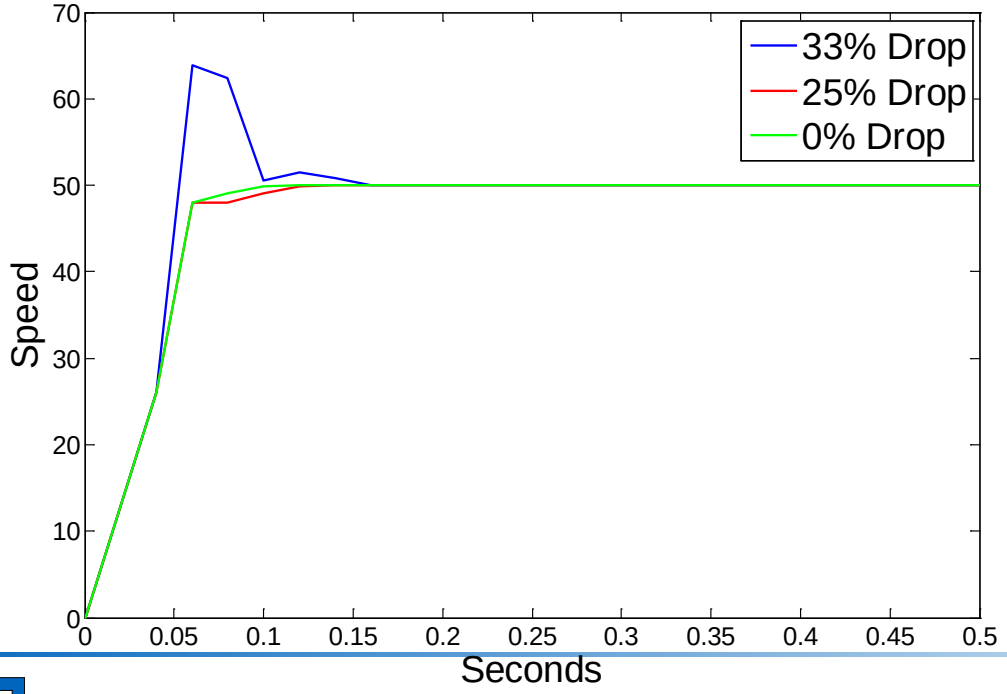
The deadlines are usually not ***hard*** for control-related messages

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

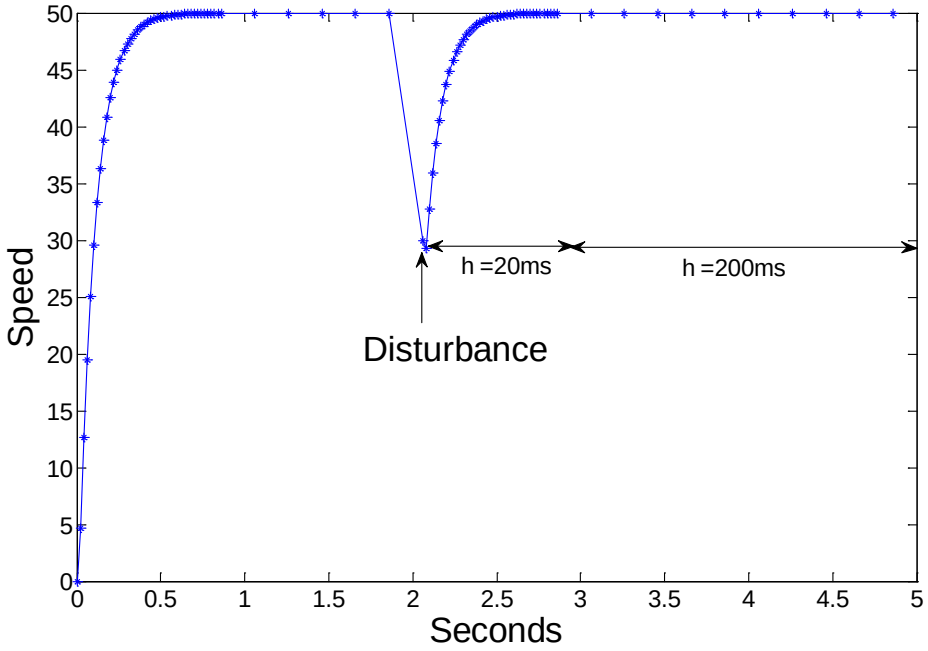
A fraction of feedback signals being dropped



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



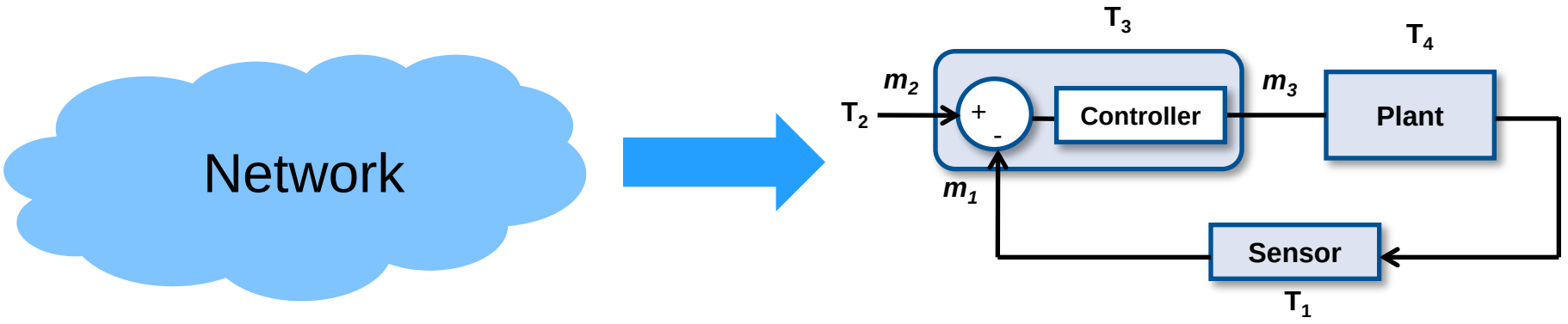
(1) The computation requirement at the steady state is less, i.e., sampling frequency can be reduced (e.g., event-triggered sampling)

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

Bottomline

- Real-time Systems
  - Meeting deadlines is the center of attraction
- CPS View
  - Deadline takes the back seat
  - As a result, the design space becomes bigger
  - Resulting design is better, robust, cost-effective …

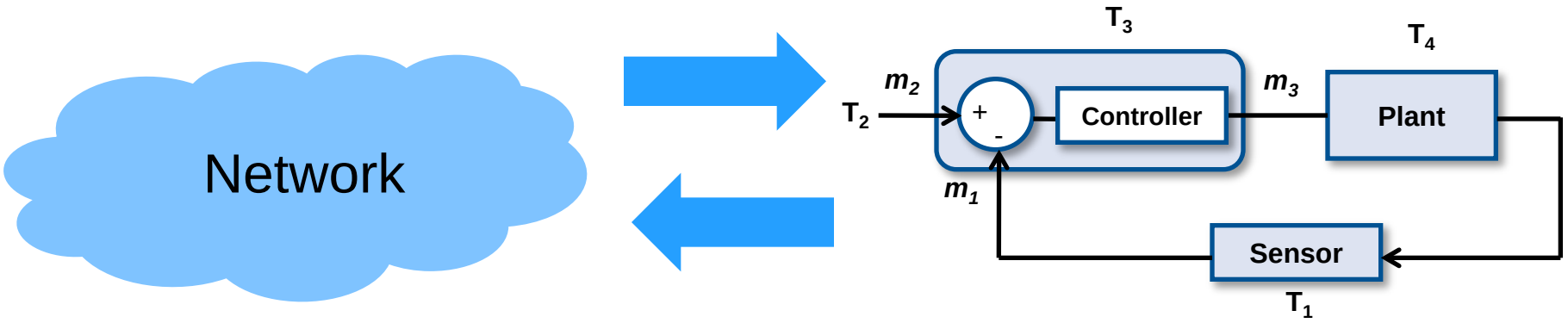
What about NCS?



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
- CPS Problem: **How to design the network and the controller together? Co-design Problem**

CPS-Oriented Design Example

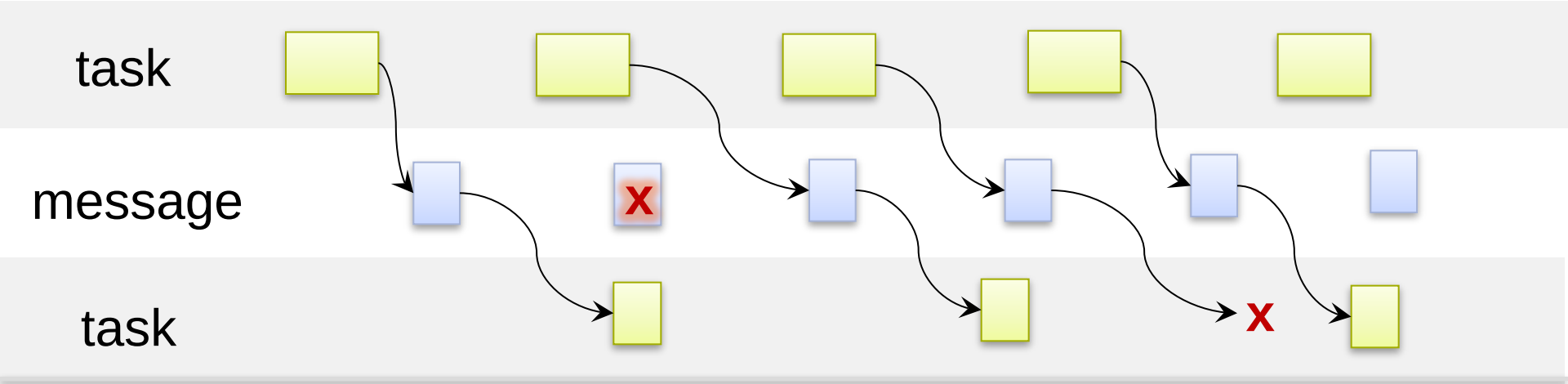
- Control over FlexRay
- FlexRay is made up of time-triggered (TT) and event-triggered (ET) segments
- Conventional design
  - Use time-triggered segment for control messages – one slot for each control message
  - Expensive

Challenge:
Can we achieve the same control performance with fewer time-triggered slots?

FlexRay – Event Vs Time-triggered

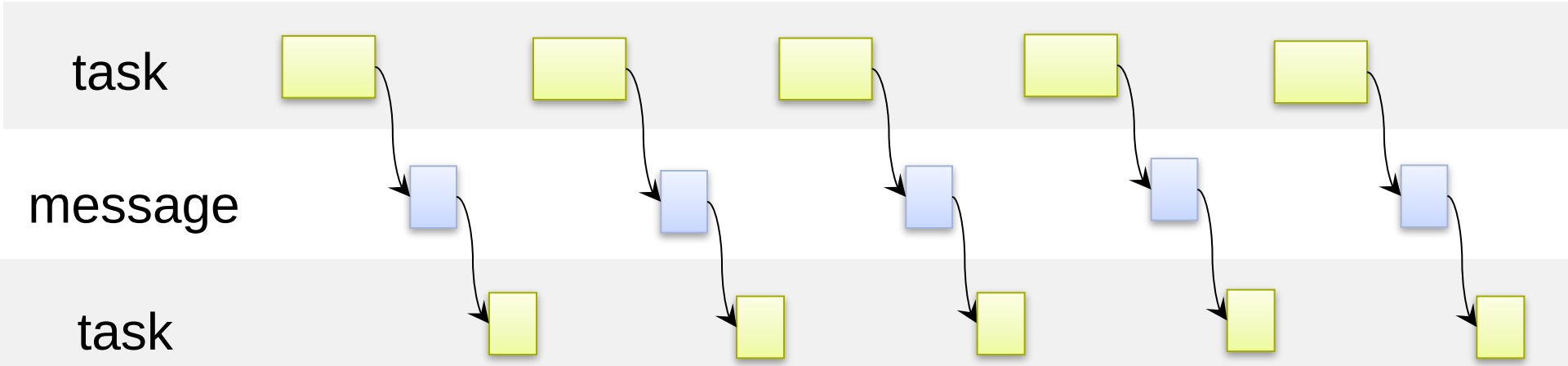
Asynchronous

Jitter, lost/unused data

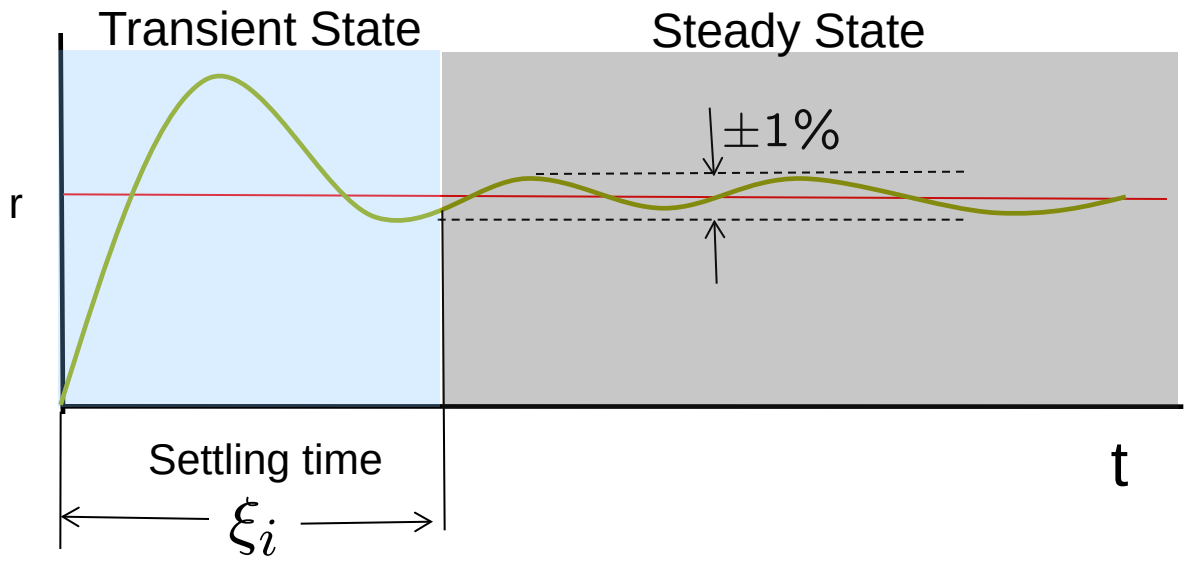


Synchronous

Predictable



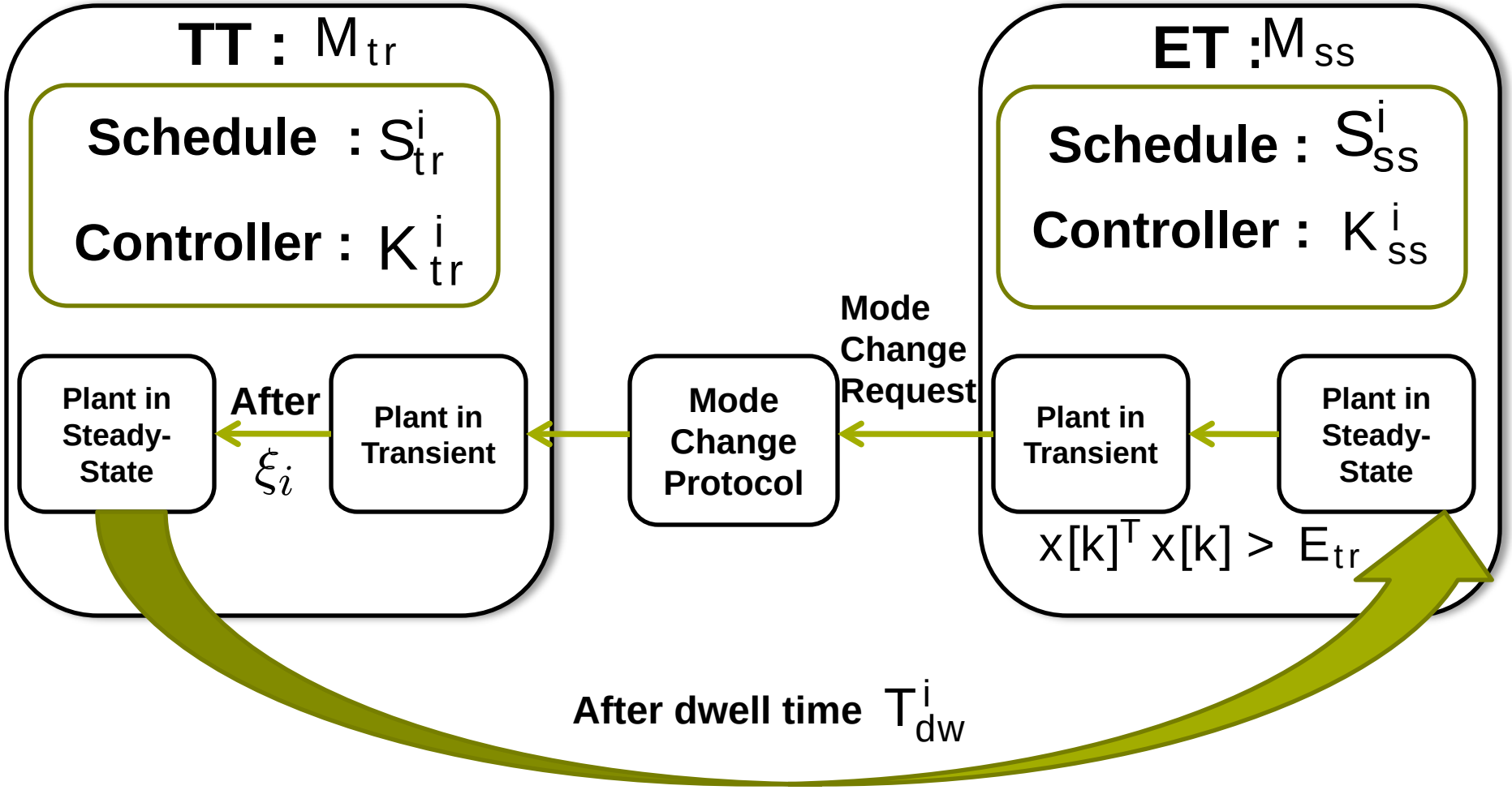
Quality of Control vs. System State



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



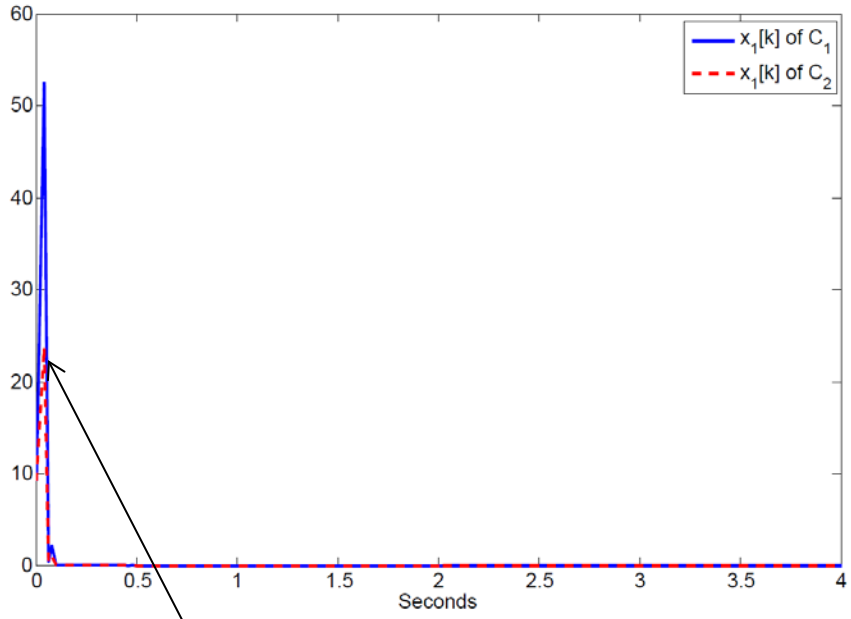
Example

- 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

$$F_{tr}^1 = \begin{bmatrix} 7.4394 & 2.6819 \end{bmatrix}$$

$$F_{tr}^2 = \begin{bmatrix} 0.0417 & 2.9722 \end{bmatrix}$$

Quality of control

C_1

$$\xi_1 = 0.14 \text{ Sec.}$$

$$\sum_k x[k]^T x[k] = 8.6 \times 10^3$$

$$\sum_k u[k]^2 = 4.7276 \times 10^4.$$

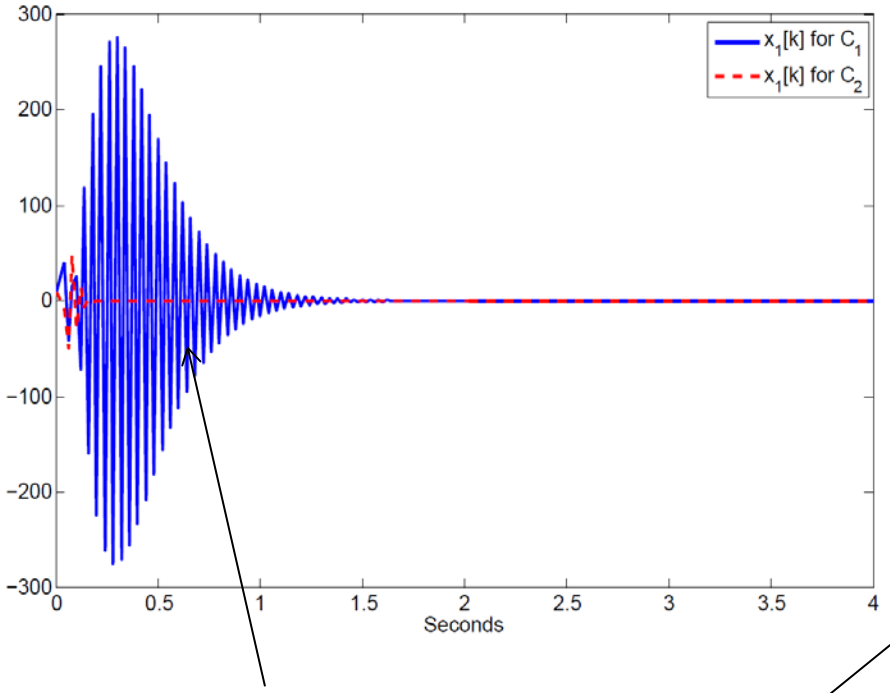
C_2

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

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

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

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

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

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

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

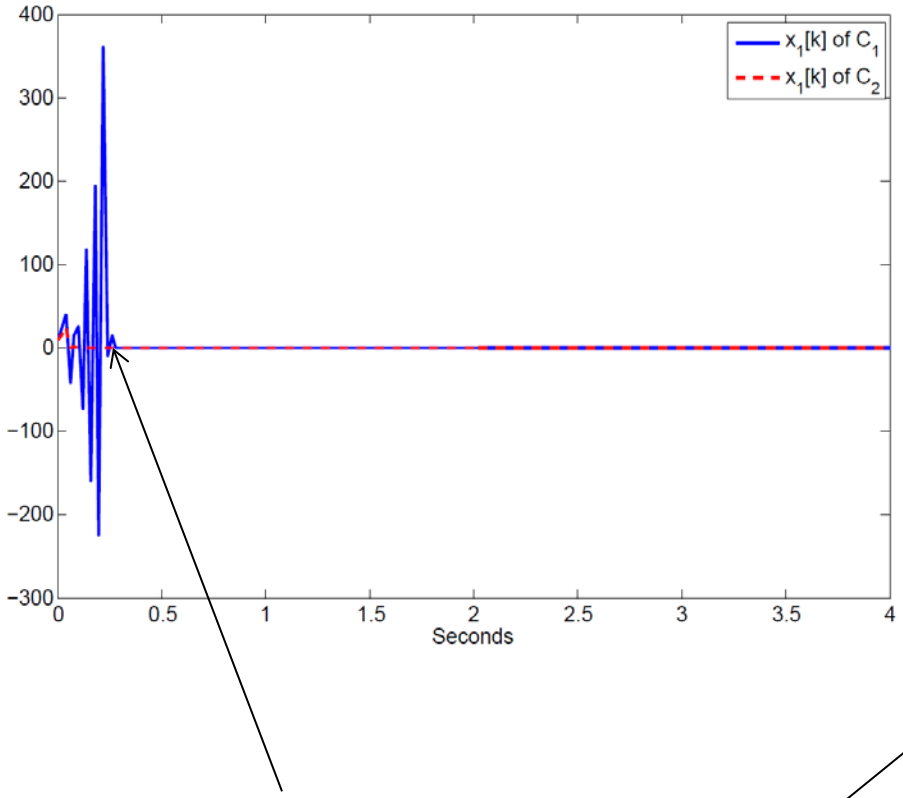
C_2

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

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

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

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

C_1

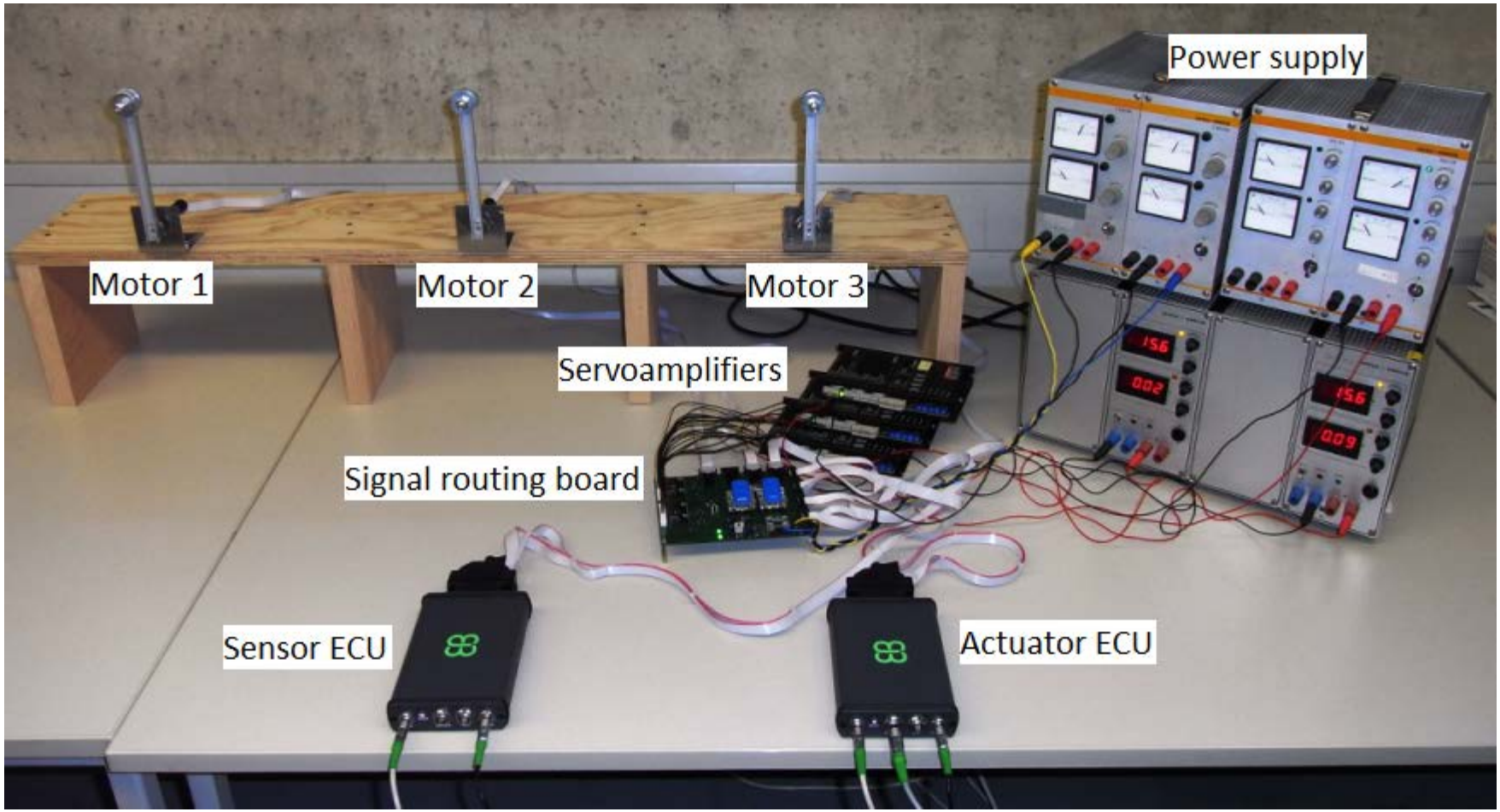
$$\xi_1 = 0.36 \text{ Sec.}$$
$$\sum_k x[k]^T x[k] = 1.3651 \times 10^6$$
$$\sum_k u[k]^2 = 2.3607 \times 10^6.$$

C_2

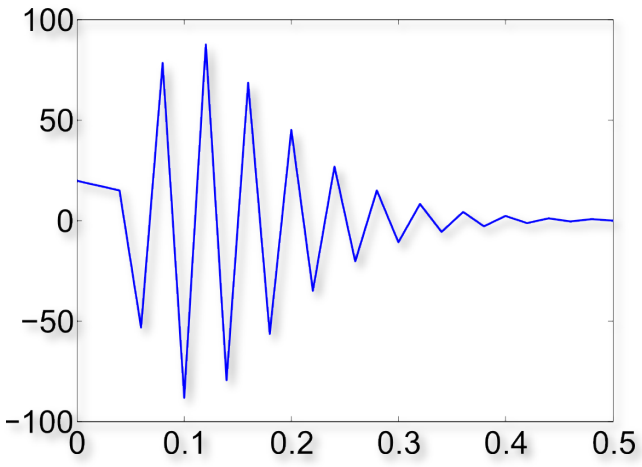
$$\xi_2 = 0.14 \text{ Sec}$$
$$\sum_k x[k]^T x[k] = 1.8136 \times 10^3$$
$$\sum_k u[k]^2 = 1.2857 \times 10^4.$$

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

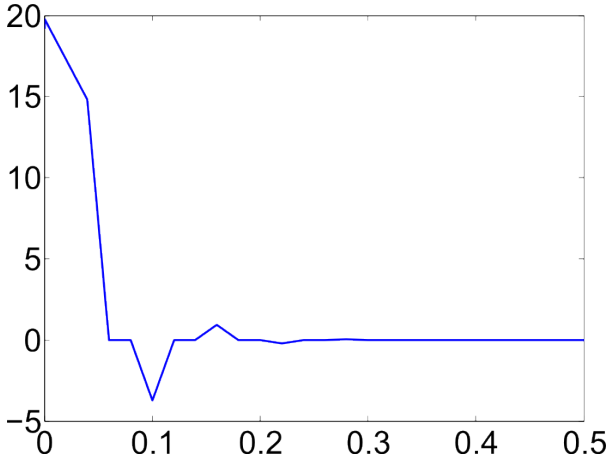
Experimental Setup



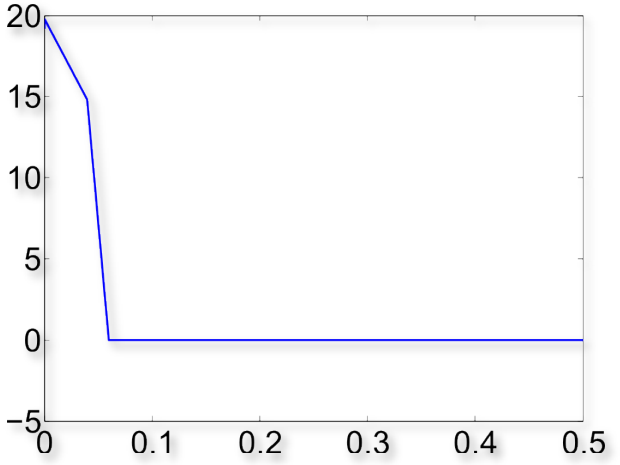
Experimental Results



Purely event-triggered



Mixed time-/event-triggered



Purely time-triggered

Concluding Remarks

- Viewing cyber-physical systems not as traditional real-time systems
- Breaks traditional concepts of abstraction?
- Current practice:
  - Specify control laws
  - Specify architecture and schedules
  - Test and debug (modify architecture and schedule)
- Problem:
  - Synthesize the architecture/schedule from control performance constraints
  - Large constrained optimization problem

… and now something more futuristic