

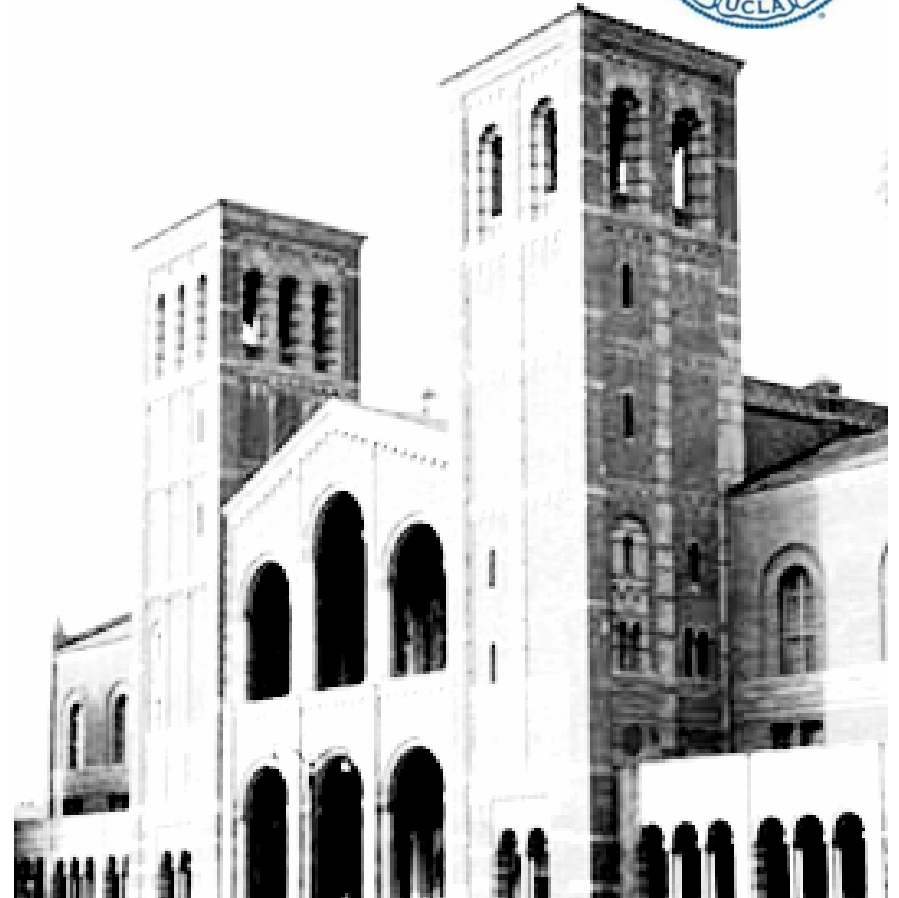
Self-Triggered Control Over CAN and FlexRay Buses

Adolfo Anta
Paulo Tabuada



CyPhyLab
Electrical Engineering Department
UCLA

Annual Research Review
April 2009



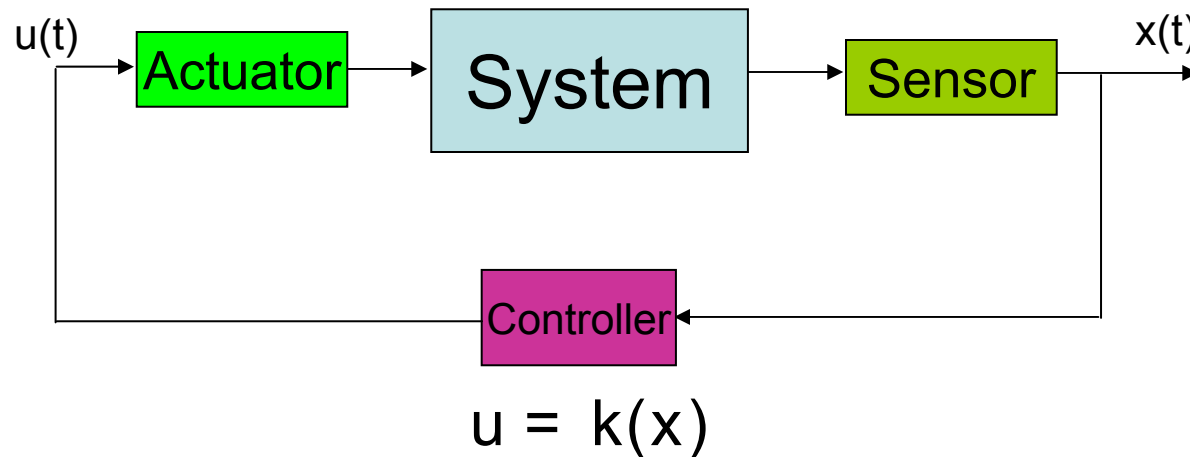


Introduction

Control system:

$$\dot{x} = f(x; u) = f(x; k(x))$$

$x(t)$: state
 $u(t)$: input



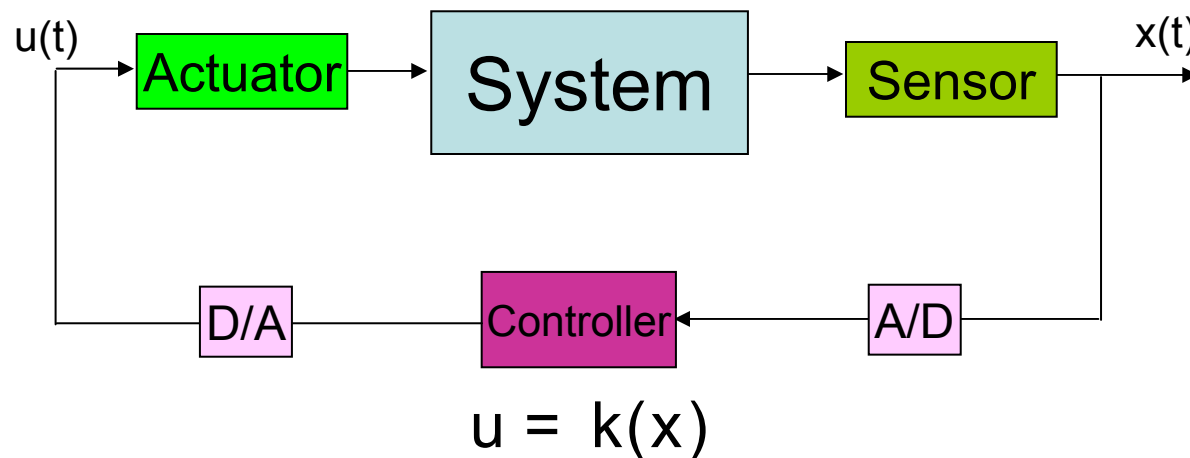


Introduction

Control system:

$$\dot{x} = f(x; u) = f(x; k(x))$$

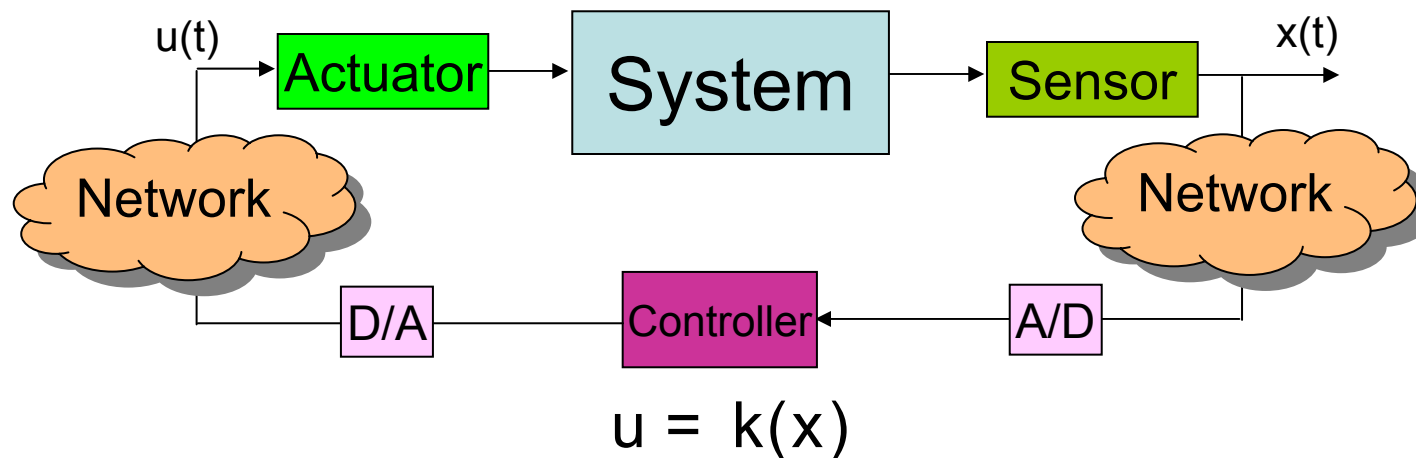
$x(t)$: state
 $u(t)$: input



- When should we sample the state?
- When should we execute the controller?
- Periodic or aperiodic execution of the control law?

Introduction

In the case of distributed control systems:



- When shall the sensor send a message to the controller?
- When shall the controller send a message to the actuator?



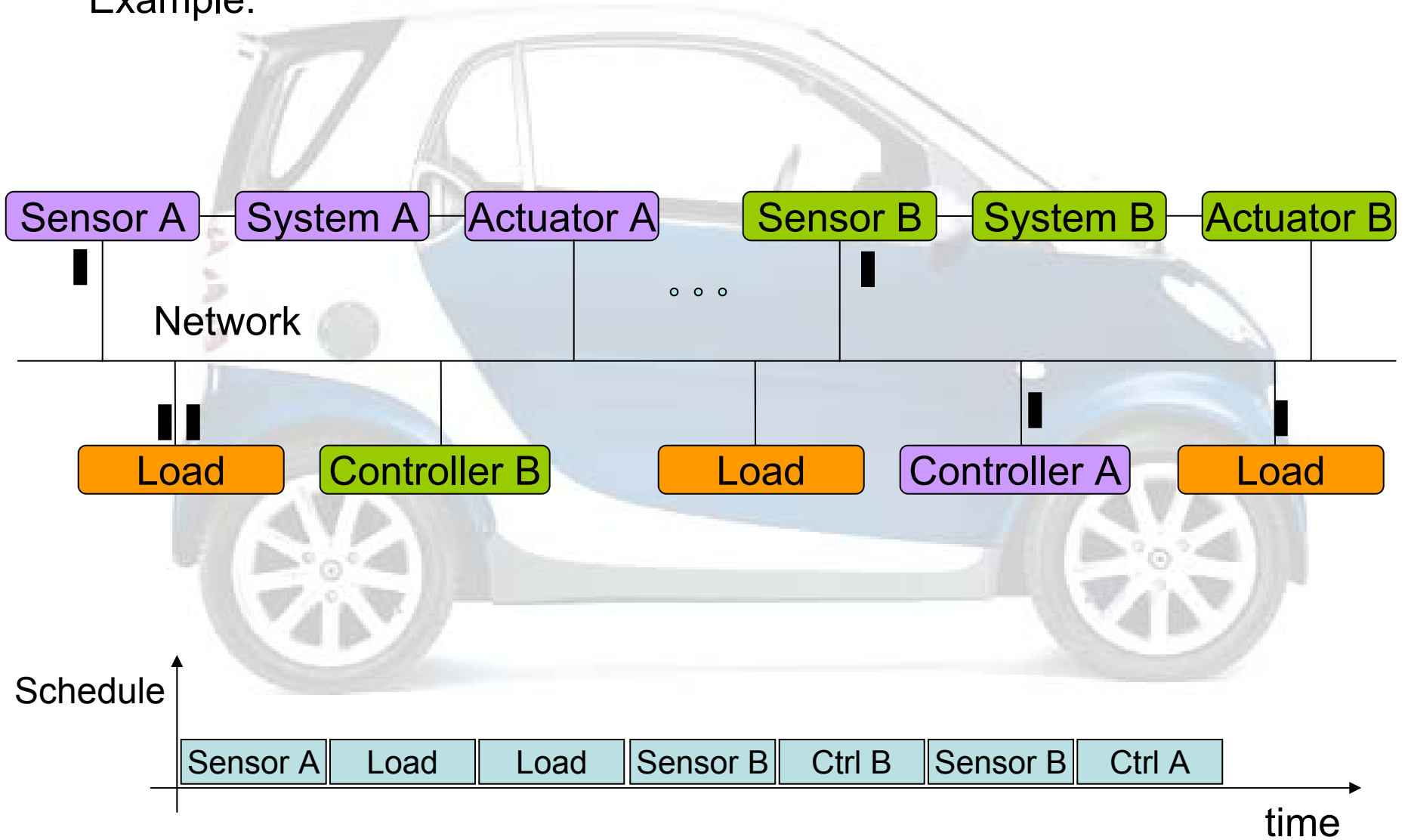
Introduction

- A vast majority of control systems are implemented nowadays over networks
- Communication between the different agents is usually established through periodic messages.
- This strategy facilitates the analysis and implementation but over-imposes hard constraints in the scheduling.
- Our goal: find a dynamic message transmission strategy that preserves stability of the original continuous system while saving bandwidth.



Introduction

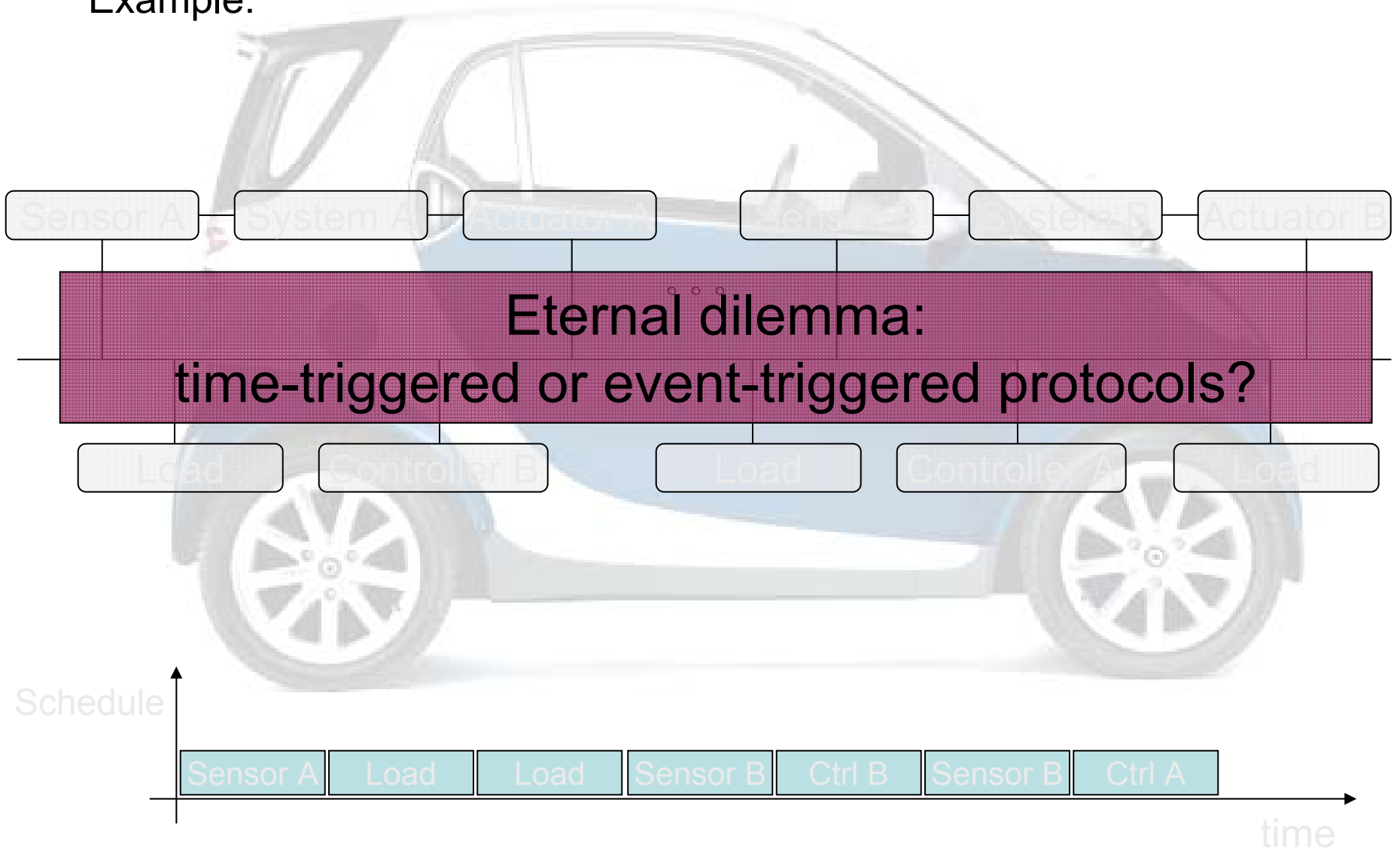
Example:





Introduction

Example:





Strategies

Closing the loop implies two messages, one from sensor to controller and another from controller to actuator.

There are 4 possible strategies for closing the loop:

1st) Periodic messages

- Simplifies the design, scheduling and the analysis of networked control systems.



Strategies

Closing the loop implies two messages, one from sensor to controller and another from controller to actuator.

There are 4 possible strategies for closing the loop:

1st) Periodic messages

- Simplifies the design, scheduling and the analysis of networked control systems.

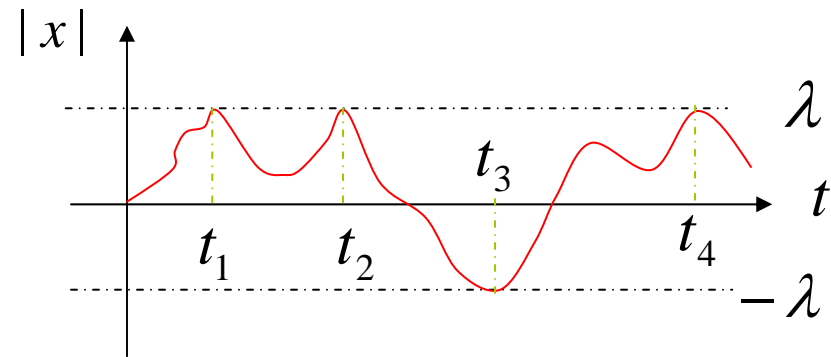
- Represents a conservative solution since the period for the transmission is chosen for a worst-case scenario.



Strategies

2nd) Event-triggered:

$$h(x(t)) = \lambda$$



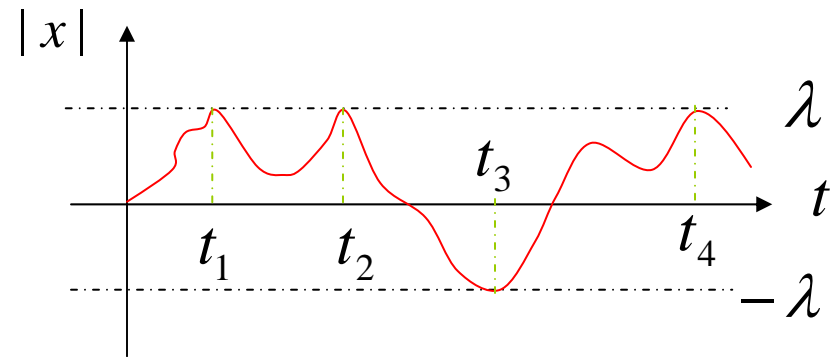
- Seems natural to update the controller when something significant happens in the system.
- Will reduce resource usage and provide robustness.



Strategies

2nd) Event-triggered:

$$h(x(t)) = \lambda$$



- Seems natural to update the controller when something significant happens in the system.

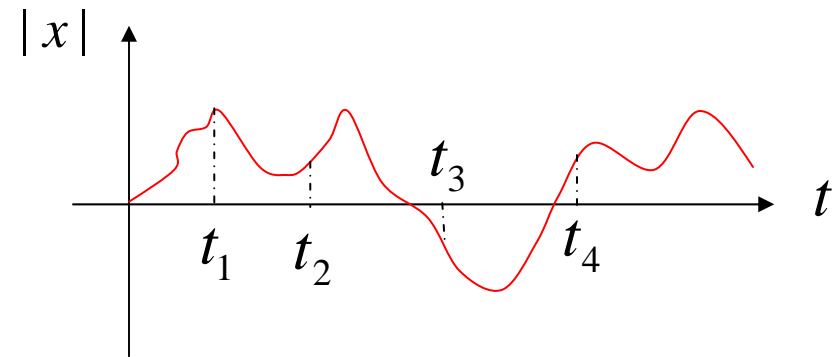
- Will reduce resource usage and provide robustness.

- Transmission times are not known in advance: scheduling analysis is not possible.

- Extra hardware is needed to check the triggering condition permanently.

Strategies

3rd) Time-triggered:



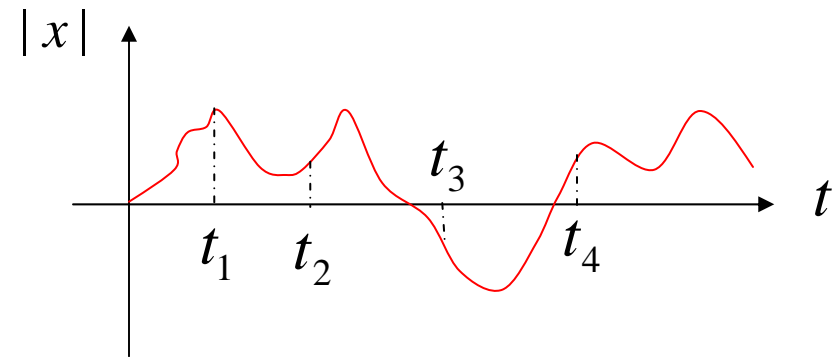
- Compute offline the sequence of times generated by the event-triggered strategy:

$$h(x(t)) = \lambda \Rightarrow t_i = g(\lambda)$$

- Given the model of a dynamical system, compute estimates of the times when the inequality would be violated: no extra hardware.

Strategies

3rd) Time-triggered:



- Compute offline the sequence of times generated by the event-triggered strategy:

$$h(x(t)) = \lambda \Rightarrow t_i = g(\lambda)$$

- Given the model of a dynamical system, compute estimates of the times when the inequality would be violated: no extra hardware.

- Estimates will be very conservative.
- Solution is not robust: we rely completely on the mathematical model of our system (no feedback in the process).



Strategies

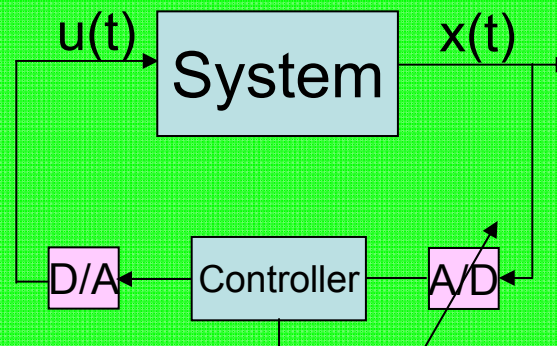
4th) Self-triggered:

- Use the available information (the state) to decide the next inter-execution time.
- The state always has to be measured (or estimated) to compute the controller.

Strategies

4th) Self-triggered:

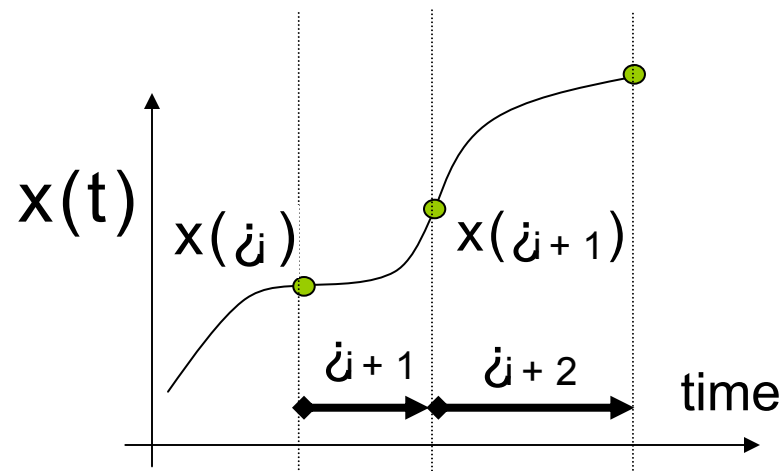
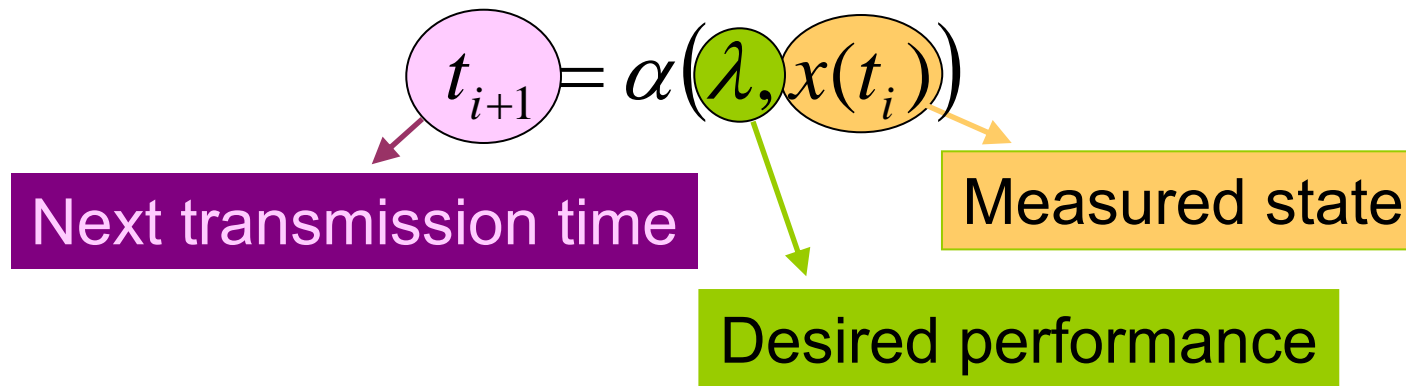
- Use the available information (the state) to decide the next inter-execution time.
- The state always has to be measured (or estimated) to compute the controller.
- Feedback is introduced in the sampling process; no extra hardware is required.





Self-triggered control

- Dynamic model: $\dot{x} = f(x, u)$
- + Event condition: $h(x(t)) = \lambda$





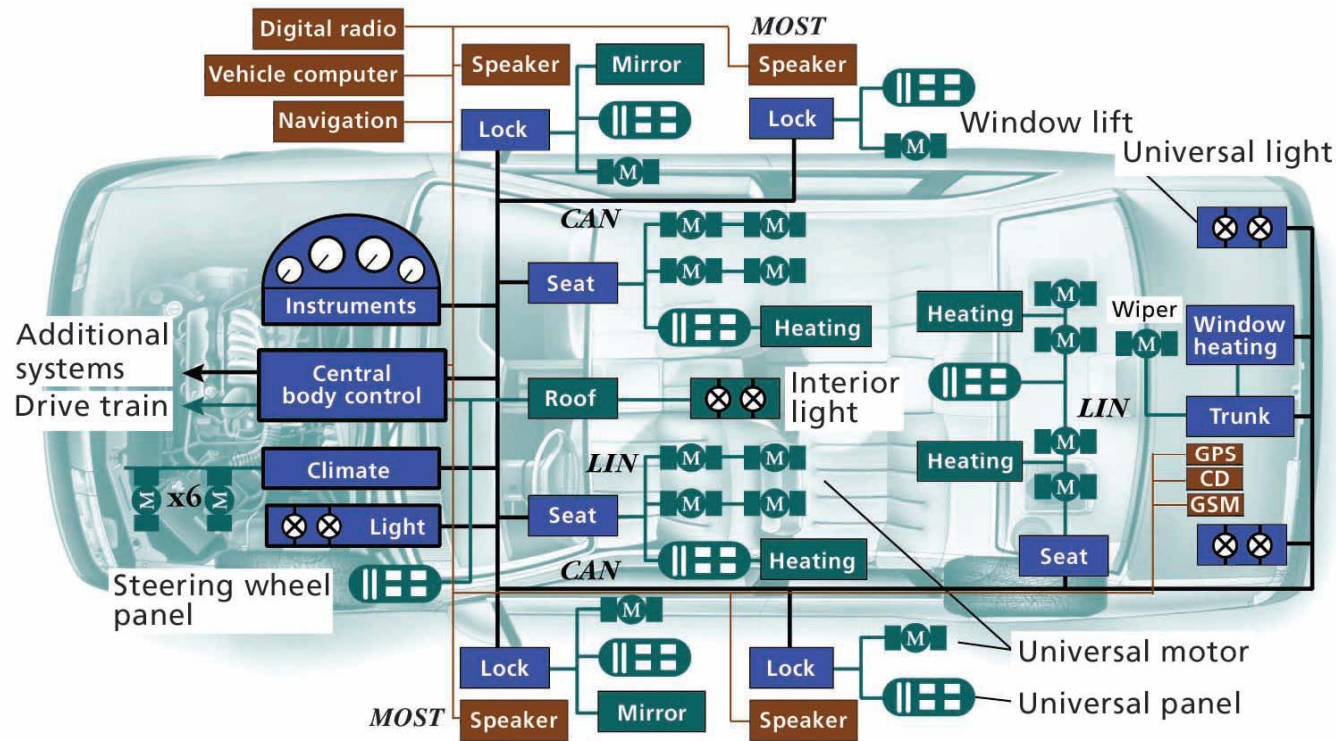
Self-triggered control

$$t_{i+1} = \alpha(\lambda, x(t_i))$$

- The sequence of times t_i represents the instants when the loop needs to be closed, i.e., when the actuator needs to be updated.
- However the control system needs to share the network with other applications.
- A schedulability analysis is required to guarantee that the loop will be closed whenever needed.
- We have developed such analysis for two common protocols.

CAN network

- The Controller Area Network (CAN) protocol is currently the most widely used protocol for vehicles.



CAN Controller area network
 GPS Global Positioning System
 GSM Global System for Mobile Communications
 LIN Local interconnect network
 MOST Media-oriented systems transport

PEI Tech, Limerick University



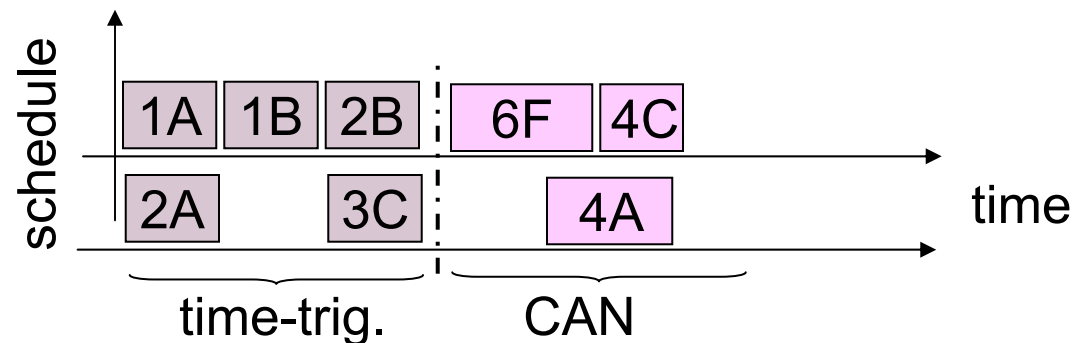
CAN network

- Under CAN protocol, every node sends messages with an assigned priority.
- We will assign priorities according to the self-trigger times t_i : the shorter t_i , the higher the priority (in the spirit of Earliest Deadline First).
- Performance and stability can be guaranteed by design using schedulability analysis (is there space for everybody?).
- Since priorities are dynamically assigned, resources can be allocated in order to optimize overall performance.



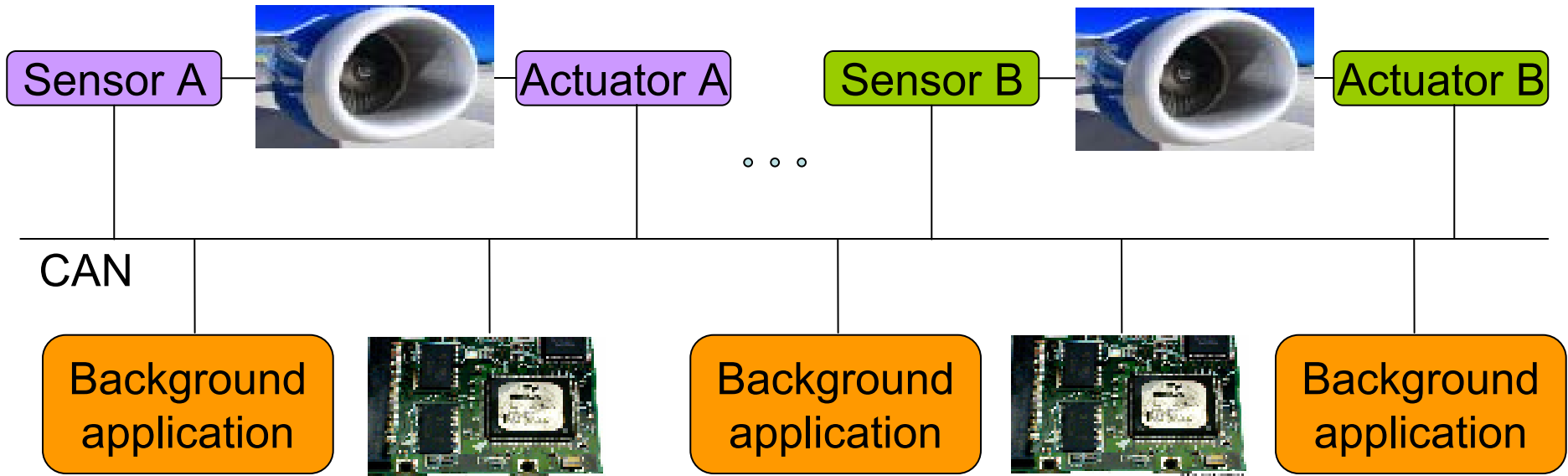
FlexRay network

- FlexRay = TDMA + CAN



- Time-triggered windows for 'hard' messages.
- But control messages do not need to be hard! Control messages are allocated in the dynamic segments.
- Priorities are assigned as in the CAN protocol. Spare bandwidth can be allocated to background applications.

Example: Jet engine compressors



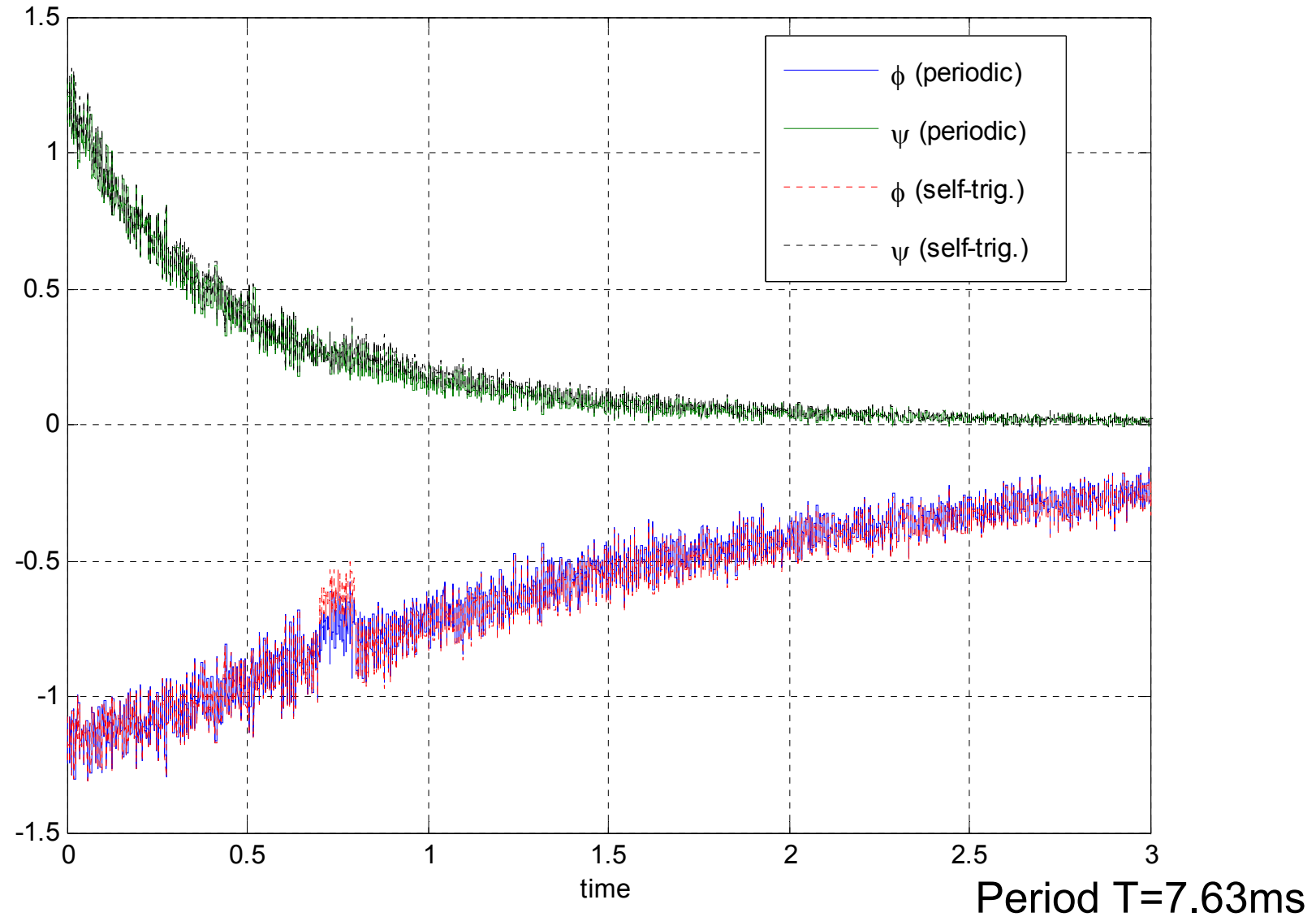
ψ = pressure rise,
 ϕ = mass flow,
 ϕ_T = throttle mass flow.

$$\begin{cases} \dot{\phi} = -\psi - \frac{3}{2}\phi^2 - \frac{1}{2}\phi^3 \\ \dot{\psi} = \frac{1}{\beta^2}(\phi - \phi_T) \end{cases}$$



Example: Jet engine compressor

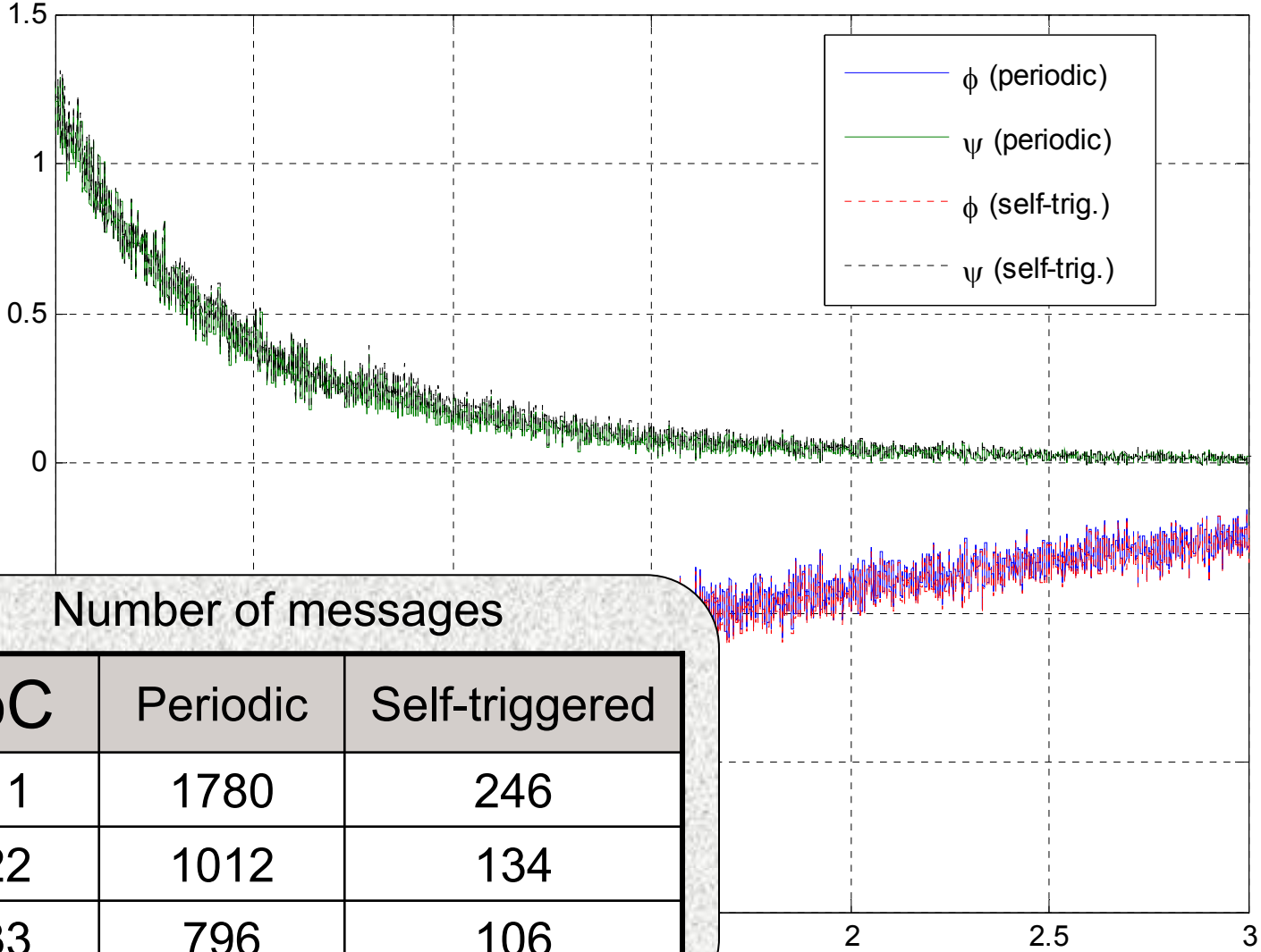
Evolution of the states for periodic and self-triggered strategies





Example: Jet engine compressor

Evolution of the states for periodic and self-triggered strategies



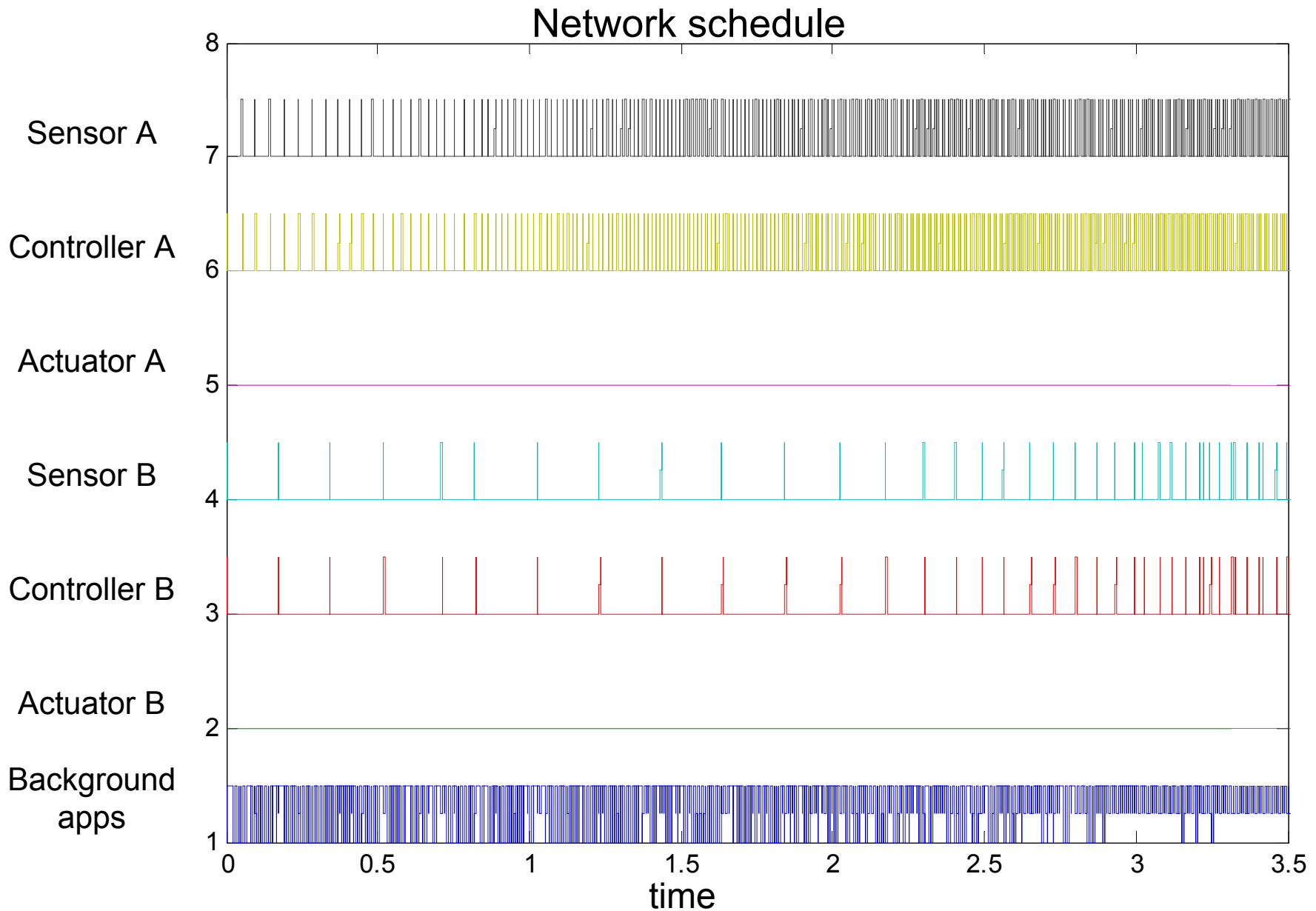
Number of messages

QoC	Periodic	Self-triggered
0.11	1780	246
0.22	1012	134
0.33	796	106

Period T=7.63ms



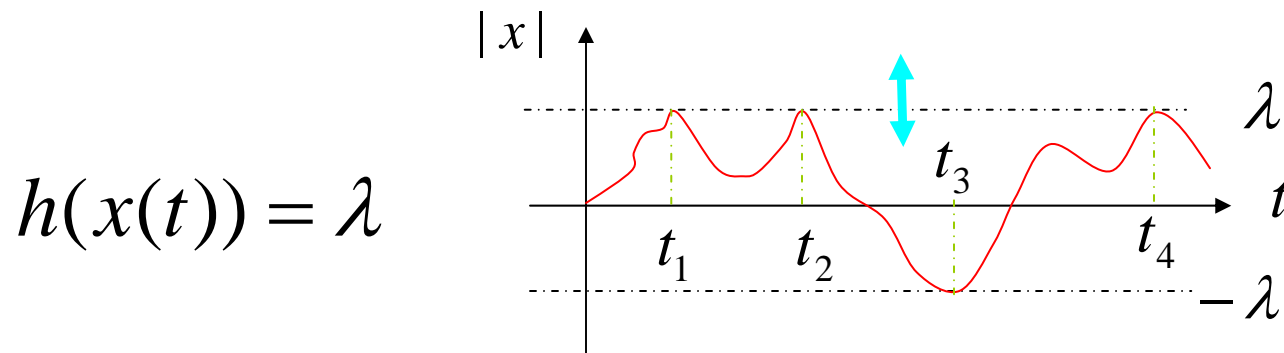
Example: Jet engine compressor





Conclusions

- The proposed model allows for a dynamic allocation of resources. Co-design is possible since we have an analytical expression relating performance and demand of resources:
 - If there is overload in the network, the control loop can reduce communication at the expense of a decrease in performance bandwidth.
 - If more bandwidth is available, the control loop can exchange more messages to achieve a better performance.



Related topics



- Research in embedded control systems:
 - Feedback control in energy aware systems
 - Automata-based real time scheduling



Anna Davitian

Sponsors:

- Fulbright Program
- Mutua Madrileña Automovilista (Spain)
- NSF EHS Award 0717188

Self-triggered control

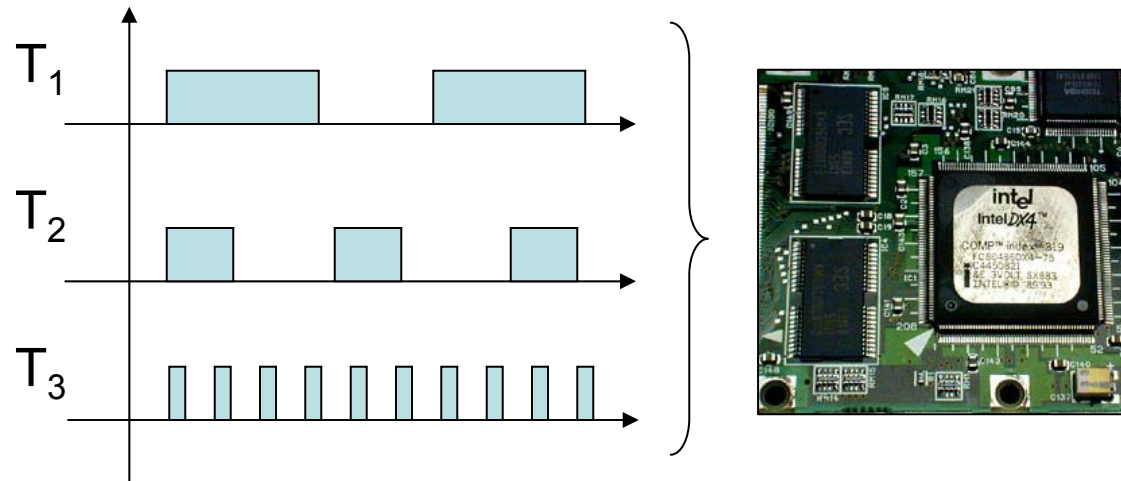


Questions



Application: Real-Time Scheduling

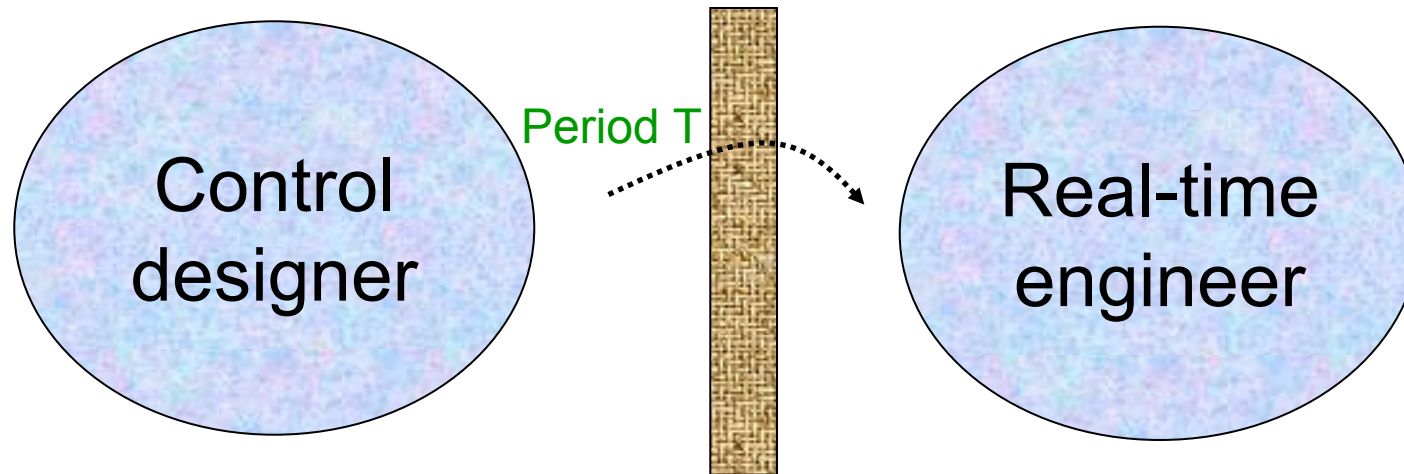
Allocation of resources:



Each task $T_i = (C_i, D_i)$ $\left\{ \begin{array}{l} C_i \equiv \text{Computation time} \\ D_i \equiv \text{temporal constraints} \end{array} \right.$

In addition, control tasks present extra requirements: $\left\{ \begin{array}{l} \text{Jitter} \\ \text{Input-output latency} \end{array} \right.$

Application: Real-Time Scheduling



But, what are the real-time requirements of control tasks?

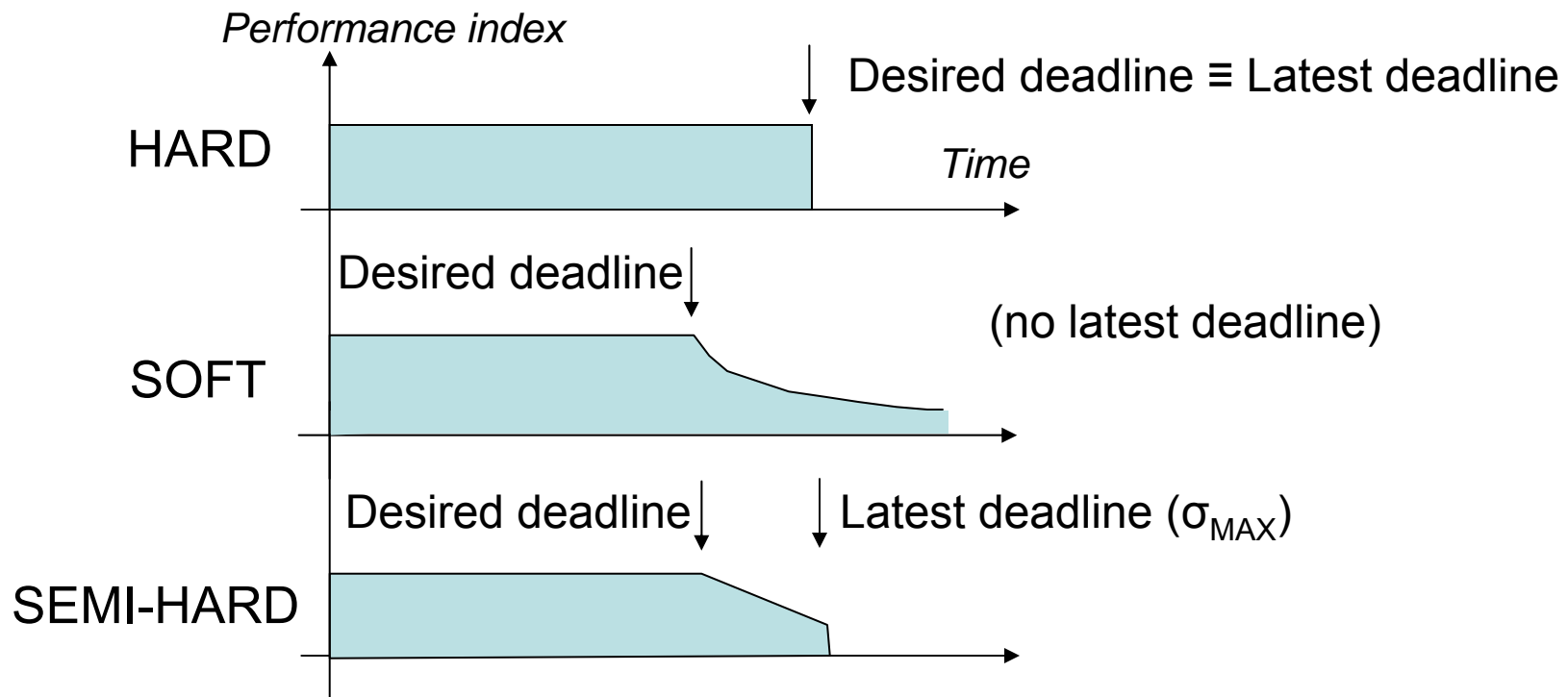
$$\frac{|e_j|}{|x_j|} \leq \frac{\sigma}{4}, \quad \frac{\sigma}{4} \in (0; \frac{\sigma}{4}_{\max}]$$

- Self-triggered implementations lead to aperiodic control tasks.
- Since less executions are required, more spare time is available.
- Larger values of σ' imply longer times.

Application: Real-Time Scheduling



- Parameter $\sigma' \in (0, \sigma_{MAX}]$ captures the tradeoff between rate of convergence (performance) and usage of resources: control tasks do not have to be considered as hard real-time tasks.



Application: Real-Time Scheduling



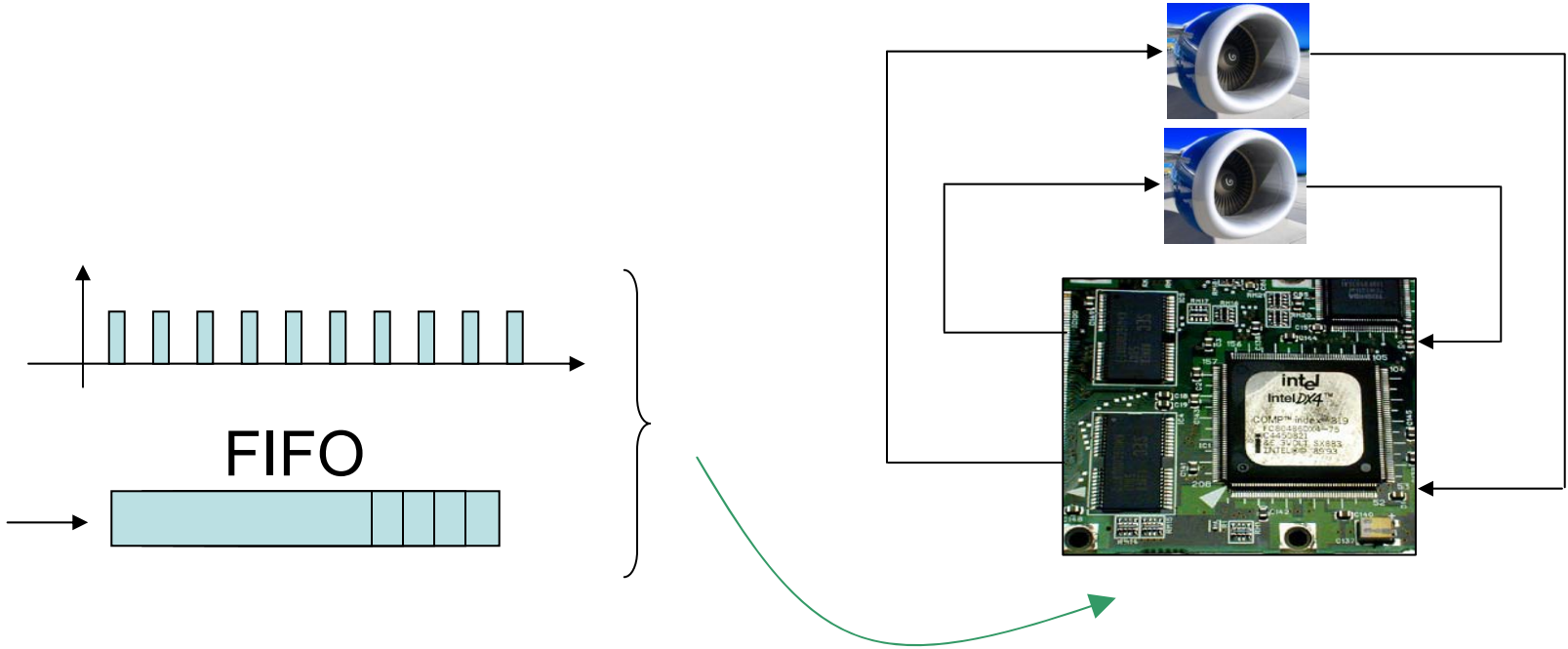
Merging all these ideas, and using the Control Server policy [Cervin&Eker 03] we were able to design a scheduling policy that deals with self-triggered tasks, reduces input-output latency and allocates the spare time to soft tasks.

- ❑ In self-triggered control the next deadline is known at the previous sample, facilitating the scheduling. However, in event-triggered control maximum priority has to be assigned to the controller.
- ❑ Since there exist an analytical expression that determines the tradeoff between performance and computations, real-time scheduling codesign is possible: the scheduler can modify online the value of σ' to adjust the amount of resources allotted to the controller (e.g., in case of overload).

Application: Real-Time Scheduling

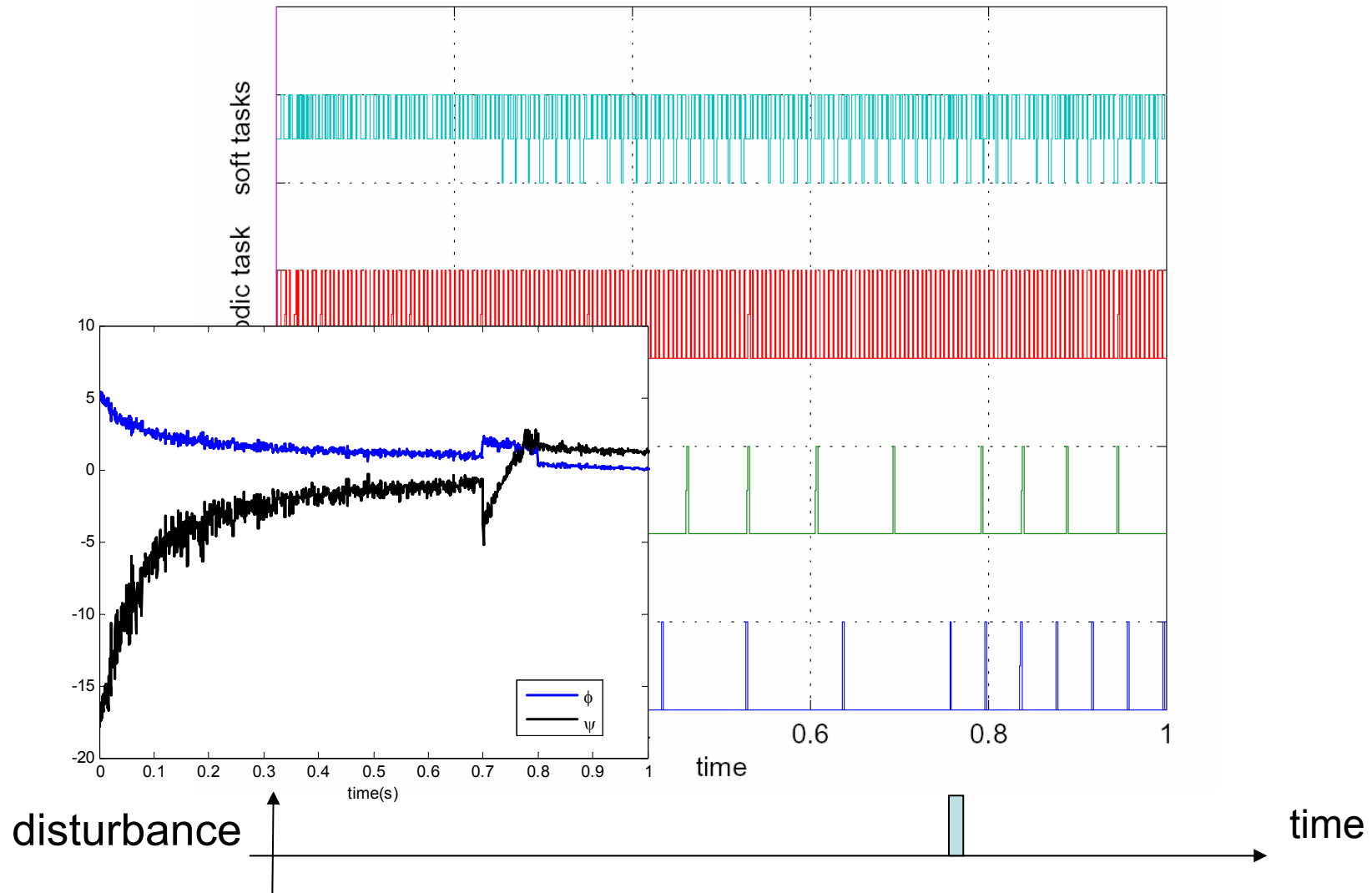
Example:

Computational unit in charge of the control of two engine compressors, plus a hard periodic task and several soft aperiodic tasks managed through a FIFO.





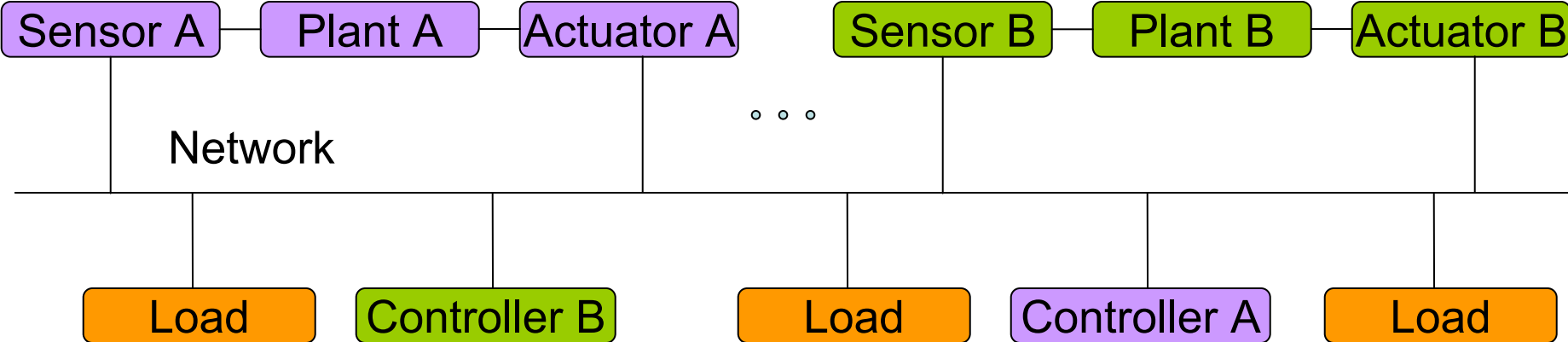
Application: Real-Time Scheduling





Application: Networked Control Systems

Self-triggered control will also reduce the communication between actuators, sensors and controllers. Now the shared resource is the network bandwidth.



Eternal dilemma: time-triggered or event-triggered protocols?



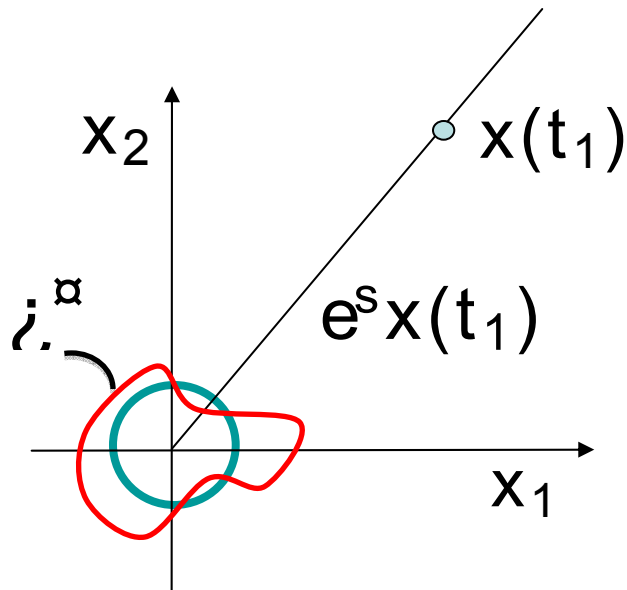
Application: Networked Control Systems

- Schedule is now nonpreemptive.
- Priorities are assigned so that control performance is guaranteed.
- Closing the loop is now a two-step process: message from sensor to controller and message from controller to actuator.



Current work

- In this presentation we have explained how execution times vary along rays. Recently we were able to model how times vary across rays.

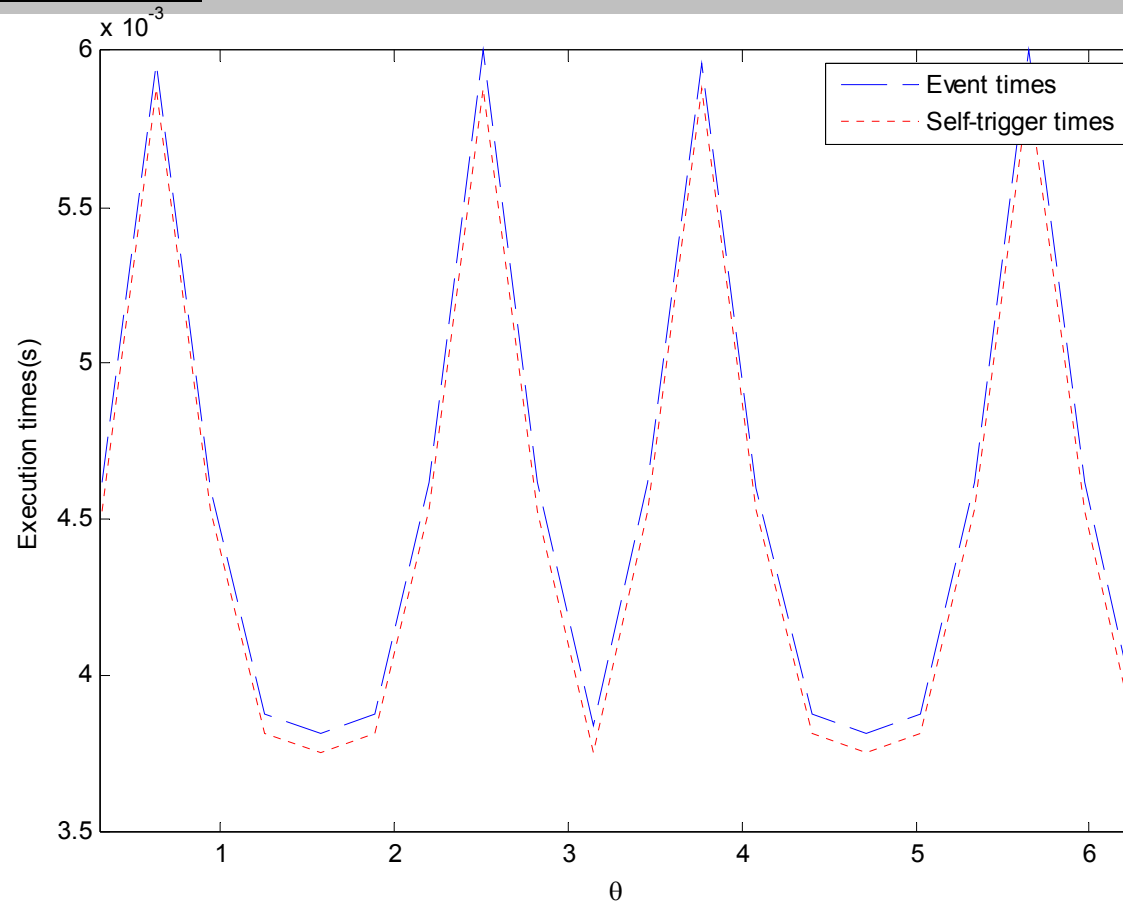


$$\tau(x(t_i)) = \underbrace{f(x(t_i))}_{\text{exact}} \underbrace{\tau^*}_{\text{conservative (in 2 ways)}}$$

Instead of using a sphere as a reference set, we use manifolds where times remain constant.



Current work



- Self-trigger conditions for general nonlinear systems have been developed as well (but computationally inefficient).
- Results can be used to estimate maximum delays and drop outs.



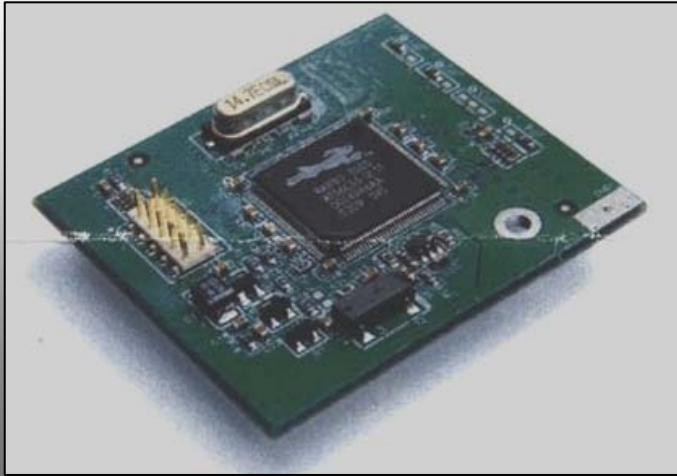
Conclusions

- Under self-triggered implementations, the next inter-execution time is given by a function of the last measurement and the desired performance.
- Self-trigger conditions were developed for homogeneous systems and polynomial systems.
- The parameter σ captures the tradeoff between rate of convergence and usage of resources.

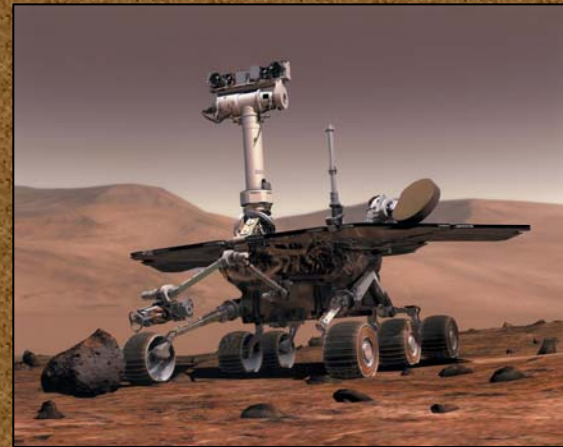
$$\frac{|e_j|}{|x_j|} \leq \frac{3\sigma}{4}$$

Introduction: applications

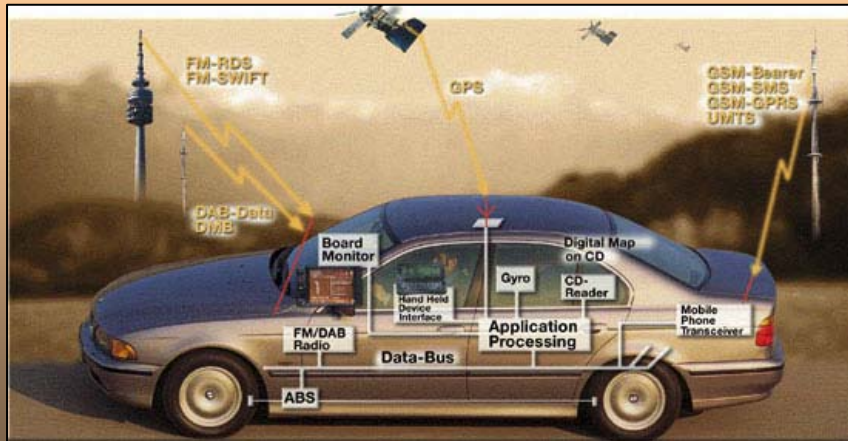
Real-time systems



Energy-aware systems



Distributed control systems



Chemical processes

