UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

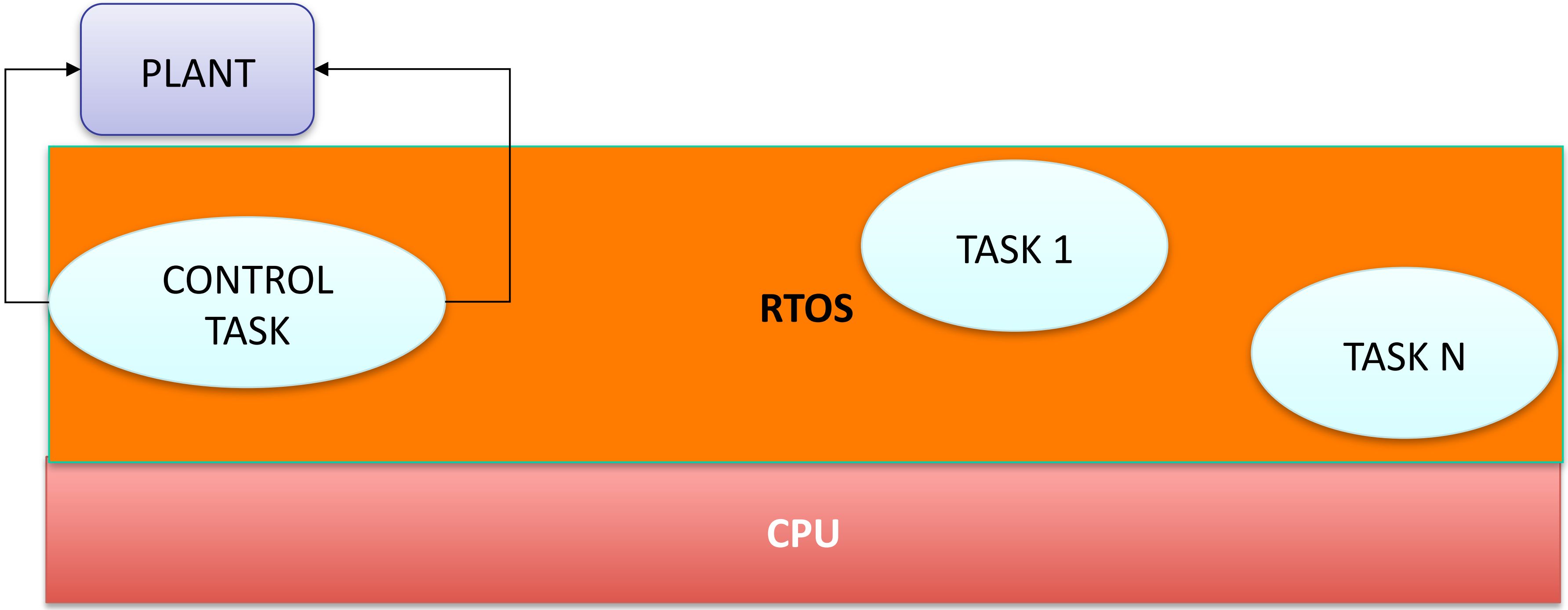
When multimedia meets control: Making the case for soft real—time in control

Daniele Fontanelli, Luca Abeni, Luigi Palopoli

Dipartimento di Scienza e Ingegneria dell'Informazione
Università di Trento,
Italy

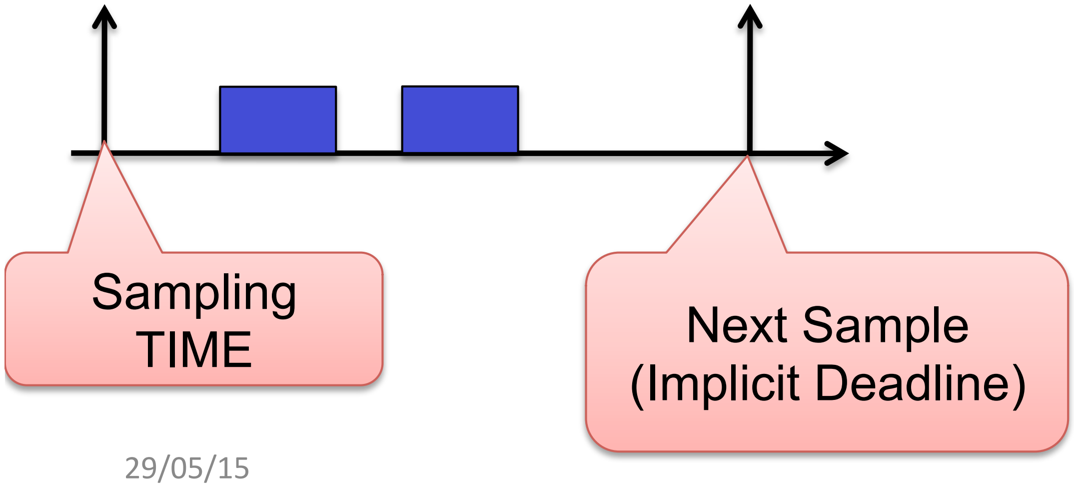
The Framework

- One computing board is used to implement a control task
- It is ***shared*** with other computing tasks
  - Some of them ***can be critical (e.g., other control tasks)***
- The use of the CPU is mediated by a real-time operating system



The Problem

- The presence of multiple (high priority) tasks has two effects:
  - restricts the number of computations that the control task can perform in unit time
  - introduces time varying delays



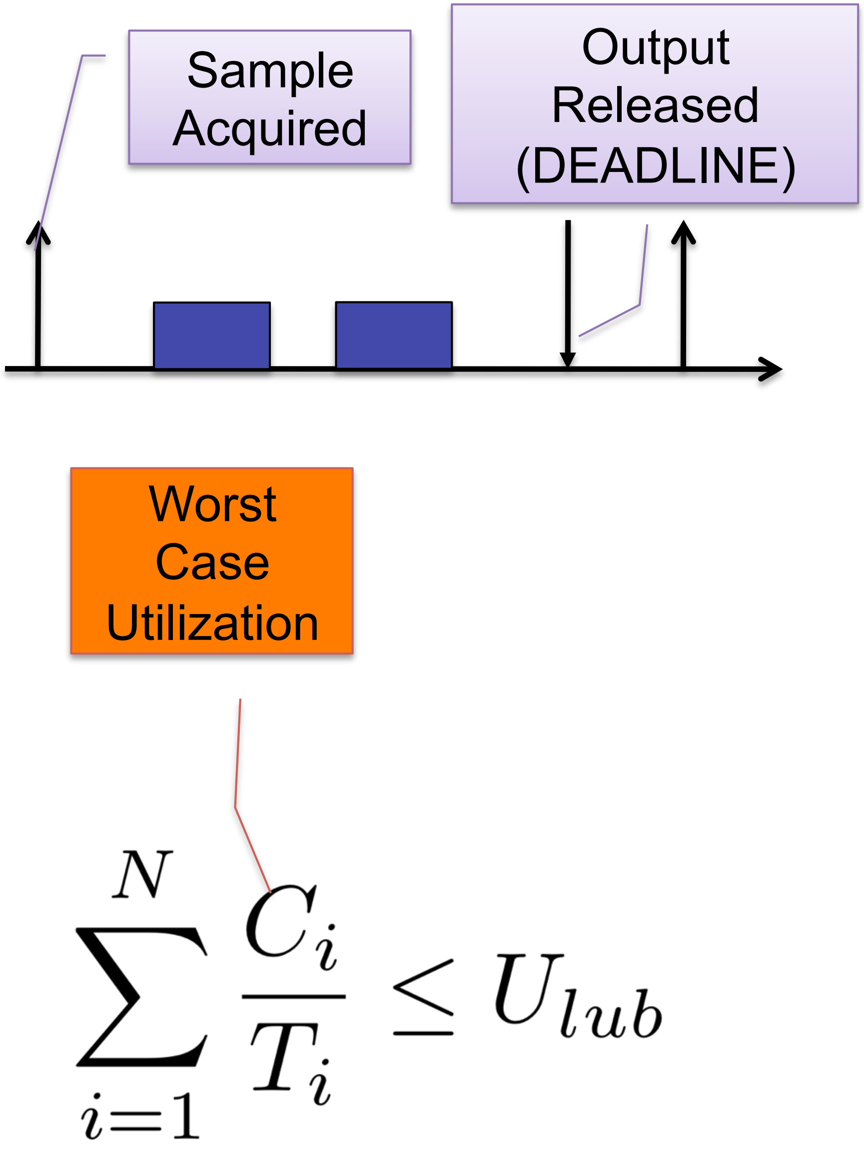
TWO CONFLICTING GOALS

Ensure high rate and a regular execution to the control application

Maximise the number of applications that can be hosted on the board

Classic Approach

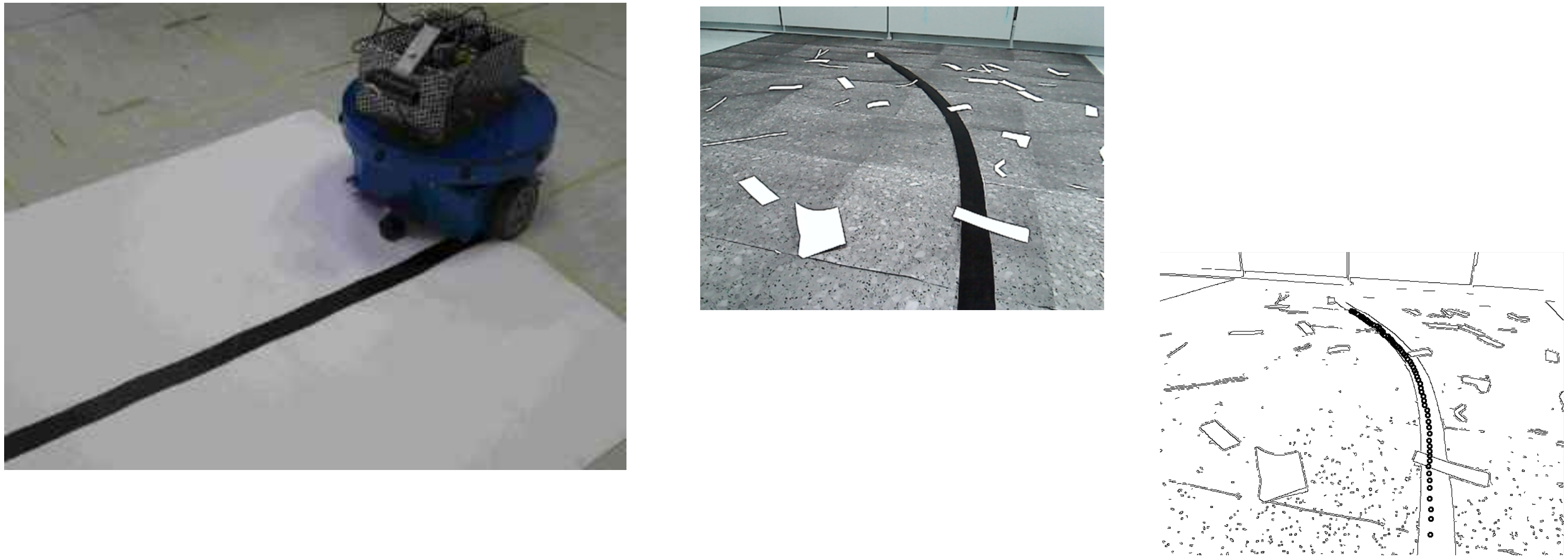
- **Contract**:
  - Platform: Fixed sampling time and fixed delay
  - Control Application: control performance by delay compensation
- Platform QoS guaranteed by:
  - Time Triggered model of computation
  - Fixed or dynamic scheduling priorities to enforce the timing interface



$$\sum_{i=1}^{N} \frac{C_i}{T_i} \leq U_{lub}$$

Limitations

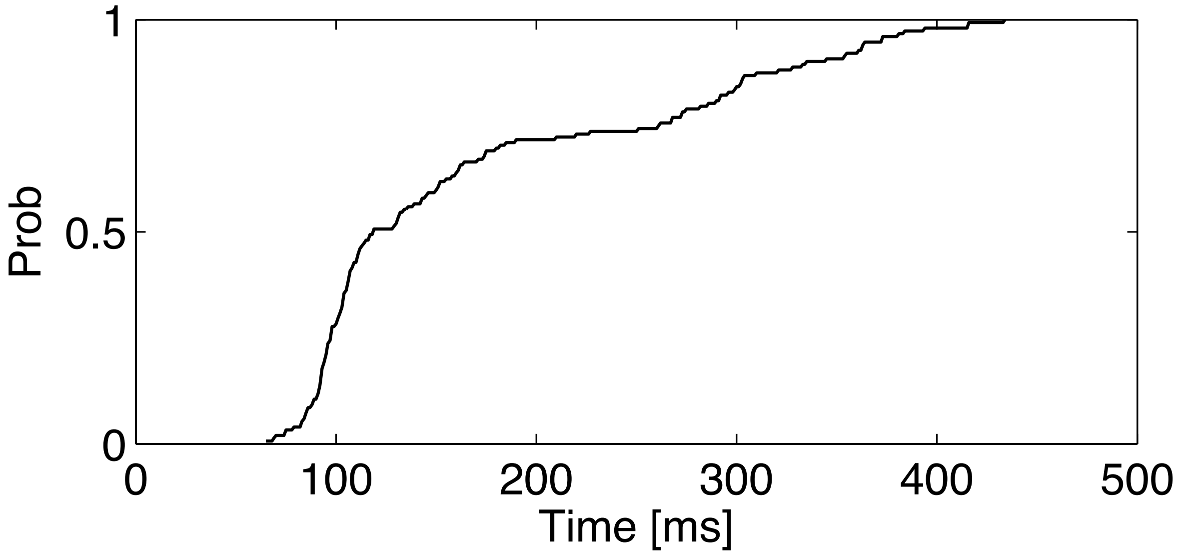
- The computation time can change very much for certain types of applications



- If we use multimedia streams, the computation time highly depends on the number of artefacts in the scene

Computation time

- CDF of the computation time of the lane detection mechanism (statistics collected on a beagle bone).



AVERAGE AROUND 220MS, MAXIMUM AROUND 500MS

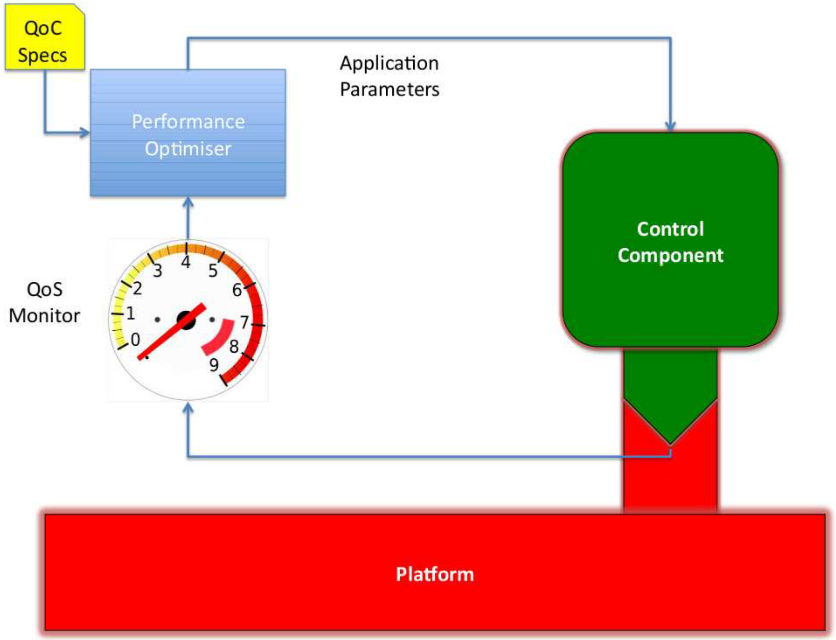
- With a worst case design resources can be significantly underutilised

Main idea

- The idea is to define the control period that is less than allowed by the WCET based guaranteed
- For example, the period can be defined on the mean execution time
- As a consequence, the CPU may sometimes be unable to respect the deadline (i.e., the period)
- How to deal with this problem?

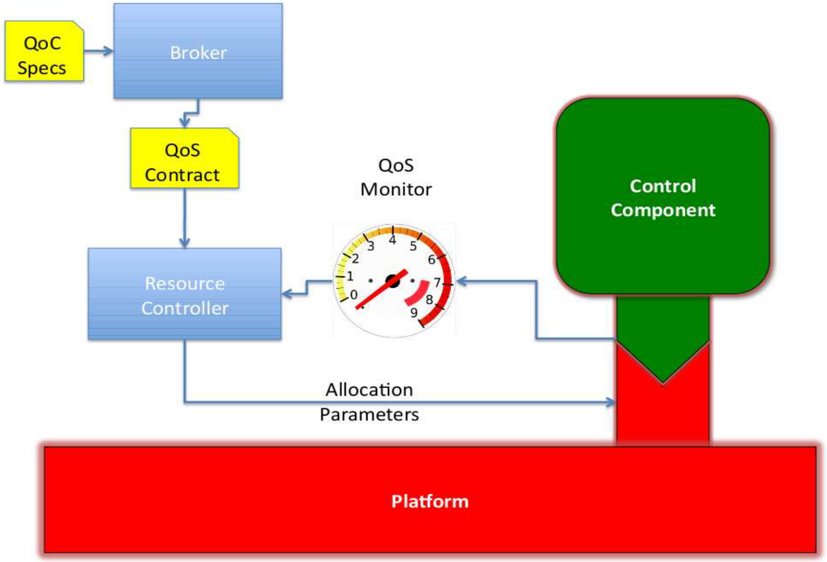
- *Changing the control law at run-time to adapt to the QoS offered by the platform*
- *Changing the platform parameters to obtain a QoS that meets the desired QoC specification*

Acting on control



- The QoS is a given, QoC is achieved by acting on the control parameters
  - Event-based controllers (Tabuada et al., …)
  - Any-time controllers (Fontanelli et al., Gupta et al., …)

Acting on the QoS


QoC
Specs
Broker
QoS
Contract
QoS
Monitor
Control
Component
Resource
Controller
Allocation
Parameters
Platform

- For real-time tasks, the controller is a given, while the scheduler can be modified on-line
- This is the approach taken in this talk

Problem

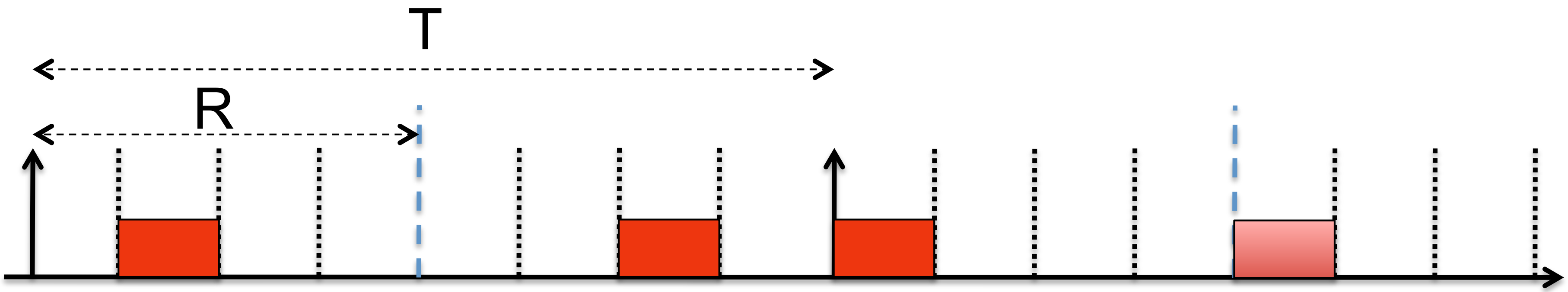
- With classic real-time scheduling only static priorities can be managed
- It is an all-or-nothing approach where we don't really control the amount of CPU each application receives
    - No real performance guarantees with overloaded CPUs

Soft real—time approaches and control

- We have shown that we can have substantial resource savings without sacrificing control performance
- And we can do this with <u>certifiable</u> performance guarantees by combining
  - RESOURCE RESERVATION SCHEDULING
    - ✓Accurate model of QoS as a function of the scheduling parameters
  - AN APPROPRIATE MODEL OF COMPUTATION
    - ✓Link between QoS and QoC

Resource reservation Scheduling

- The time-line is split into server periods of length *R*
- The application is guaranteed to receive a budget *Q* of the computation time (exact for hard reservations)
- Hence, we can predict its evolution regardless of the other tasks



Q=1, R=4, T = 2R, C_1 = 3

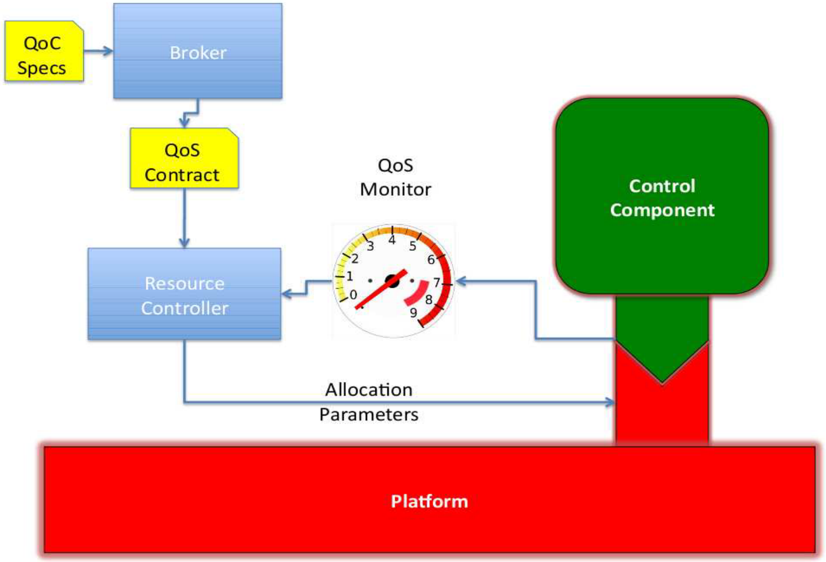
$$B = \frac{Q}{R}$$ is the bandwidth (fraction of the processor)

Correctness of the algorithm guaranteed if $$\sum_{i=1}^{n} B_i \leq 1$$

QoS Models

Mapping QoS to scheduling parameters

- In our setting, the QoS of the platform is specified by the delays introduced in the computation
- Because the computation time is assumed stochastic, so will be the delays
- We will therefore study the distribution of the scheduling delays and expose their link with the scheduling parameters



In order to do so, we need to address the direct and the inverse scheduling problem.

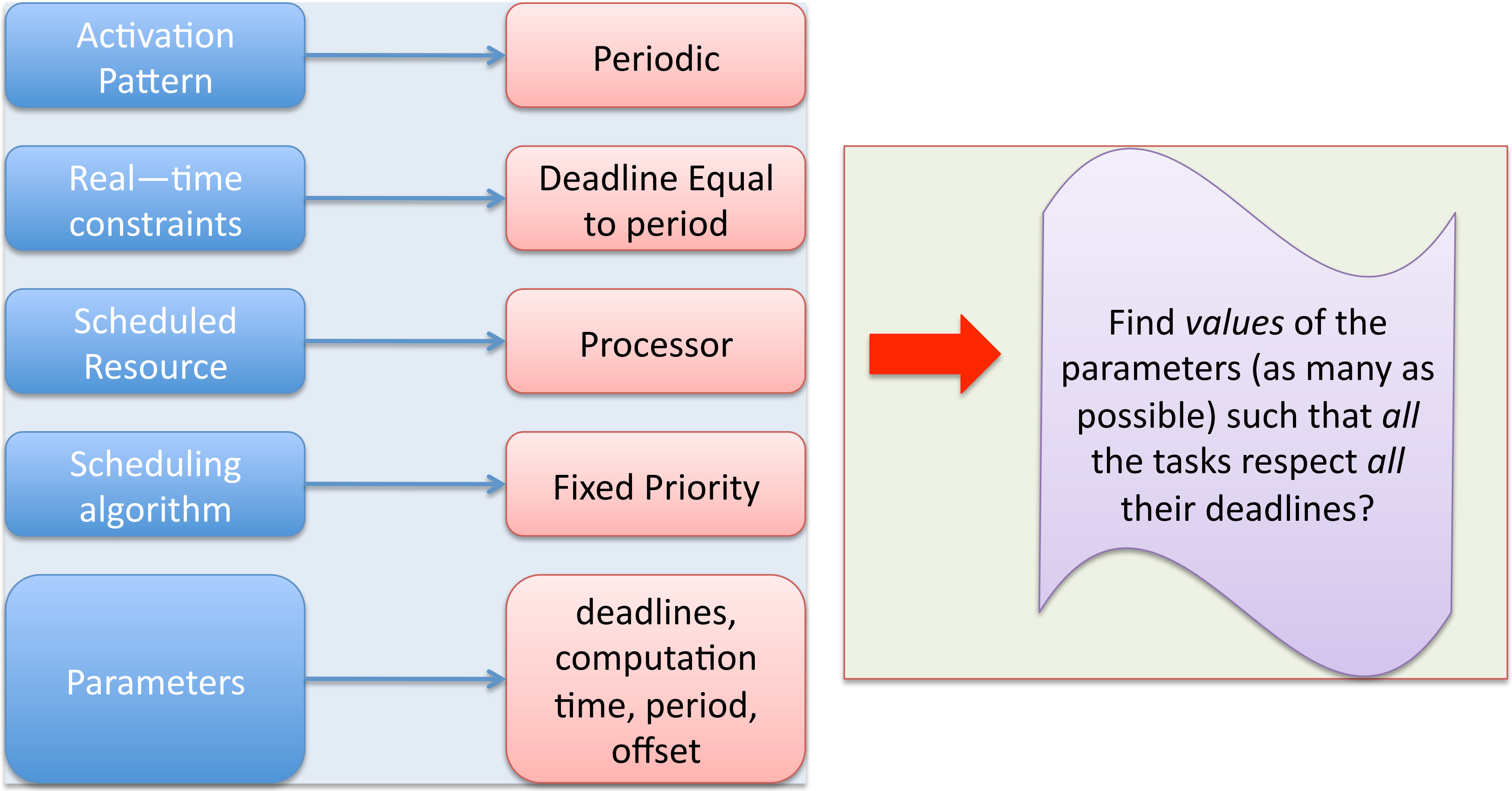
The Direct Real—time scheduling problem

- Given
    - a set of concurrent activities (task), each one associated with
        - ✓ an activation pattern
        - ✓ some temporal constraints
    - a set of scheduled resources (typically processors)
    - a scheduling algorithm
    - a set of parameters
        - ✓ computation time
        - ✓ inter--arrival time
        - ✓ scheduling parameters
- Say
    - If all the task meet their temporal constraints

The Inverse Real—time scheduling problem

- Given
  - a set of concurrent activities (task), each one associated with
    - ✓ an activation pattern
    - ✓ some temporal constraints
  - a set of scheduled resources (typically processors)
  - a scheduling algorithm
  - a set of parameters
    - ✓ computation time
    - ✓ inter--arrival time
- Find
  - The *set of scheduling parameters* that produce a feasible schedule

One possible instance of the problem



Activation Pattern	Periodic
Real—time constraints	Deadline Equal to period
Scheduled Resource	Processor
Scheduling algorithm	Fixed Priority
Parameters	deadlines, computation time, period, offset

Find *values* of the parameters (as many as possible) such that *all* the tasks respect *all* their deadlines?

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Available solutions

- To solve the inverse problem effectively we need analytical expressions delimiting the region of feasible parameters.

- Utilisation test
  - Liu and Layland 1973

$$\sum_{i \in \text{tasks}} \frac{C_i}{T_i} \leq U_l$$

- Time Demand Analysis
  - Lehoczky, Sha et al 1989

$$\exists t, \;\; L_i(0,t) \leq t$$

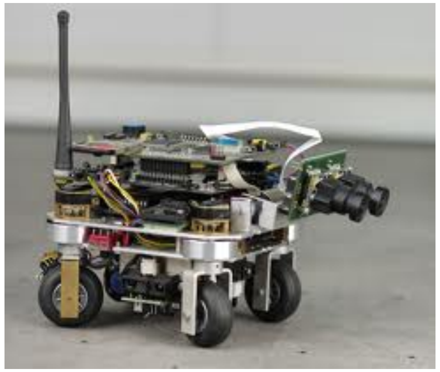
Can be used to derive a set disjunction of polyhedral regions in the parameter space
(Bini 2004)

Soft real—time sytems

- The research on hard real—time systems has produced mature and effective methods
- Some of them produce analytic expressions that solve the inverse *problem*
  - *can be effectively used in system design*
- However, many recent real—time systems *are not* hard real—time
  - A timing violation is not a big issue as far as the anomaly is kept in check
- So these methods *are not* applicable

Examples

WEB TV



Visual Control

Automotive Infotainment

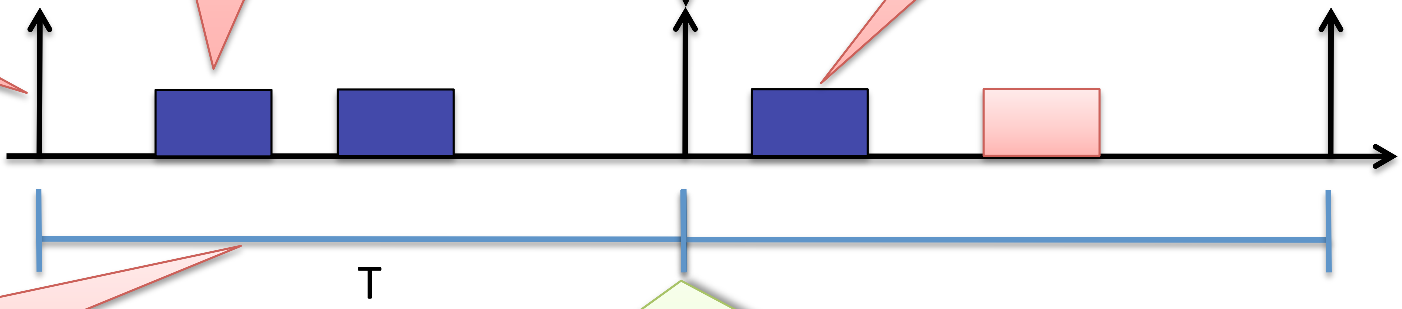
Probabilistic Guarantees

- For soft real—time systems, the performance specifications can be effectively stated in probabilistic terms
- In essence we can associate with each task a probability of respecting the deadline
- The *timing requirement* we want to enforce is to respect the deadline with a given probability

The current state of the art offers methods for (long) numerical computation.

- Hard to use online (for admission tests).
- No analytical tool for the inverse problem.

Our real—time task model



A task is a stream of jobs

Computation time c_k is a stochastic process with distribution
$$U(c) = \mathrm{Prob}\left\{c_k = c\right\}$$

Deadlines are set equal to the period

A job that finishes after the deadline runs to completion

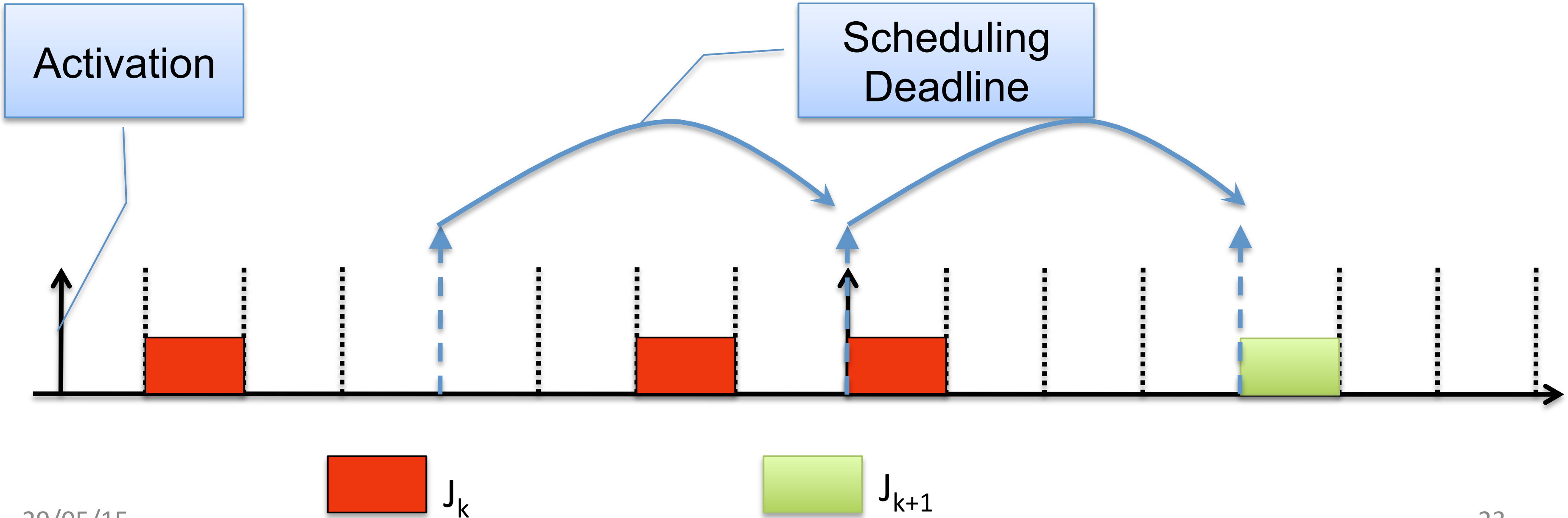
Activations are periodic

The requirement is to respect the deadline with probability $\pi^{(0)}$

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

CBS Scheduling

- We adopt a scheduler, the CBS, complying with the resource reservation paradigm
- Each scheduled task is associated with a pair (Q^s, T^s), meaning that is will receive Q^s units of computation every server period T^s
- Each job is associated with a scheduling deadline d^s which is used to decided the CPU allocation using an EDF paradigm and with a budget q^s
  - q^s is initally set equal to Q^s and is depleted while the task executes
  - when the budget is depleted, the scheduling deadline is postponed by T^s



Key Observations

- **Observation 1**: the temporal isolation property enables us to model the evolution of the tasks independently

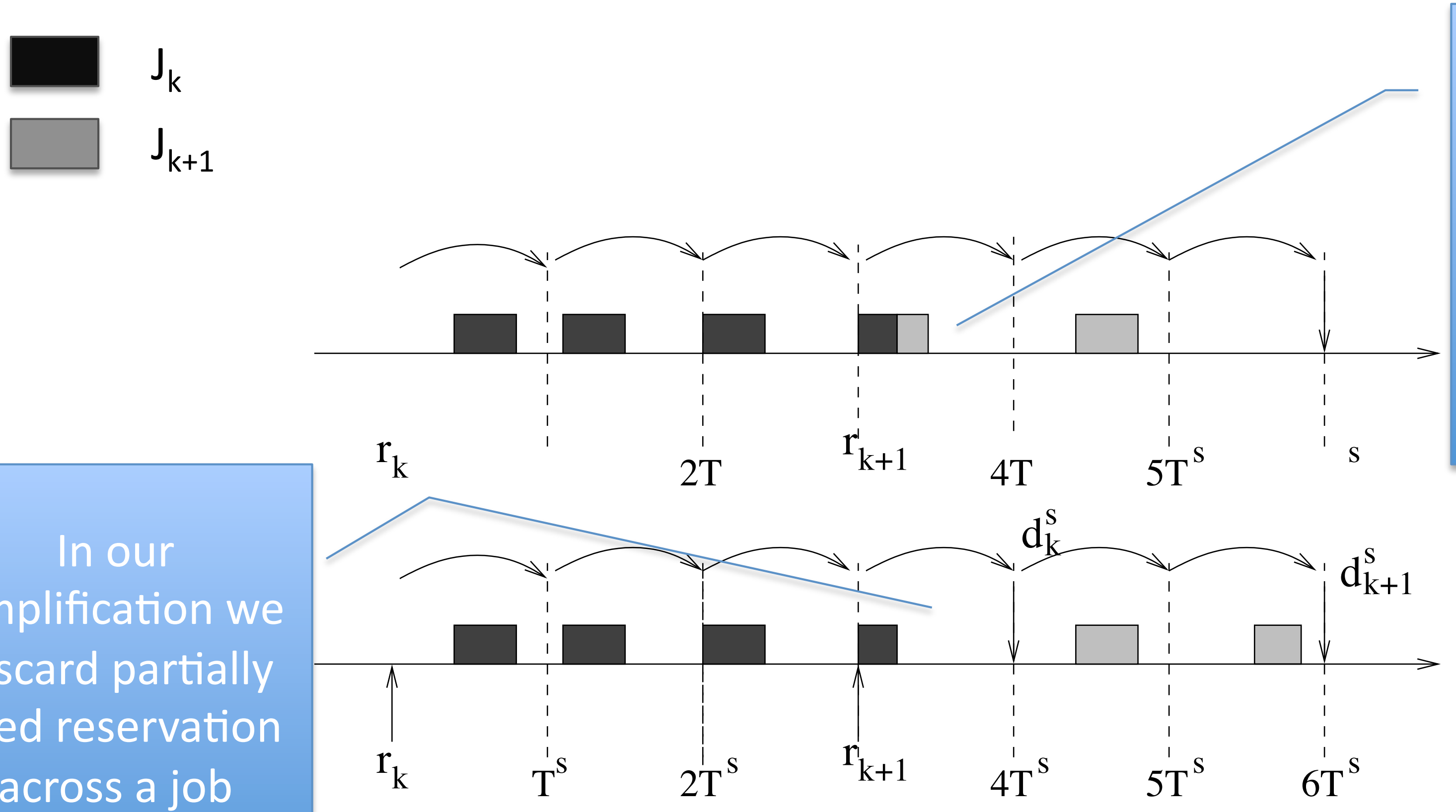
- **Observation 2**: the latest deadline used by a job d^s_k is an upper bound of its finishing time
  - Therefore:
    - ✓ $\delta_k = d^s_k - r_k$ is an upper bound of the response time
    - ✓ The probability of a deadline miss is $\Pr\{\delta_k \leq NT^s\}$

- **Observation 3**: We can assume that the task receives only the guaranteed bandwidth
  - It is then possible to find a model for the evolution of dk

$$\delta_{k+1} \leq \max\{0,\, \delta_k - NT_s\} + \lceil \frac{c_{k+1}}{Q^s} \rceil T^s$$

Price to pay

- we have introduced some degree of conservativeness

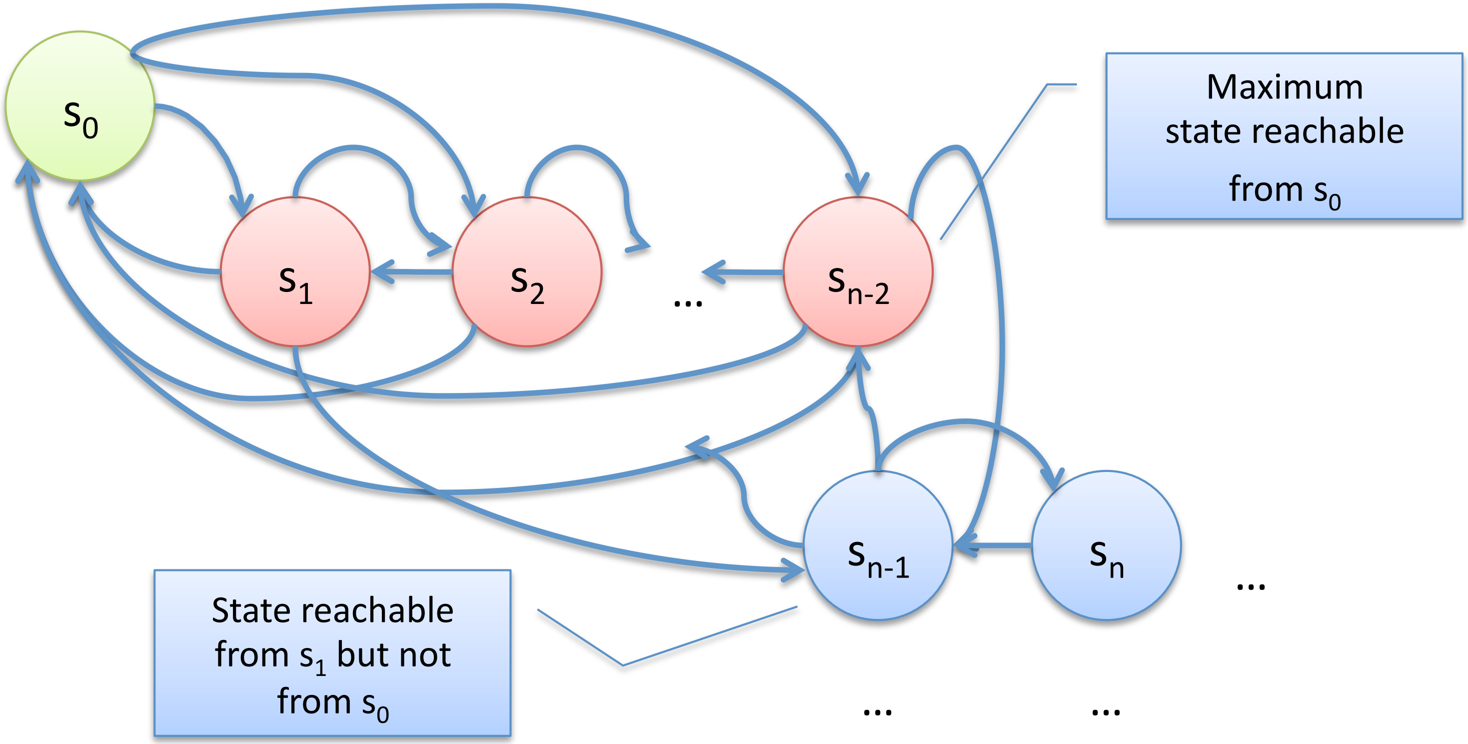


With the CBS a reservation across two job activations partially used by job k can be reused by job k+1

In our simplification we discard partially used reservation across a job activation

Modelling the system as a Markov Chain

- We can define a Markov Chain describing the system
  - State 0: $\delta_k \leq NT^s$
  - State i: $\delta_k = NT^s + iT^s$



Maximum state reachable from s_0

State reachable from s_1 but not from s_0

Matrix form for probability evolution

- The probability evolution can be expressed by a matrix expression consisting of an infinite number of elements

$$\pi(h+1) = \begin{bmatrix} \pi^{(0)}(h+1) \\ \pi^{(1)}(h+1) \\ \dots \end{bmatrix} = \pi(h)P$$

$$p_{i,j} = \Pr\left\{\delta(h+1) = NT^s + j | \delta(h) = NT^s + i\right\}$$

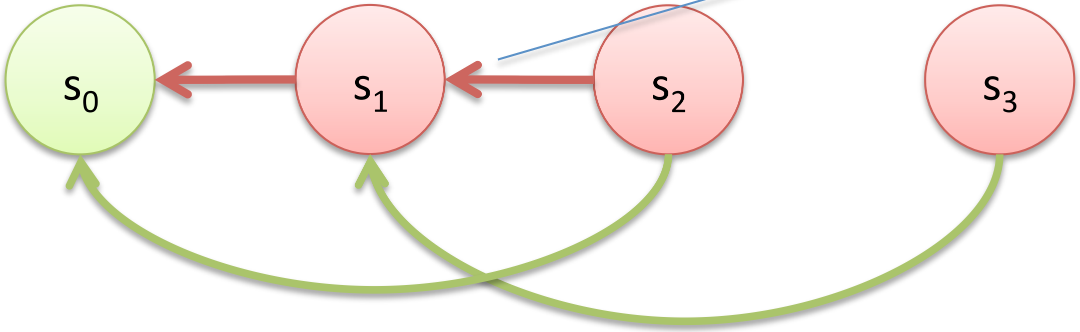
- The transition matrix can be easily computed as follows:

$$p_{i,j} = \begin{cases} U'\left((N-i+j)Q^s\right) & \text{if } i > 0 \\ U'\left((j+1)Q^s\right) & \text{if } i = 0 \end{cases}$$

$$U'\left(kQ^s\right) = \Pr\left\{kQ^s \geq c > (k-1)Q^s\right\}$$

Facts

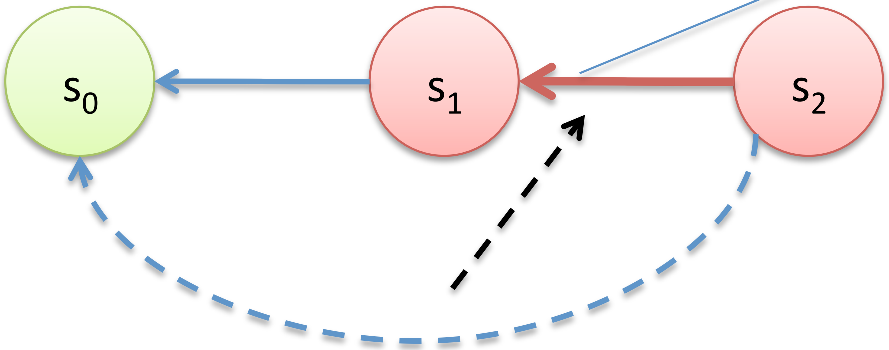
- Fact 1: the transition probability only depends on the difference between i and j and not on their values
    - The transition matrix has a recursive structure where each row is obtained by shifting to the right the previous one and padding with zeros



EXAMPLE: The probability of going from s_2 to s_1 is the same as from s_1 to s_0

Facts

- Fact 2: we can obtain a simpler Markov Chain by collapsing all the backward arcs into the arc going backwards to the closest neighbour



EXAMPLE: The probability of going from s_2 to s_0 is zeroed and summed to the probability of going from s_2 to s_1

Quasi Birth death processes

- The transition matrix

$$P = \begin{bmatrix} b_0 & a_{n-3} & a_{n-4} & \dots & a_1 & a_0 & 0 & \dots & \dots & \dots\dots \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_2 & a_1 & a_0 & 0 & \dots & \dots\dots \\ 0 & a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_2 & a_1 & a_0 & 0 & \dots\dots \\ 0 & 0 & a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_2 & a_1 & a_0 & 0\dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\dots \end{bmatrix}$$

qualifies the system as a QBDP.

From the second row, each row is obtained from the previous by shifting to the right and padding with 0

Main result

- For a birth death process the following holds

$$P = \begin{bmatrix} b_0 & a_0 & 0 & 0 & 0 & 0 & 0 & \dots & \dots & \dots\dots \\ a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & \dots & \dots\dots \\ 0 & a_2 & a_1 & a_0 & \dots & 0 & 0 & 0 & 0 & \dots\dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\dots \end{bmatrix} \rightarrow \pi^0(\infty) = 1 - \frac{a_0}{a_2}$$

- Our generalisation

$$P = \begin{bmatrix} b_0 & a_{n-3} & a_{n-4} & \dots & a_1 & a_0 & 0 & \dots \\ b_1 & a_{n-2} & a_{n-3} & \dots & a_2 & a_1 & a_0 & \dots \\ 0 & a_{n-1} & a_{n-2} & \dots & a_3 & a_2 & a_1 & \dots \\ 0 & 0 & a_{n-1} & a_{n-2} & \dots & a_3 & a_2 & \dots \\ 0 & 0 & 0 & a_{n-1} & a_{n-2} & \dots & a_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \ddots & \ddots \end{bmatrix} \rightarrow$$

$$\rightarrow \pi^{(0)}(\infty) = 1 - \frac{a_{n-3}}{a_{n-1}} - 2\frac{a_{n-4}}{a_{n-1}} - \dots - (n-2)\frac{a_0}{a_{n-1}}$$

Probabilistic Guarantees

- The theorem above leads us to an analytical formulation of the probabilistic guarantees

$$\pi(0)(\infty) \geq 1 - \sum_{j=1}^{n-2} (n-1-j) \frac{U'\left((N+n-j-1)Q^s\right)}{\sum_{h=0}^{N-1} U'\left(hQ^s\right)}$$

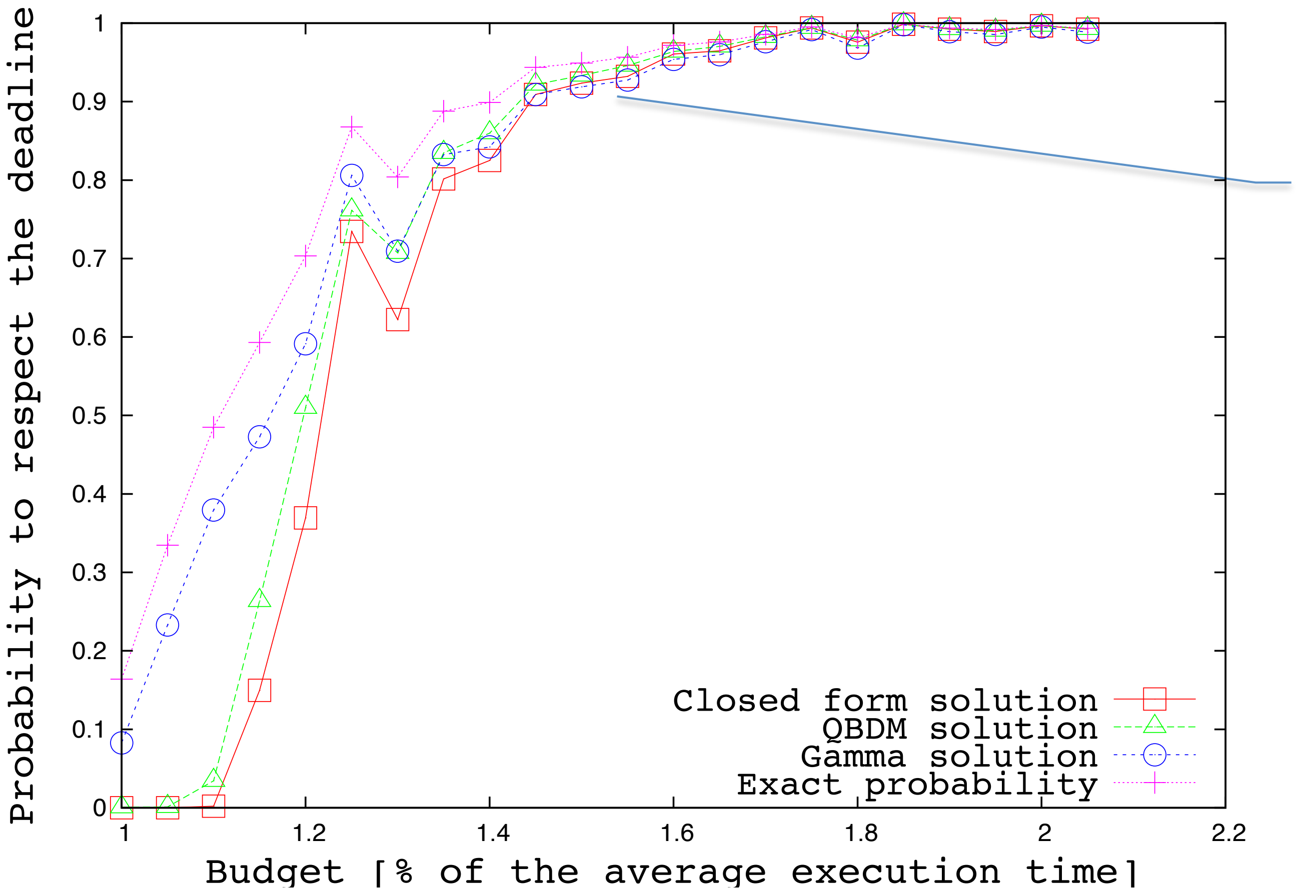
Analytically related to the distribution of the computation time

Budget

- Analytical bound usable both for the direct and for the inverse real—time scheduling problem with probabilistic guarantees

Quality of the bound (synthetic applications)

- We made extensive numeric evaluations to evaluate the quality
  - Data reported on a large number of synthetic distributions



The error becomes tiny when the probability of meeting the deadline is realistic

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Computation time (synthetic applications)

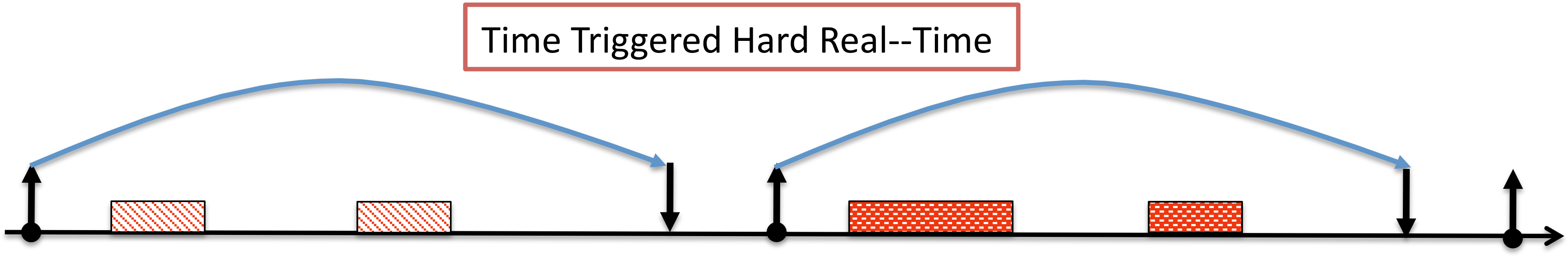
- The computation time is five orders of magnitude below the state of the art methods

Extensions

- The model presented above is a conservative one.
- We have recently shown that the same ideas are also applicable to a non conservative model but…. we have to give up the idea of closed form expressions.
- We are currently working on
  - non i.i.d. computation time
  - distributed computation

Models of Computation

Time Triggered



Time Triggered Hard Real--Time

Temporal Analysis:

Bandwidth

$$\frac{Q_s}{T_s} \geq \frac{C}{T}$$

Worst case Task Utilisation

LEGEND

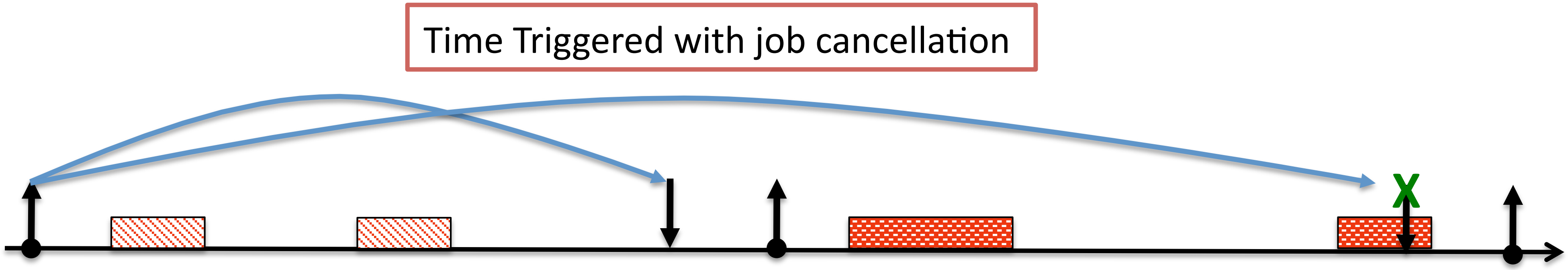
↑ Job Activation

↓ Output Release

⫰ Input Acquisition

↷ IO Relation

Time Triggered with job cancellation



Temporal Analysis:

Probability of completing within the deadline

$$\pi = \Pr\left(\left\lceil \frac{C_i}{Q_s} \right\rceil T_s \le T\right)$$

- The idea is very easy to implement and analyse…
  but the missed update can generate serious stability problems (especially for dynamic controllers)
- Outright cancellation can be too drastic

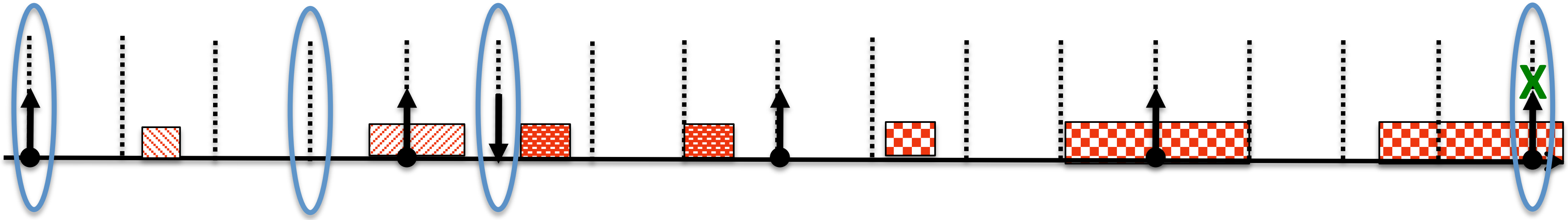
Soft real—time time--triggered

- Periodic Activation and Input Sampling

- Execution permitted beyond the deadline with output released on the first available interaction point

- IO operations only permitted upon specified interaction points. The interval R between two adjacent interaction point set equal to Ts

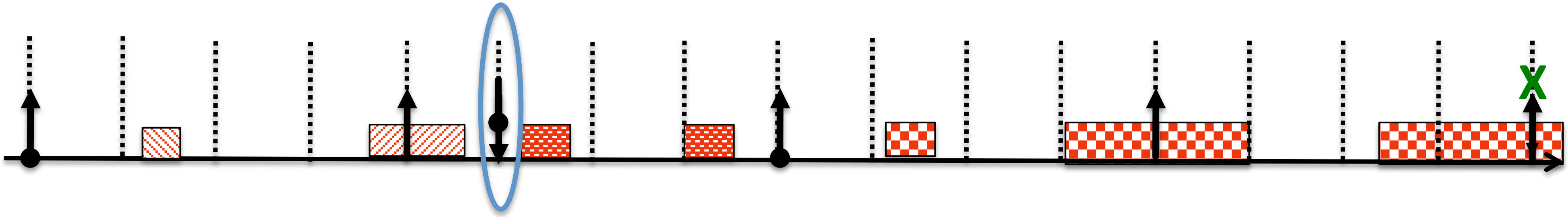
- If the delay goes beyond a threshold the job is cancelled



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

An optimisation

- If a job gets delayed the input sample can be deferred to when it starts



- This allows me to use fresher data **but** the task execution remains driven by the absolute time

Temporal Analysis

- If we schedule the task through a reservation, a close approximation for the evolution of the delays is given by the following dynamic model

Delays are the states (multiple of R)

$$D_{j+1} = S^{\star}(D_j) + \left\lceil \frac{c_{j+1}}{Q_{j+1}} \right\rceil - N$$

$$S^{\star}(x) = \begin{cases} \frac{\overline{\Delta}}{R} & \text{if } x \geq \frac{\overline{\Delta}}{R} \\ x & \text{if } \frac{\overline{\Delta}}{R} > x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Evolution Governed by Qi, and Ci

If Jobs go longer than a bound, they are dropped

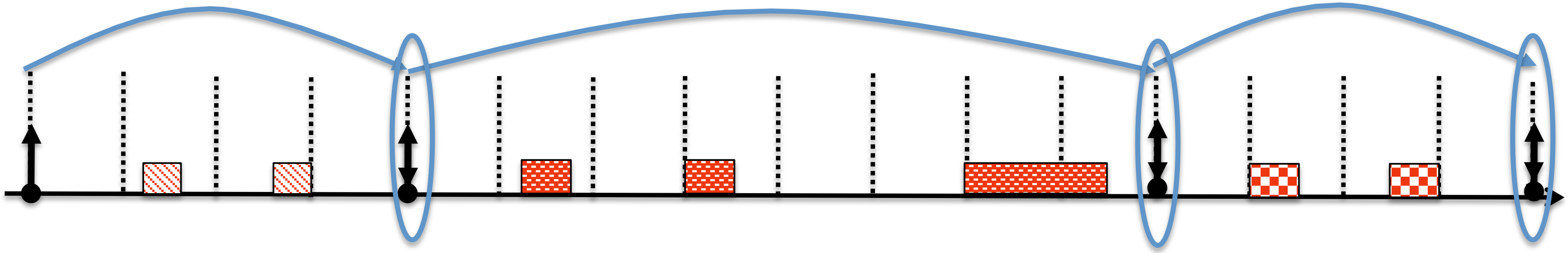
- Contrary to the simple case of time—triggered with job cancellation, we have a dynamic model (Markov Chain)
- The model is much more flexible than simple job cancellation
- But the analysis is difficult. When it is combined with the evolution of the system, we obtain a Markov Jump System for which only sufficient stability conditions exist

The Continuous Stream MoC

- When a job starts it is assigned a relative deadline. If the job completes within the deadline, its output is released, a new input acquired and a new job is started

- If a job execution goes beyond its deadline, no IO operations is permitted nor any job is started until the job completes. If the delay goes beyond a deadline the job is canceled

- When a new job starts, its delay is considered only with respect to its start time (which is not absolute).



Temporal Analysis

Theorem

- If we use the continuous stream model in combination with a resource reservation scheduler, each job is guaranteed to terminate within

$$a_j + \left\lceil \frac{C_j}{Q_s} \right\rceil T_s$$

Arrival time of the job

The response time of the job does not depend on the previous job but only on the computation time!!

The evolution of the delay simply reduces to:

$$D_j = \left\lceil \frac{C_j}{Q_s} \right\rceil T_s$$

Number of server periods containing Cj

Probabilities

- The process describing the delays is memory-less

$$\begin{array}{ll}\text{Finish before the deadline:} & \mu_N(Q) = \text{Prob}\left\{\left\lceil \frac{c_j}{Q} \right\rceil \le N\right\},\ N = \frac{P}{R}, \\ \text{Delay of } i \text{ reservation periods:} & \mu_i(Q) = \text{Prob}\left\{\left\lceil \frac{c_j}{Q} \right\rceil = i\right\}, \\ \text{Job Cancellation:} & \mu_o(Q) = \text{Prob}\left\{\left\lceil \frac{c_j}{Q} \right\rceil > N_r\right\},\end{array}$$

The Continuous stream Model combines the flexibility of the soft real—time model with the simplicity of temporal analysis of time—triggered with job cancellation.

What about Control?

Problem Description

- We consider the problem of controlling a Linear and Time Invariant system

$$\dot{x} = \overline{A}x + \overline{B}u$$
$$y = \overline{C}x$$

- If we discretise the system using a sampling interval equal to the difference between two adjacent interaction points (chosen equal to the reservation period P)

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

Problem Description (cont.)

- We assume the application of a controller designed with a nominal period T = N R

$$\begin{aligned} z_{j+1} &= A_c z_j + B_c y_j \\ u_j &= C_c z_j + H_c y_j , \end{aligned}$$

- The controller is designed assuming a delay of N reservation periods
- Since we use the Continuous Stream MoC the delay will be a stochastic variable

$$D_j = \left\lceil \frac{c_j}{Q} \right\rceil R$$

ranging in the set

$$\mathcal{D} = \{N, \ldots, N_r\}$$

Problem Description (cont.)

- The delay can we included in the description of the plant coming up with:

$$x_{j+1} = A^{D_j} x_j + \sum_{t=0}^{D_j - 1} A^{D_j - t - 1} B u_{j-1}$$

$$y_j = C x_j .$$

- As a result the closed system switches stochastically between different dynamics:

$$\zeta_{j+1} = A_{\phi(j)} \zeta_j$$

$$y_j = \tilde{C} \zeta_j ,$$

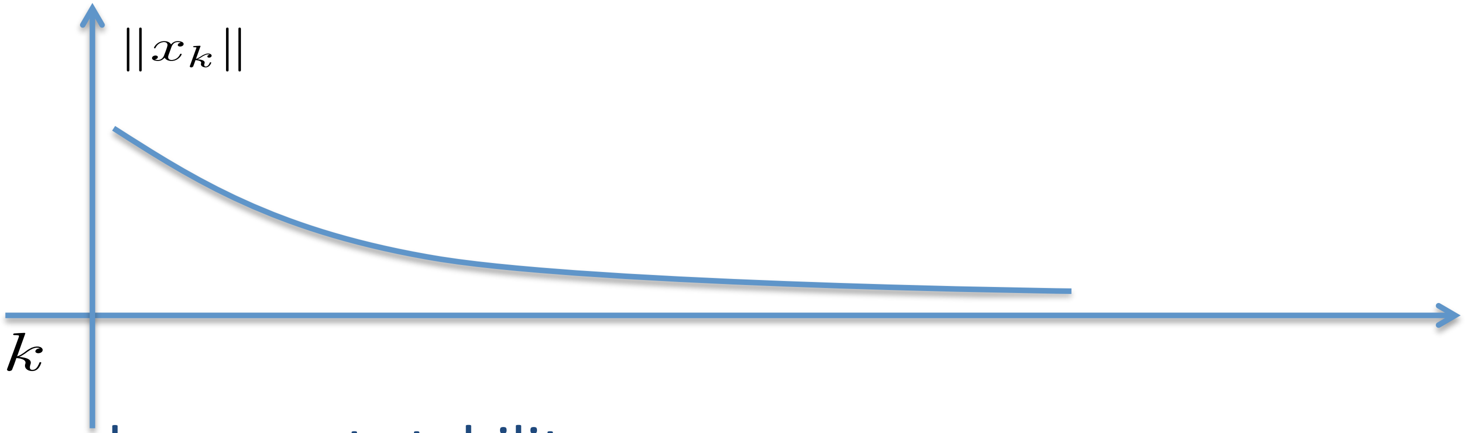
$$\text{with } \zeta_j = [x_j^T, z_j^T, \xi_j^T]^T$$

Observation

- The function $\phi(j)$ ruling the switches is tied to the evolution of the delay D_j and is:
  - A stochastic Markov Process for soft real—time time—triggered MoC
  - A bivariate random variable for time triggered with job cancellation
  - A multivariate random variable for Continuous Stream taking values in a set that contains the two values for time—triggered with job cancellation

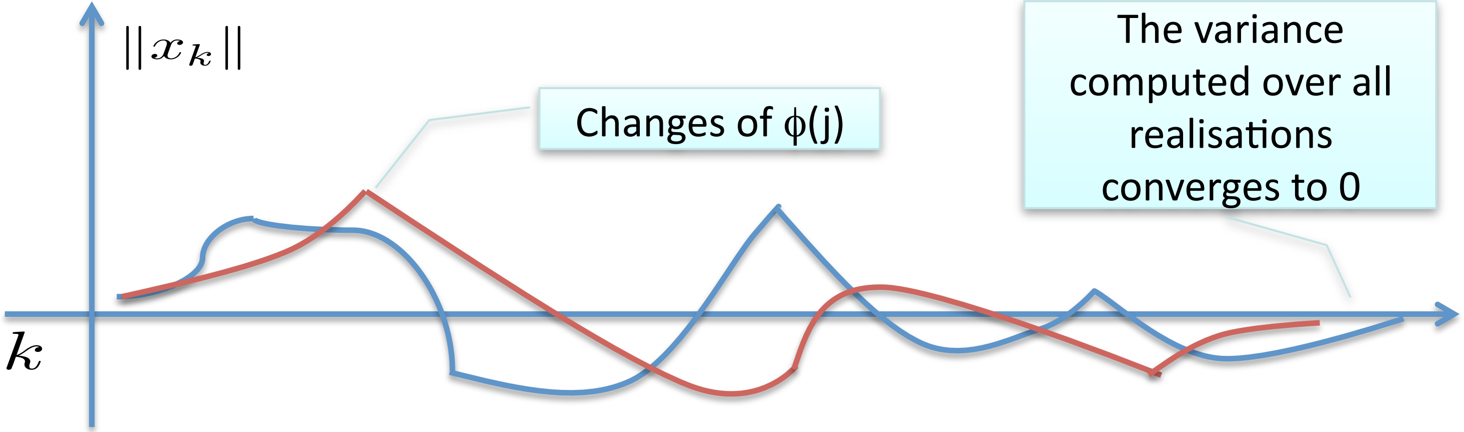
A notion of Stability

- Asymptotic stability



- Second moment stability

$$\lim_{k\to\infty} E(\|x_k\|^2) = 0$$



Changes of ϕ(j)

The variance computed over all realisations converges to 0

A Theoretical Result

- The system is second moment stable iff

$$\rho(\Gamma_A) = \max_k |\lambda_k(\Gamma_A)| \leq 1$$

All eigenvalue have modulus smaller than 1

where

$$\Gamma_A = \mu_0 A_i^{[2]} + \sum_{i=N}^{N_r} \mu_i A_i^{[2]}$$

$$A_i^{[2]} \triangleq A_i \otimes A_i$$

Computation of the probabilities

- Given the absence of memory, the probabilities are easier computed as a function of the budget:

$$\mu_N(Q^s) = \mathsf{Pr}\left(c_j \leq Q^s N\right) = \int_{-\infty}^{Q^s N} f(x)dx,$$

$$\mu_i(Q^s) = \mathsf{Pr}\left(Q^s(i-1) < c_j \leq Q^s i\right) = \int_{Q^s(i-1)}^{Q^s i} f(x)dx,$$

$$\mu_o(Q^s) = \mathsf{Pr}\left(c_j > Q^s(N+N_r)\right) = \int_{Q^s(N_r)}^{+\infty} f(x)dx,$$

Computation of the minimum budget

- The computation of the optimal budget is conceptually simple:

```
for $Q_s = 1 \dots T_s$ do
    $[\mu_0, \dots, \mu_i, \dots, \mu_N] =$ compute_probability($Q_s$);
    if $\rho(\Gamma_A) \leq 1.0$ break;
end for;
return $Q_s$;
```

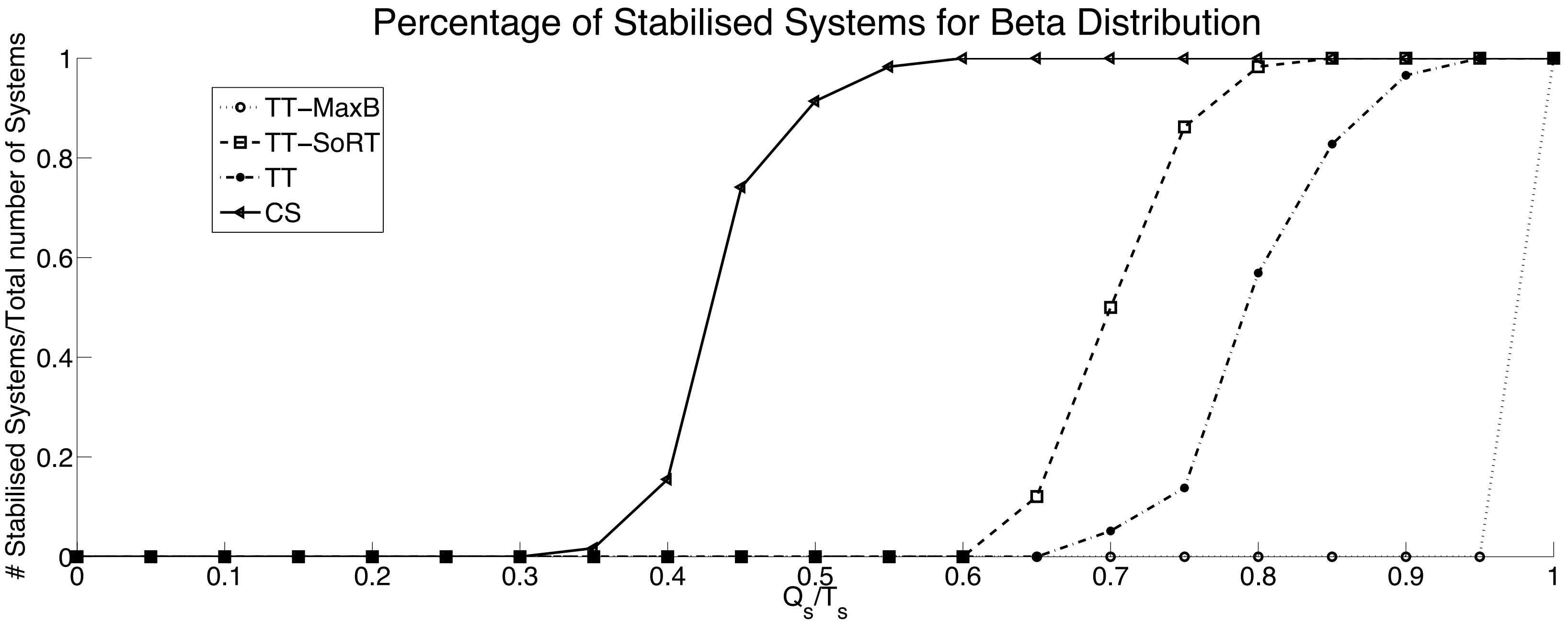
- Since we rely on a necessary and sufficient condition the result is tight

Simulation Results

Simulation Setup

- We considered 60 randomly generated systems, with order ranging form 2 to 5
- For each system we designed an LQG regulator, assuming:
  - a sampling time equal to 10ms
  - one sampling period delay
- We considered three different distributions for the computation time
  - Uniform
  - Exponential
  - Beta
- Me have measured the number of systems that we could stabilise for each choice of the bandwidth

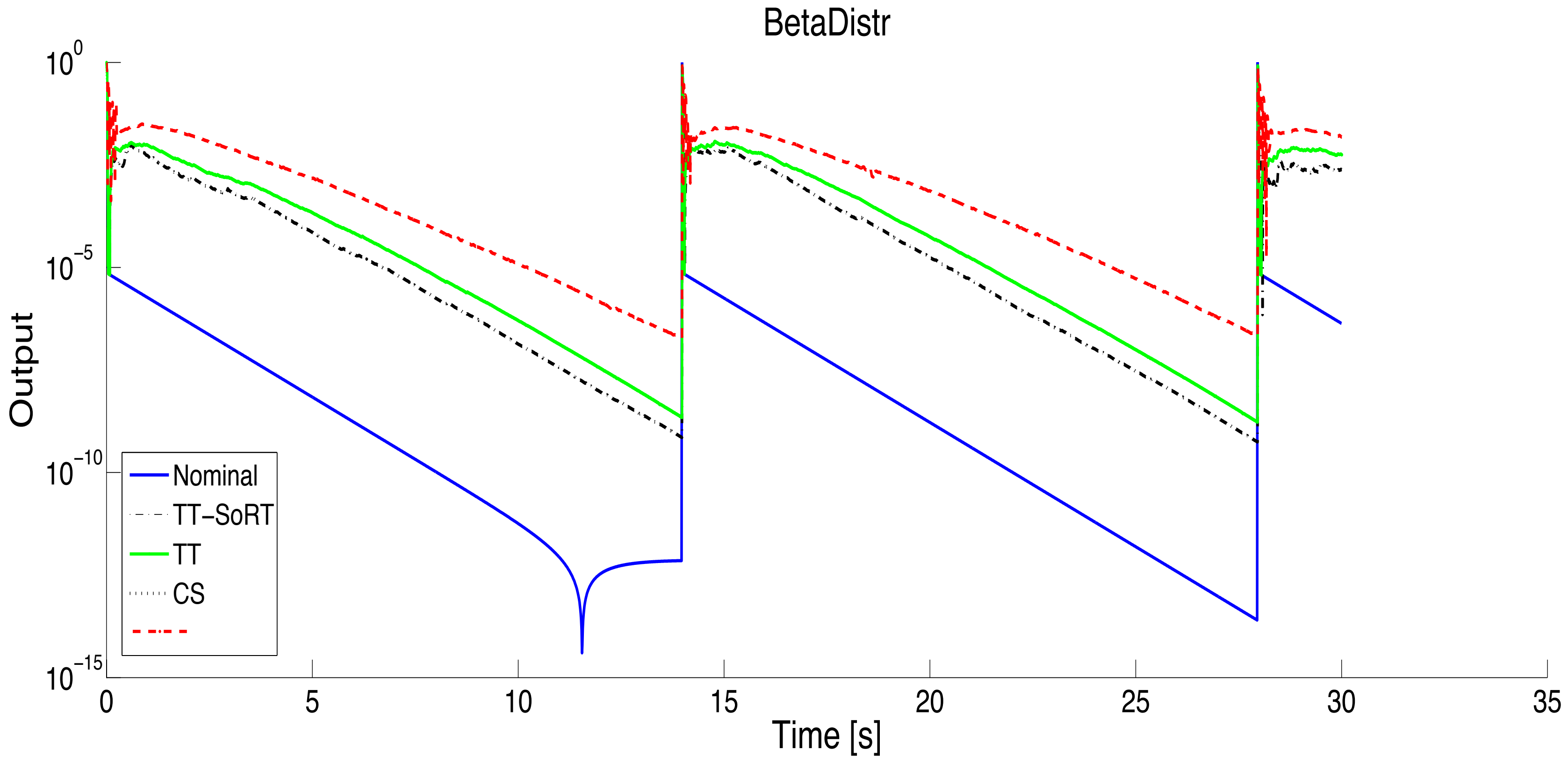
Results



- With 50% of the bandwidth used for hard rea—time we stabilise almost all systems
- The saving with respect to other approaches is substantial (40/30%) less
- But what about Quality of Control?

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria

A representative simulation

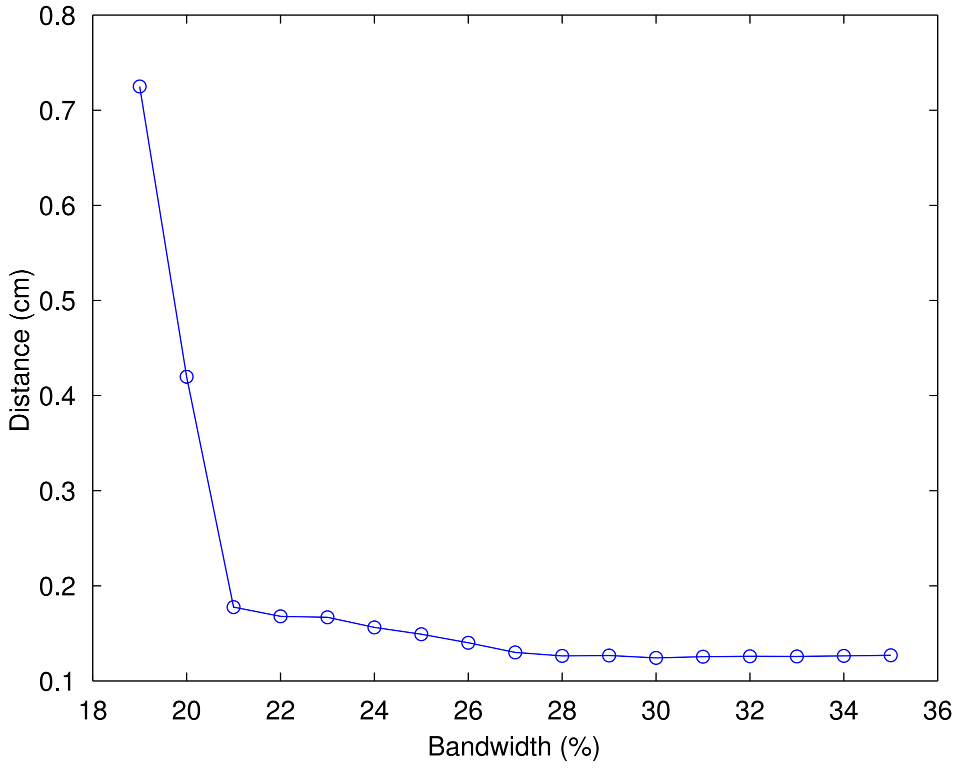


- The CS performance is a bit worse, but ….
  We need the logarithmic scale to show the the difference!!

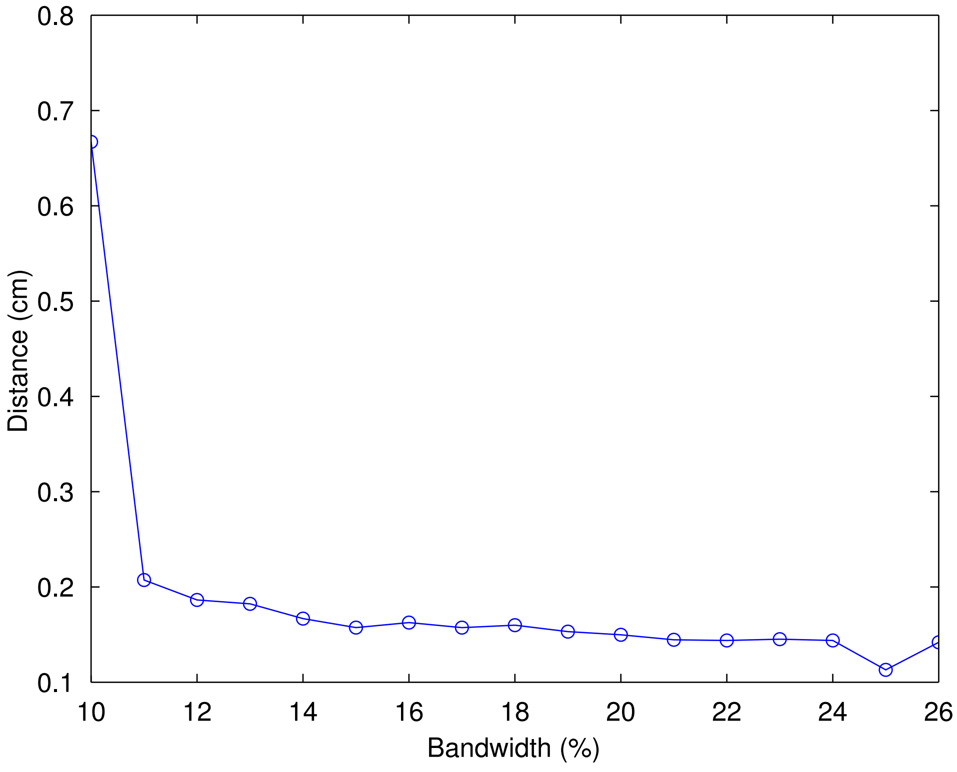
An experiment

- Resource Reservation scheduling is now available on the standard Linux Kernel.
- We could implement our ideas on a robotic case study
- The problem was following a line using visual sensors
- We tested and compared different solutions

Accuracy of the tracking



TT-SoRT



Continuous Stream

Conclusions

Conclusions

- The idea that we need hard real—time scheduling to implement feedback control systems is dead and buried
- Soft real—time scheduling brings important advantages
- The use of RR scheduling provides us with analysable models linking QoS with the scheduling parameters.
- The adoption of adequate models of computation allows us to translate QoS into QoC and therefore optimise the choice of scheduling parameters for a set of applications sharing a processor

Conclusions

- The advantages become substantial is we give up the idea of using absolute time and adopt the CS model of computation proposed in this paper
- This convenience has been shown by both theoretical results and experimental evidence
- Future investigation area
    - Are there control applications for which absolute time could still be needed/useful?
    - What about distributed/multicore?

A Theoretical Result

- Define the covariance matrix as

$$P_i = E\left\{x_i x_i^T\right\}$$

- We can stack up the elements of this matrix and write its dynamics

$$\mathrm{vec}(P_{j+1}) = \Gamma_A \mathrm{vec}(P_j)$$

with

$$\Gamma_A = \mu_0 A_i^{[2]} + \sum_{i=N}^{N_r} \mu_i A_i^{[2]}$$

$$A_i^{[2]} \triangleq A_i \otimes A_i$$

Probability associated to the closed loop dynamics

Closed loop Dynamics