INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

Beyond the m-k model: restoring performance considerations in the time abstraction

Marco Di Natale
Scuola Superiore S. Anna – Pisa, Italy

Based on work with Y. Sun, P. Pazzaglia, L. Pannocchi

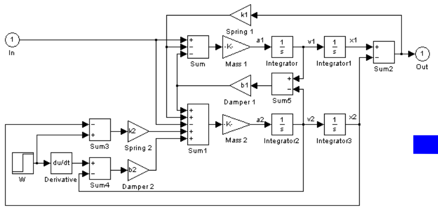
EMSOFT Tutorial - Seoul
October, 2017

Summarizing…

- It is about the timing abstractions…
- But touches on a lot of issues
  - What are the available abstractions?
  - To what degree are timing abstractions possible/desirable at all?
  - What is the real consequence for the application of missing a deadline (depends on the program structure) ?
  - How do you manage a missed deadline (continue, drop…) ?
  - Availability vs. Dependability – should we drop the pretense of being fully predictable at design time?

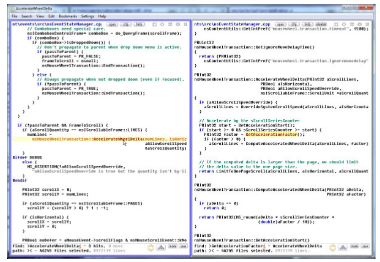
Typical assumption for separations of concerns

Controls Domain



Task (thread) abstraction

SW Domain



Functional Design

<u>Contract</u> is often very simplistic

Platform Design

Performance evaluation

Example, periodic tasks with hard deadlines

$\tau_i = (T_i, C_i, D_i)$

Satisfy the task contract (schedulability)

But you often can miss deadlines

- There is a very small number of “true” hard real-time systems
- Need for models that account for overload in several control domains (where deadline can be missed)
- Task models: from Hard to Firm to Soft deadline model to Value-based scheduling
  - Still not very satisfactory (declining trend in conference proceed.)
- The timing analysis community likes to build its own models (possibli inspired but drifting away from requirements)
  - Informally, we can miss deadlines but not too many and not too many in a row
  - (we don’t get too much help from industry)
  - This is where the m-k model probably originated

The m-k model

- By now you heard plenty about it
- You can miss at most m deadlines out of k instances
- Not the only possible abstraction
  - Lenght and number of deadline misses in longest busy period
  - Other options …

The m-k model: Our solution

A summary of our work on m-k analysis

To be presented here (EMSOFT) on Tuesday

Y. Sun, M. Di Natale “Weakly Hard Schedulability Analysis for Fixed Priority Scheduling of Periodic Real-Time Tasks”

Contributions wrt previous work

- Is not restricted to offset determined systems
- Can sweep a range of m-k options and even find the minimum m for a given k
  - Easily done since it is based on an optimization formulation

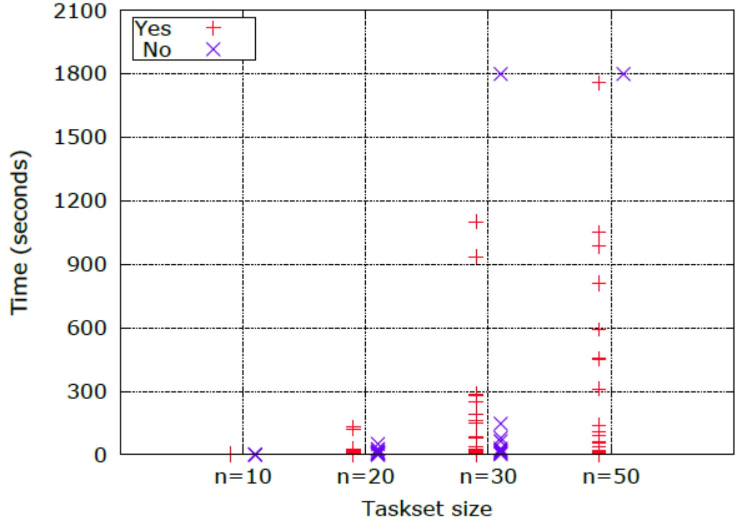
Limitations

- Still limited to periodic load
- Does not scale beyond 20 tasks and k>10
  - Do we really need those?

The m-k model: Our solution

- How? Formulate the problem as a MILP
- Relaxing some constraints (number of interferences) and then refining to limit pessimism
- Feasibility or optimization formulation
  - ***Maximize # of misses m in any given window of k***
- Results *very* close to true optimum
- Runtimes acceptable for many configurations

m=2, k=5



But you often can miss deadlines

- Problem with m-k model
- It is still a binary assessment (black and white)
  - No deadline miss→correct
  - Deadline miss →critical failure
  - Does not account for performance
- It is stateless (the position of the deadline misses in the sequence does not count)
  is MMMHHH same as MHMHMH ???? (unlikely)

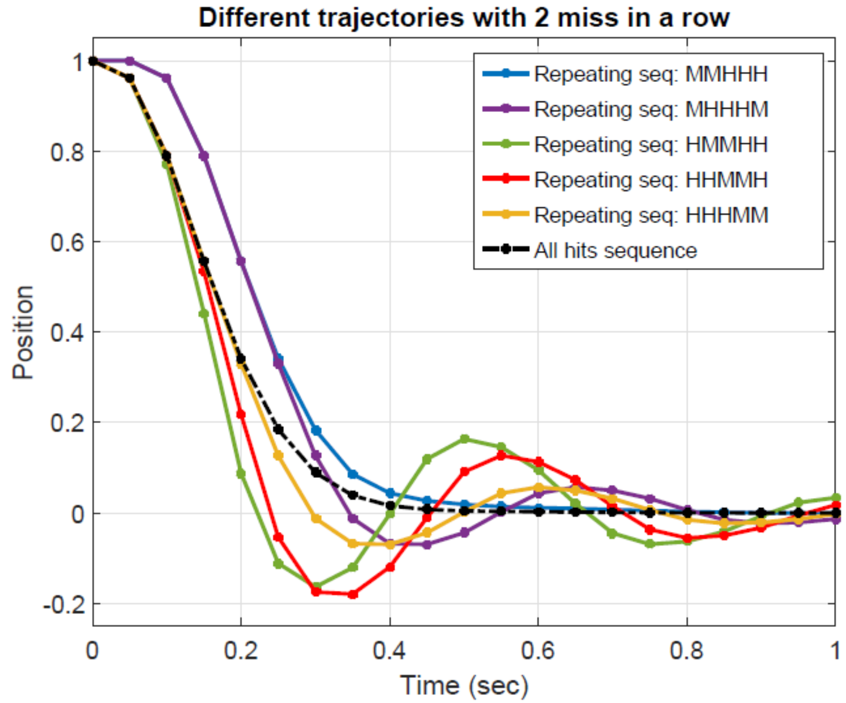


Fig. 1. Comparing different trajectories with the same constraint of deadline missed in a row. In general, changing the order of the missed deadlines in a H/M sequence leads to different behaviours, that cannot be discerned by the (h, n) description.

But you often can miss deadlines

- We want to restore performance in the task contract
- How can you do that?
- It turns out it is not simple at all ….
  - And not because of a lack of fantasy in creating the contract…
- Depends on many assumptions and design choices

Also, consider the following observation

(M. Neukirchner @Waters workshop)

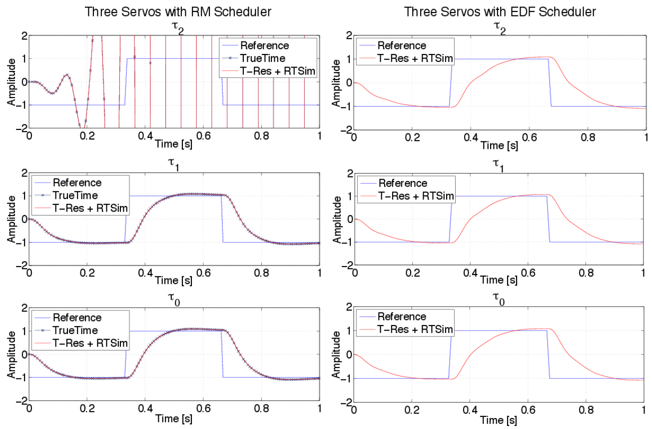
- “The real time community is very concerned about availability but should be more concerned about reliability”
- Support synthesis of monitors

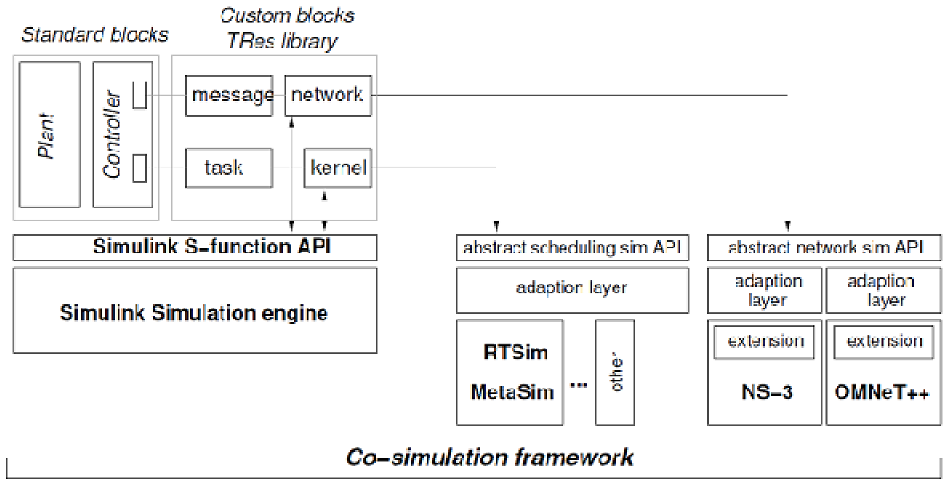
Performance consequences of missed deadlines

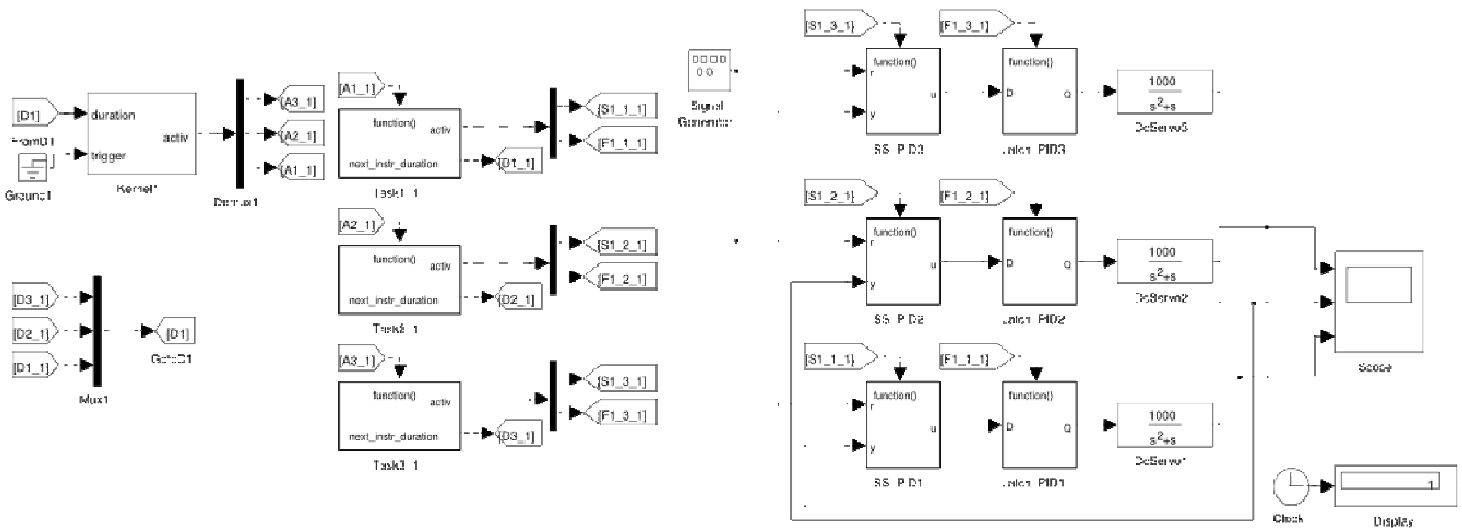
- Other alternative: avoid abstractions and perform the joint analysis (by model checking or (co)simulation) of the task and controls model
- By simulation – Truetime, TRes (Simulink based), simulate the scheduler and the tasks together with the controls logic
- By formal models – Hybrid systems, SpaceX

The T-Res approach

Co-simulation in Simulink







What is the effect of a missed deadline?

The effect of a deadline miss heavily depends on the SW architecture

- It may be a delayed output (the task directly outputs)
- It may be an output at the usual time with old data (the task fails to update a TPU programming or misses a cycle in asynchronous cmmunication)
- It may be data that gets overwritten and completely missed

How do you manage a missed deadline?

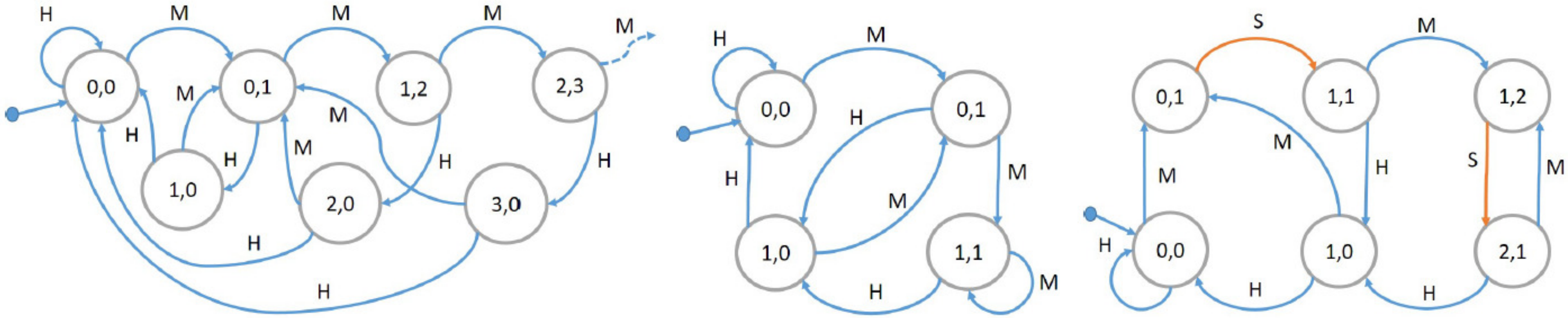
- When a task misses a deadline
  - Should you let it finish?

    Accept the late termination, hope it is temporary
  - Should you terminate it at the deadline?

    spare load, give better chance to next instance to complete in time
  - Should you let it finish but skip the next instance?

    try to recover from overload
  - Should you terminate and skip?

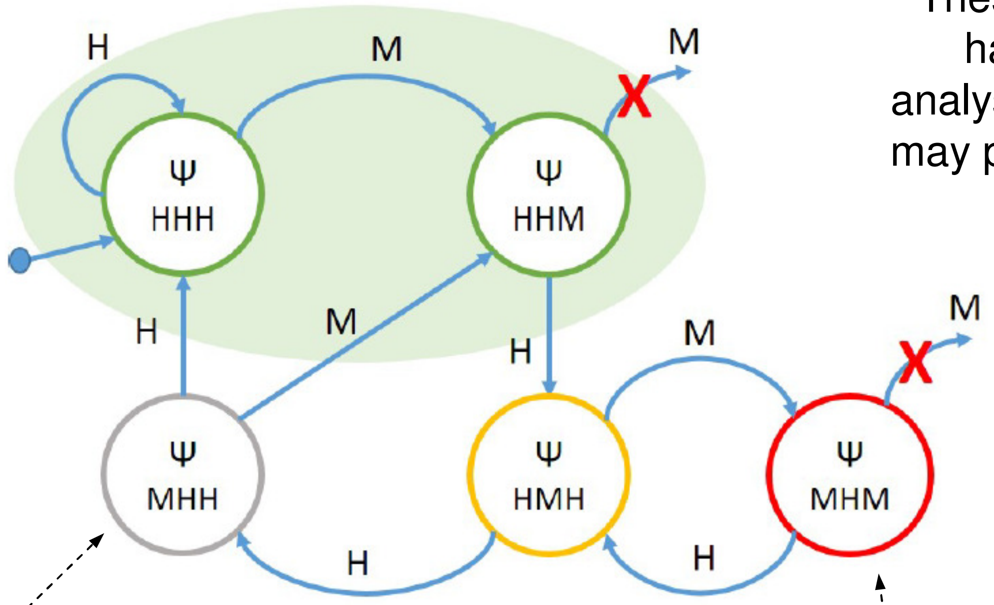
Effect of deadline misses



(a) Kill job (b) Continue job (c) Skip next job

- ***Hypothesis***: a Deadline miss results in use of old data by one cycle
- The delay of the output value depends on the management miss policy
- If you bound the possible sequences of Hits/Misses (by standard m-k analysis) you get a finite number of possible states for the data delays
- Each state can now be annotated with the corresponding performance

States as representatives of performance regions



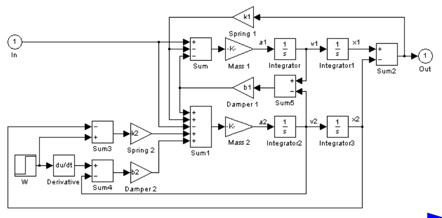
These should never happen by m-k analysis (but a monitor may perform a runtime check)

Each state is annotated by performance

When the performance is critical a monitor might activate a recovery action

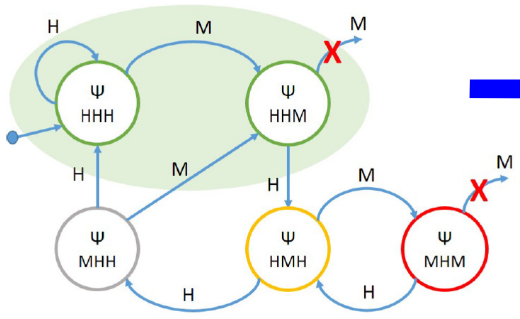
Typical assumption for separations of concerns

Controls Domain



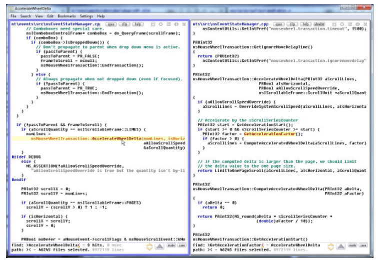
Functional Design

Performance evaluation



Possible Contract

SW Domain



Platform Design

Satisfy the task contract (schedulability)

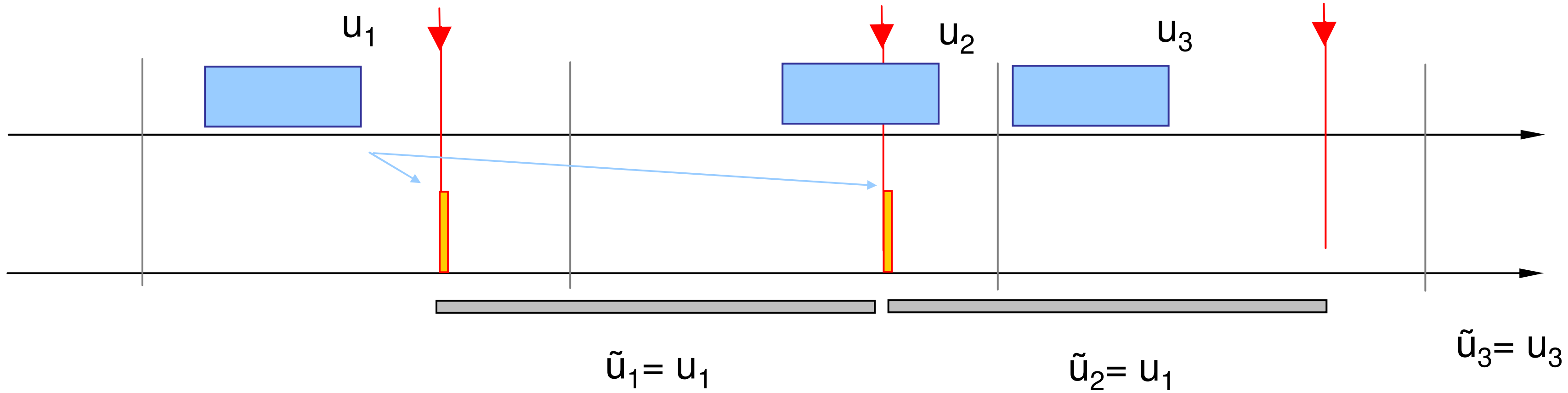
How do you obtain the performance model?

- How is this state machine computed?
- Analytically
  - For (simple?) LTI models
  - With a number of assumptions on sampling, actuation, deadline management …
- Experimentally/by Simulation
  - Truetime, TRes …

Simple LTI system: Example with assumptions

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t),$$
$$y_c(t) = C_c x_c(t),$$

- Controller implemented by τ_i period T_i, deadline D_i with $D_i \leq T_i$
- The job activated at time kT_i uses the state sensed at activation time kT_i (no sensing jitter). The actuator uses the output computed at the deadline ($kT_i + D_i$), and keeps it constant until (($k + 1)T_i + D_i$).



Simple LTI: Assumption

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t),$$
$$y_c(t) = C_c x_c(t),$$

- The system is discretized as

$$x[k+1] = A_d x[k] + B_{d1} u[k-1] + B_{d2} u[k],$$
$$y[k] = C_d x[k]$$

- With Bd1 and Bd2 expressed by

$$A_d = e^{A_c T_i},\ C_d = C_c$$
$$B_{d1} = \int_{T_i - D_i}^{T_i} e^{A_c s} ds\, B_c,\ B_{d2} = \int_0^{T_i - D_i} e^{A_c s} ds\, B_c$$

Simple LTI: Assumption

If no deadline miss

$$\tilde{u}_k = \begin{cases} u[k-1] & \text{for } t \in [kT_i,\, kT_i + D_i) \\ u[k] & \text{for } t \in [kT_i + D_i,\, (k+1)T_i) \end{cases}$$

If deadlines are missed

$$\tilde{u}_k = \begin{cases} u[k-\Delta_p-1] & \text{for } t \in [kT_i,\, kT_i + D_i) \\ u[k-\Delta_c] & \text{for } t \in [kT_i + D_i,\, (k+1)T_i) \end{cases}$$

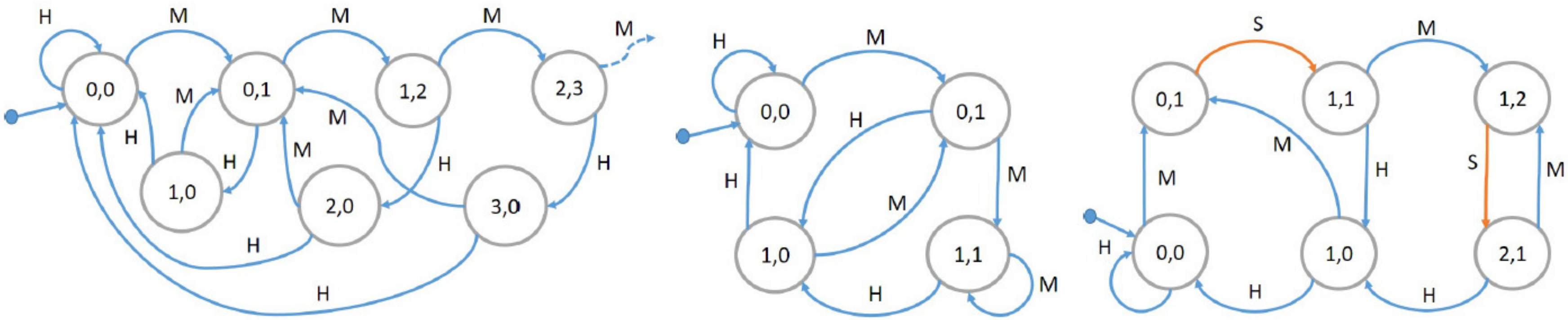
Δ_p (previous) is the update age of the control output in the time interval $[(k+1)T_i + D_i; kT_i + D_i)$

Δ_c (current) output age in the interval $[kT_i + D_i; (k + 1)T_i + D_i)$.

Need the pair Δ_p, Δ_c for all possible sequences of deadline hits and misses

Simple LTI: Assumption

Need the pair Δ_p, Δ_c for all possible sequences of deadline hits and misses



(a) Kill job (b) Continue job (c) Skip next job

Simple LTI: Assumption

Expressing u as function of Δ_p, Δ_c in the state equation

$$x[k+1] = A_d x[k] + B_{d1} u[k-1-\Delta_p] + B_{d2} u[k-\Delta_c]$$

For a simple state feedback control

$$u[k] = K(r[k] - x[k])$$

If the reference *r[k]* is the null vector

$$x[k+1] = A_d x[k] - B_{d1} K x[k-1-\Delta_p] - B_{d2} K x[k-\Delta_c]$$

Simple LTI: Evolution at each hit/miss

If $\mathbf{x}[k]$ is an extended state vector representing the state at the past Δ_{max} +1 steps

$$\mathbf{x}[k] = \left[x[k], x[k-1], \cdots, x[k-\Delta_{max}-1]\right]^T$$

We have

$$\mathbf{x}[k{+}1] = \begin{bmatrix} A_d & \cdots & -B_{d2}K & \cdots & -B_{d1}K & \cdots \\ \mathbf{I}_n & \mathbf{0}_n & \cdots & \cdots & \cdots & \cdots \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n & \cdots & \cdots & \cdots \\ \vdots & \cdots & \ddots & \ddots & \cdots & \cdots \end{bmatrix} \mathbf{x}[k]$$

The position of the terms B_{d1} and B_{d2} depends on Δ_c and Δ_p

$$\mathbf{x}[k{+}1] = \mathbf{\Phi}(\Delta_p, \Delta_c, \Delta_{max})\ \mathbf{x}[k]$$

Simple LTI: The Φ matrix at each hit/miss step

- Φ depends on the deadline miss management
- If jobs that miss deadlines are allowed to continue, for example …

$$\mathbf{\Phi}(0,0,1) = \begin{bmatrix} A_d - B_{d2}K & -B_{d1}K & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

$$\mathbf{\Phi}(0,1,1) = \begin{bmatrix} A_d & -BK & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

$$\mathbf{\Phi}(1,0,1) = \begin{bmatrix} A_d - B_{d2}K & \mathbf{0}_n & -B_{d1}K \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

$$\mathbf{\Phi}(1,1,1) = \begin{bmatrix} A_d & -B_{d2}K & -B_{d1}K \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}$$

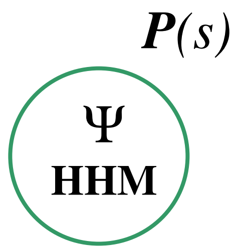
Simple LTI: Assumption on metric

For the cumulative quadratic error index metric (for example)

$$P_{cqe} = \sum_{i=0}^{N} x[i]^T x[i].$$

Assuming a given sequence of hits/misses

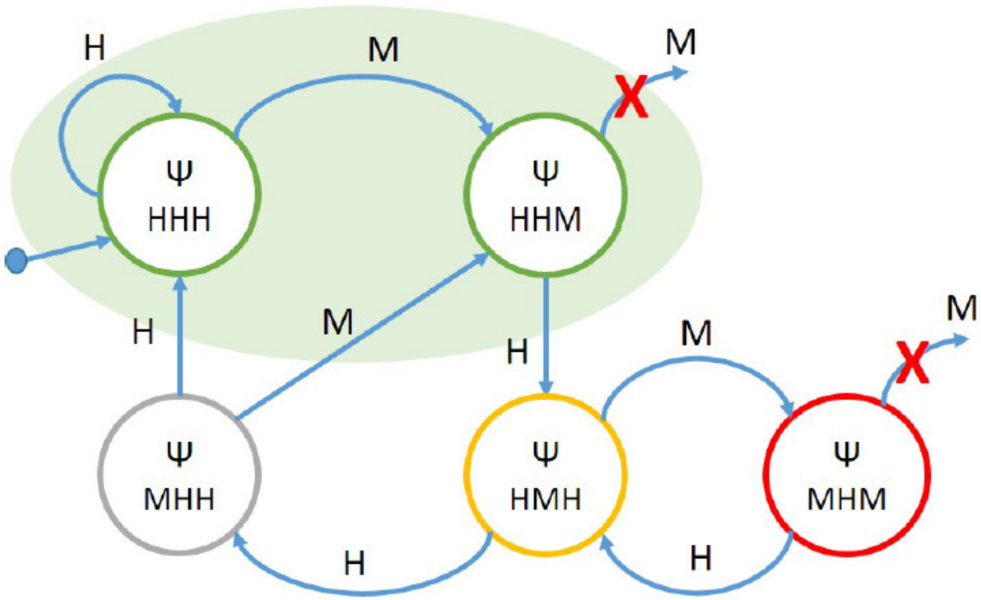
$$\begin{aligned}
\mathbf{P}_{cqe}(s) &= \sum_{i=0}^{N} \mathbf{x}[i]^T \mathbf{x}[i] \\
&= \mathbf{x}_0^T \Big(\mathbf{I} + \mathbf{\Phi}_1^T \mathbf{\Phi}_1 + \mathbf{\Phi}_1^T \mathbf{\Phi}_2^T \mathbf{\Phi}_2 \mathbf{\Phi}_1 + \dots \\
&+ \mathbf{\Phi}_1^T \mathbf{\Phi}_2^T \cdots \mathbf{\Phi}_{N-1}^T \mathbf{\Phi}_N^T \mathbf{\Phi}_N \mathbf{\Phi}_{N-1} \cdots \mathbf{\Phi}_2 \mathbf{\Phi}_1 \Big) \mathbf{x}_0 \\
&= \mathbf{x}_0^T \mathbf{\Psi}(s) \mathbf{x}_0
\end{aligned}$$



The matrix $\Psi(s)$ can be computed as a function of the matrices $\Phi(\Delta_p, \Delta_c, \Delta_{max})$ generated by the sequence

States as representatives of performance regions

- Hence the performance annotation for the state model …



Paper # 2016-01-0017

Conclusion

Thank you!