MPC Design Methods for Networked Systems

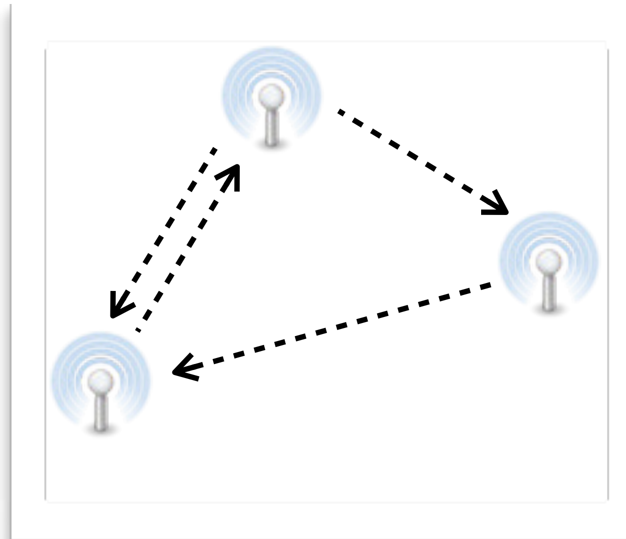
Alberto Bemporad

http://www.imtlucca.it/alberto.bemporad

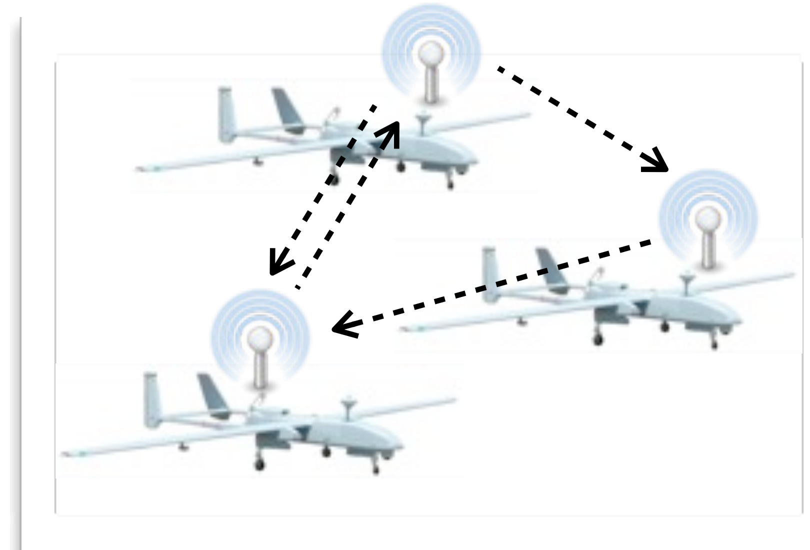
IMT Institute for
Advanced Studies Lucca

What is a networked system ?

+



=



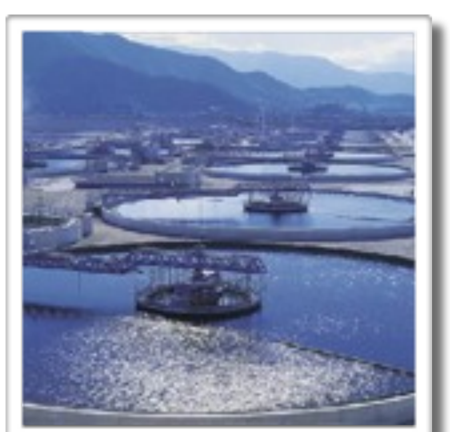
process dynamics

interactions (ex: communication infrastructure)

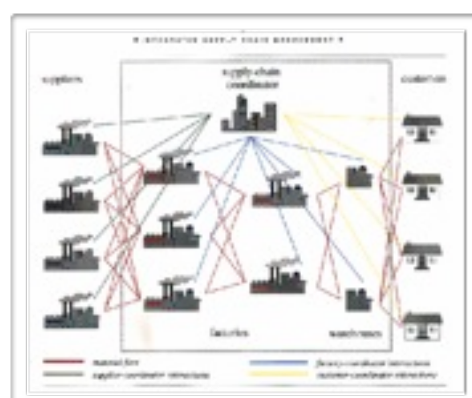
networked system

- Real-life examples of networked systems abound !

traffic networks

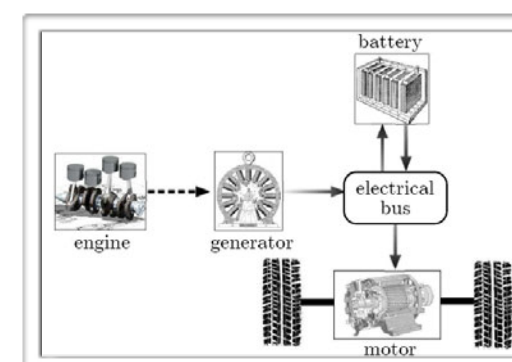


water networks



logistic networks (supply chains)

power networks (smart grids)

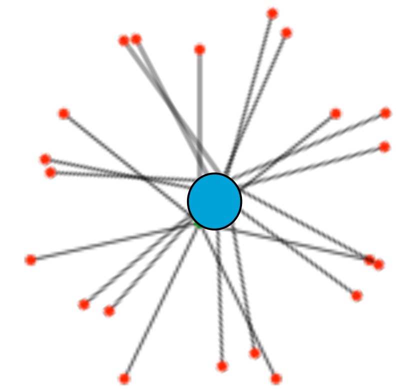


networked components of a vehicle

Main issues in controlling networked systems

- Dynamics are typically **spatially distributed** and **large-scale** (=many states)

- **Centralizing all measurements** can be difficult:
  - too many data to handle
  - too many transmissions
  - confidentiality issues



- **Centralizing all computations** can be difficult:
  - computations too complex (scalability issues)
  - big models difficult to setup and maintain

- **Wireless communication** is very flexible, but introduces further issues:

  - random packet dropouts
  - time-varying delays & jitter
  - battery energy consumption

Need for control design methods that can handle these issues !

Outline of this talk

- Quick review of MPC basics

- MPC that limits the use of network resources

→ **Energy-aware MPC**

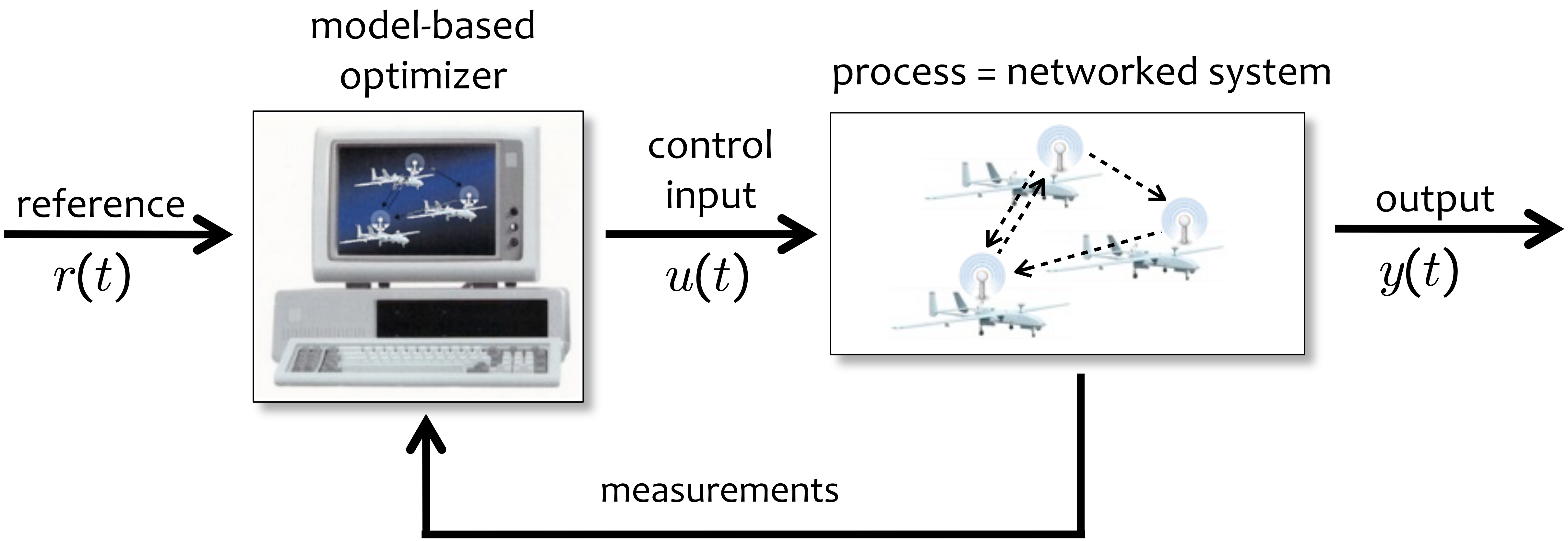
- MPC that tames model complexity

→ **Decentralized MPC**

- MPC that takes into account communication imperfections

→ **Stochastic MPC**

Model Predictive Control (MPC)



*Use a dynamical **model** of the process to **predict** its future evolution and choose the "best" **control** action*

- Prediction model = dynamics + network
- Models can be *deterministic* or *stochastic*, *global* or *local*

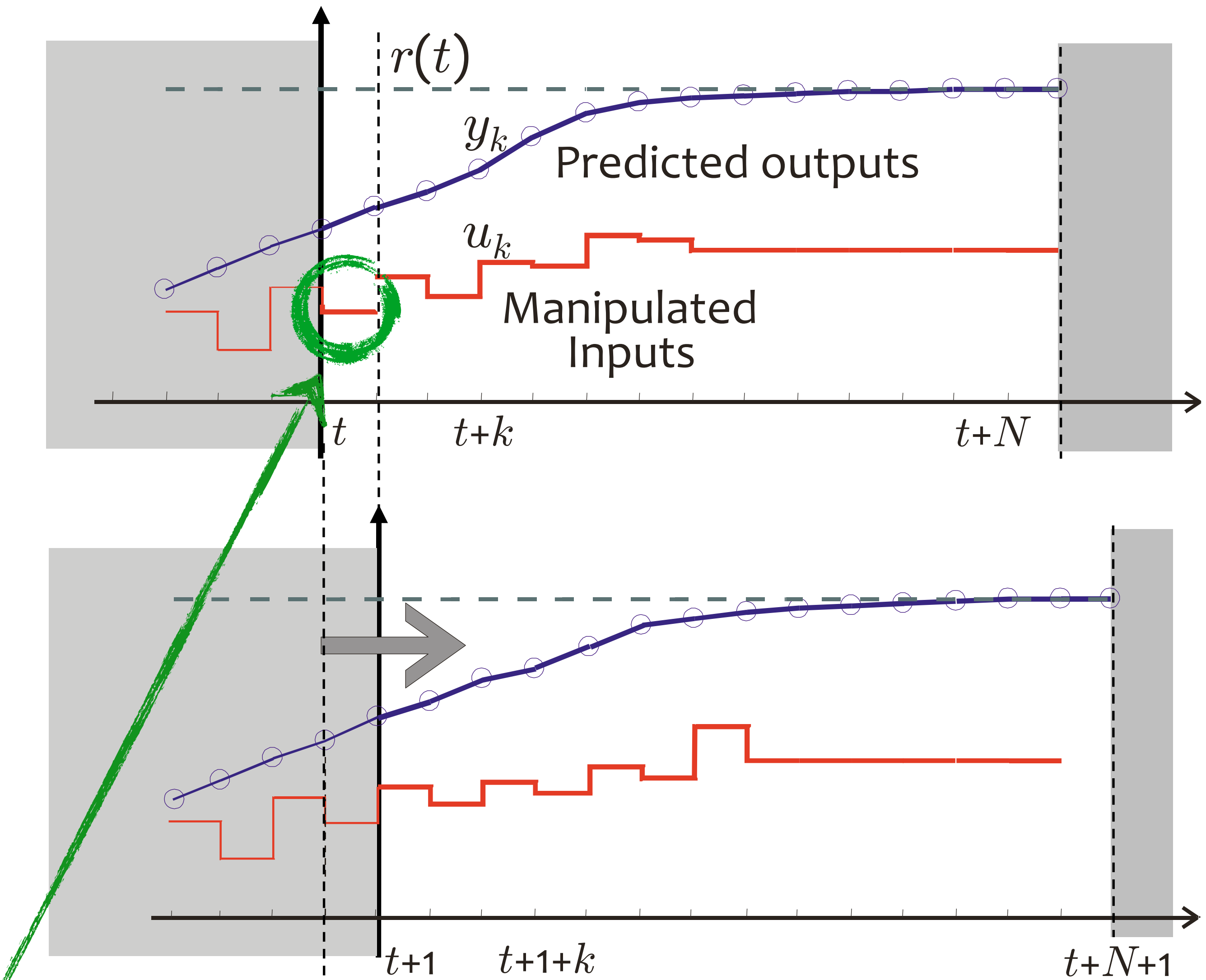
MPC algorithm

- **At time** t: solve an **optimal control** problem over a future horizon of N steps

penalty on tracking error

penalty on actuation effort

$$
\begin{array}{ll}
\min & \sum_{k=0}^{N-1} \ell(y_k - r(t+k), u_k) \\
\text{s.t.} & x_{k+1} = f(x_k, u_k) \\
& y_k = g(x_k, u_k) \\
& \text{constraints on } u_k, x_k, y_k \\
& x_0 = x(t)
\end{array}
$$



- Apply only the first optimal move $u^*(t)$, throw the rest of the sequence away
- **At time** $t+1$: Get new measurements, repeat the optimization. And so on ...

MPC transforms open-loop optimal control into feedback control

MPC of linear systems

linear model
$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{cases}$$

$$R = R' \succ 0 \qquad Q = Q' \succeq 0 \qquad P = P' \succeq 0$$

performance index
$$\min_U x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k$$

$$x_0 = x(t)$$

$$\begin{array}{ll} \displaystyle\min_U & \frac{1}{2}U'HU + x'(t)F'U + \cancel{\frac{1}{2}x'(t)Yx(t)} \\ \text{s.t.} & GU \leq W + Sx(t) \end{array}$$

$$U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}$$

constraints
$$\begin{cases} u_{\min} \leq u_k \leq u_{\max} \\ y_{\min} \leq Cx_k \leq y_{\max} \end{cases}$$

MPC implemented by solving a **(convex) Quadratic Program (QP)**

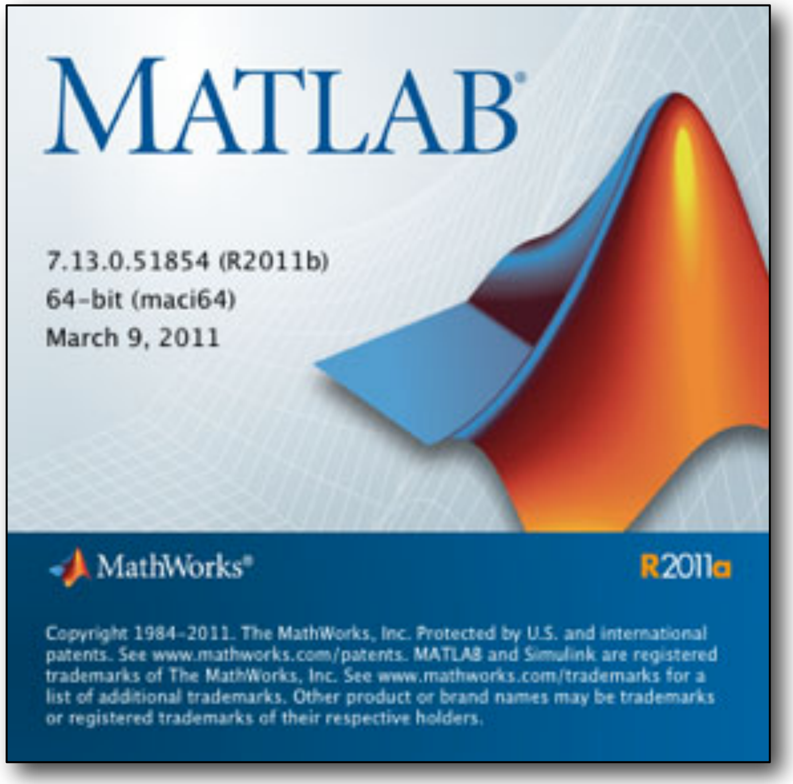
Routinely used in the process industries

Honeywell aspentech ABB

Model Predictive Control Toolbox

(Bemporad, Ricker, Morari, 1998-2011)

- **MPC Toolbox 4.0** (The Mathworks, Inc.)
  - New QP solver (active set method)
  - New MPC Simulink Library (based on EML code)
  - MPC Graphical User Interface
  - Code generation [RTW, xPC Target, dSpace, etc.]
  - Linked to OPC Toolbox v2.0.1, SYS-ID Toolbox



Easy-to-use solution for linear MPC design based on QP

http://www.mathworks.com/products/mpc/

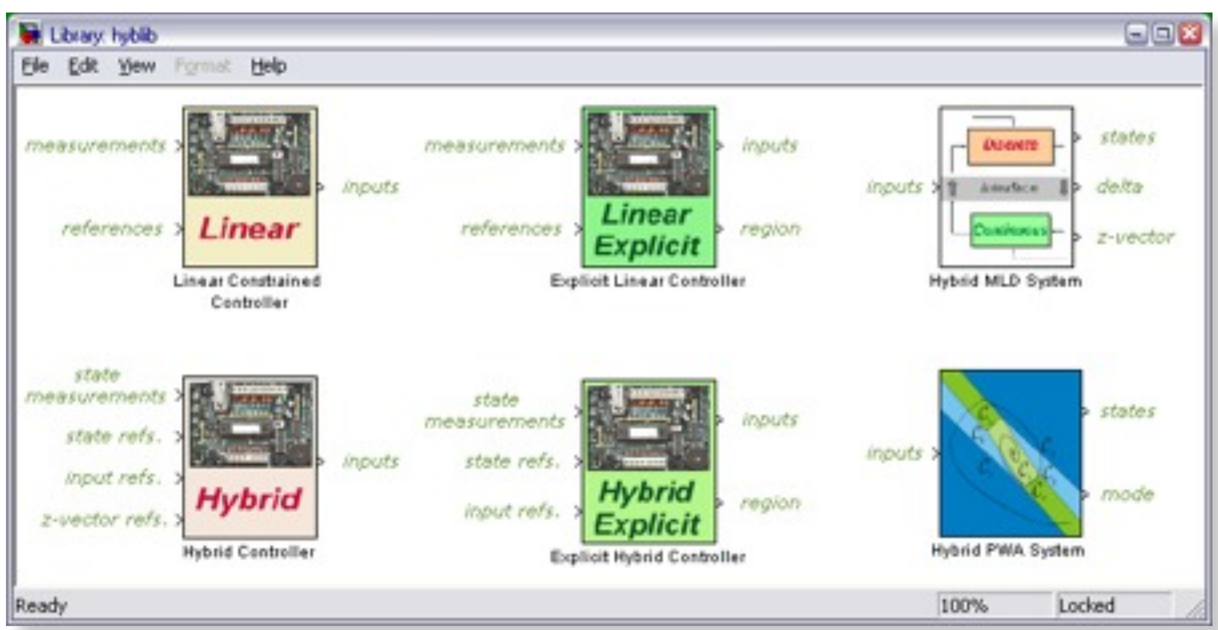
Hybrid Toolbox for MATLAB

(Bemporad, 2003-2011)

Features:

- **Hybrid** models: design, simulation, verification
- Control design for linear systems w/ constraints and hybrid systems (on-line optimization via QP/MILP/MIQP)
- **Explicit** MPC control (via multi-parametric programming)
- C-code generation
- Simulink library

Supported by



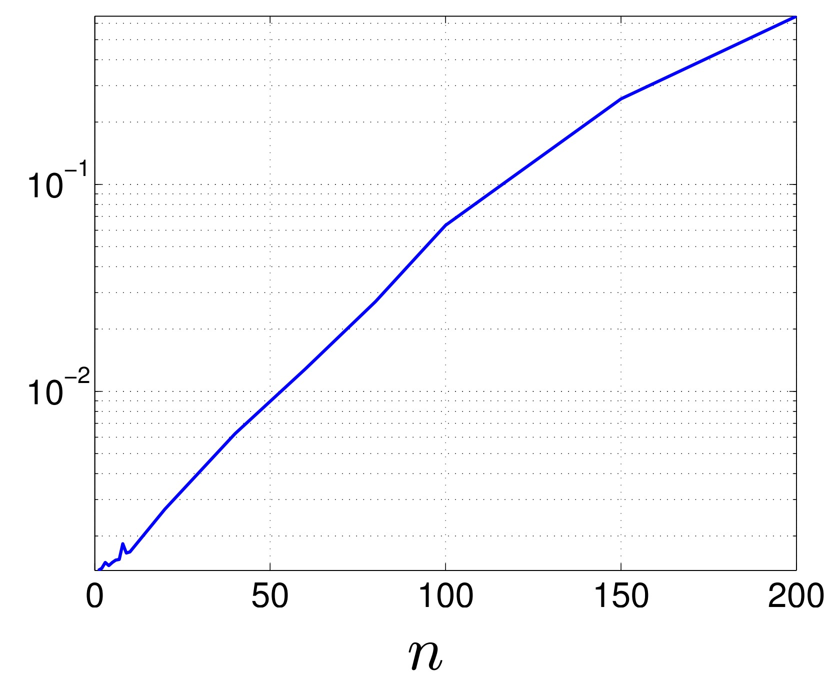
3500+ download requests since October 2004

http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/

Numerical complexity of MPC - An example

- Linear MPC of random square MIMO systems
  - n outputs, n inputs, $3n$ states
  - prediction horizon N=10, control horizon m=2
  - constraints: $-1 \leq u_k \leq 1, \ \ -1 \leq y_k \leq 1$
  - QP size: $(mn+1)$ variables, $(2Nn+2mn)$ constraints

CPU time per QP (s)



n	#vars	# constraints	CPU time (s)
1	3	24	0.00136
5	11	120	0.00149
20	41	480	0.00270
100	201	2400	0.06432
150	301	3600	0.25873
200	401	4800	0.64981

Macbook Air 2.13 GHz (this macbook !)

Inter Core 2 Duo 4GB RAM

MPC Toolbox 4.0, MATLAB R2011b

New active set QP in EML (dense matrices)

System with **200** inputs and **200** outputs w/ constraints: **less than 1 s** !

The example shows that **CPU time** is not a problem in many applications.
Getting the **model** (off-line) and all **measurements** (on-line) is the largest effort !

Outline of this talk

✓ Quick review of MPC basics

- **MPC that limits the use of network resources**

→ **Energy-aware MPC**

- MPC that tames model complexity

- MPC that takes into account communication imperfections

Energy-aware control in wireless networks

- Control over **wireless networks**:
  - **Pros**: flexibility, low cost
  - **Cons**: wireless nodes are **battery operated** and have short lifetime

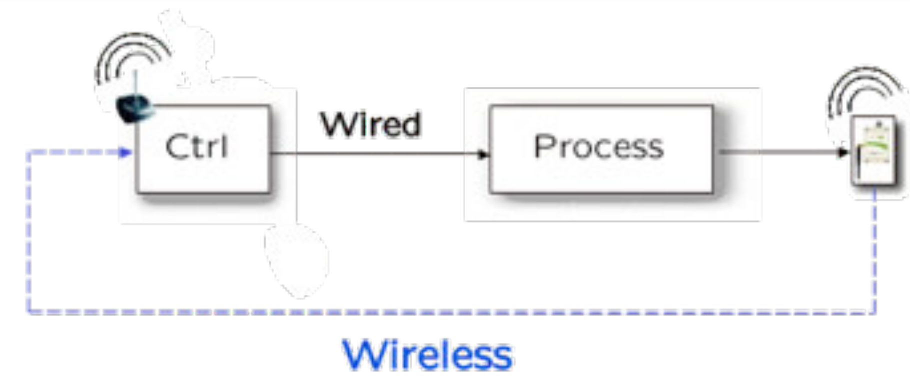
- How to save battery capacity:
  - Limit **radio activity** (idle listening and transmission), radio chip is the one consuming the most power

- Recent work to minimize **energy consumption**:
  - Consumption-efficient routing protocols (C. E. Jones et al., 2004)
  - Dynamic power management techniques (V. Raghunathan et al., 2005)

Goal: consider **energy consumption** problem in MPC control design

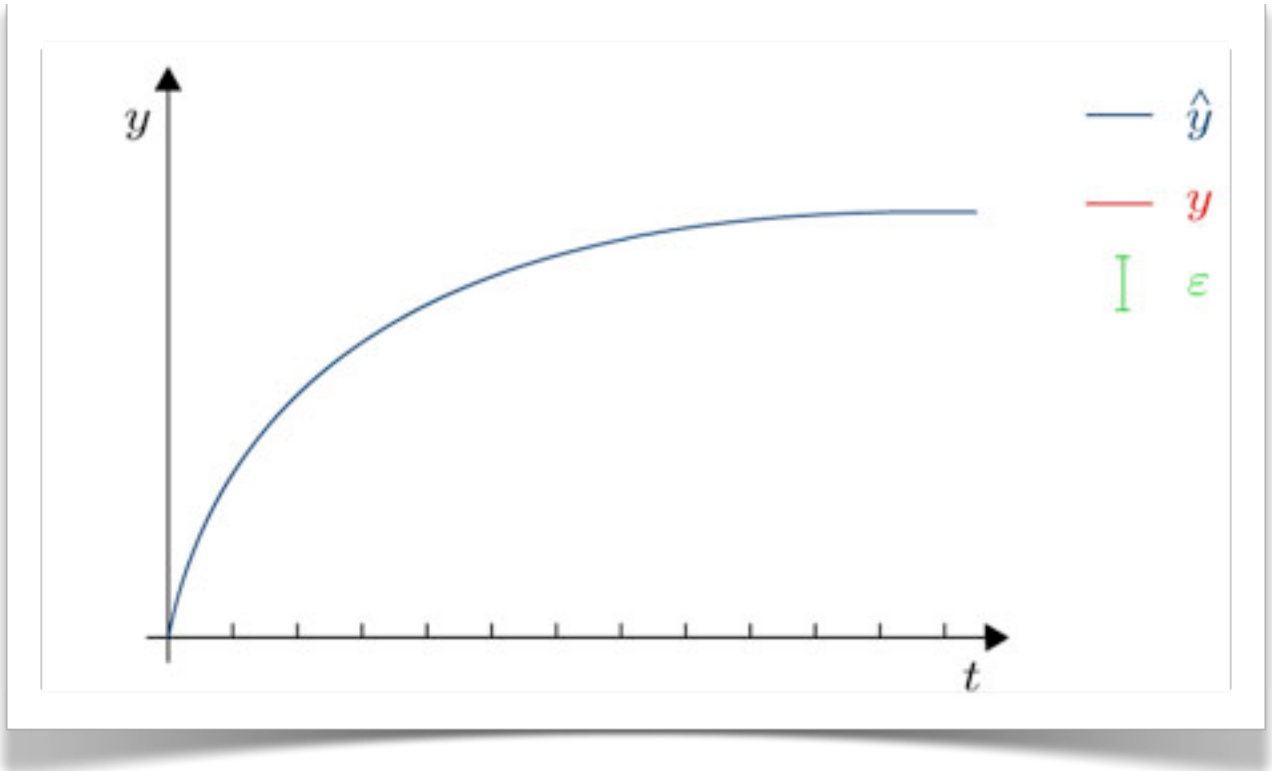
Energy-aware control - Key idea

- Control strategy should keep the radio off ***(both Tx and Rx)*** as much as possible!



- Trade-off: closed-loop performance vs. transmission rate

- **Key idea:** provide a prediction $\hat{y}(k)$ in advance to the sensor of the output $y(k)$ it measures.
  The sensor only transmits when measurement and prediction differ enough:



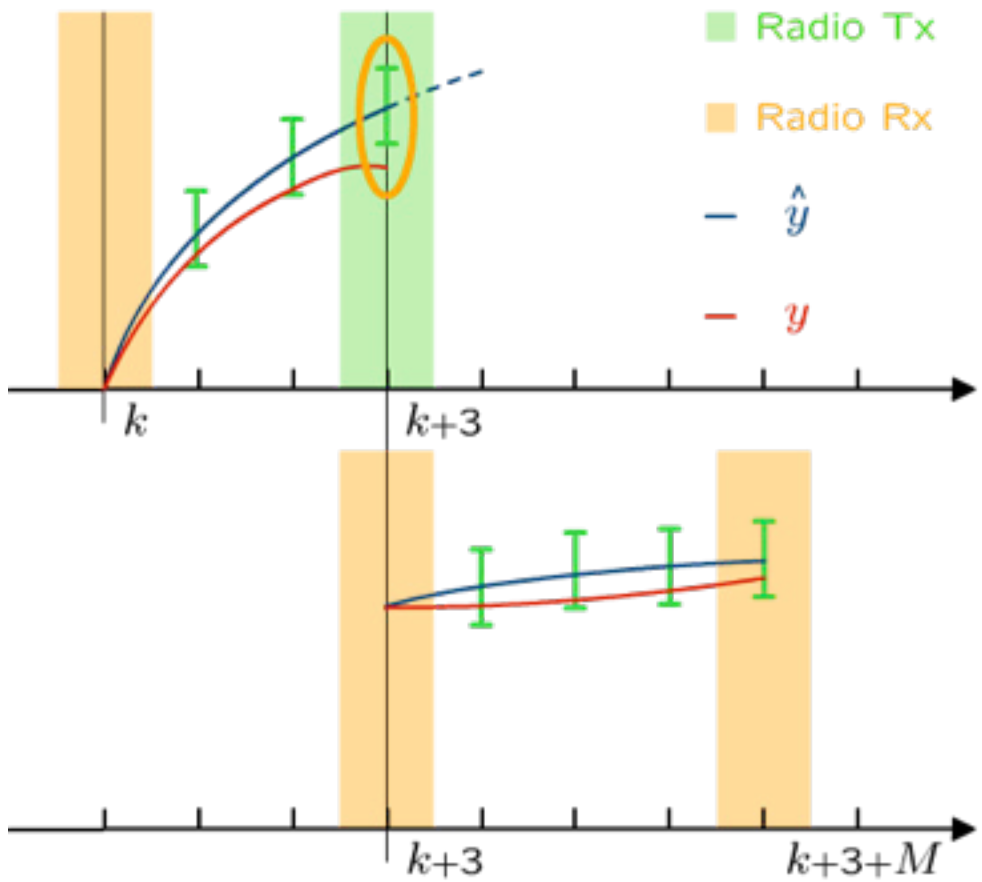
$y(k)$ is transmitted $\Leftrightarrow |y(k) - \hat{y}(k)| > \varepsilon$

- **Working assumption:** no packet loss

Energy-aware control - Key idea

- When the controller receives the measurement, it computes and transmits a new set of M predictions

- In the ideal case of no disturbances, i.e. $\hat{y}(k) = y(k), \forall k,$ the overall transmission rate is $1/M$



- Note: wireless communication protocols require a **minimum frame size** (e.g. 248 bits in ZigBee)

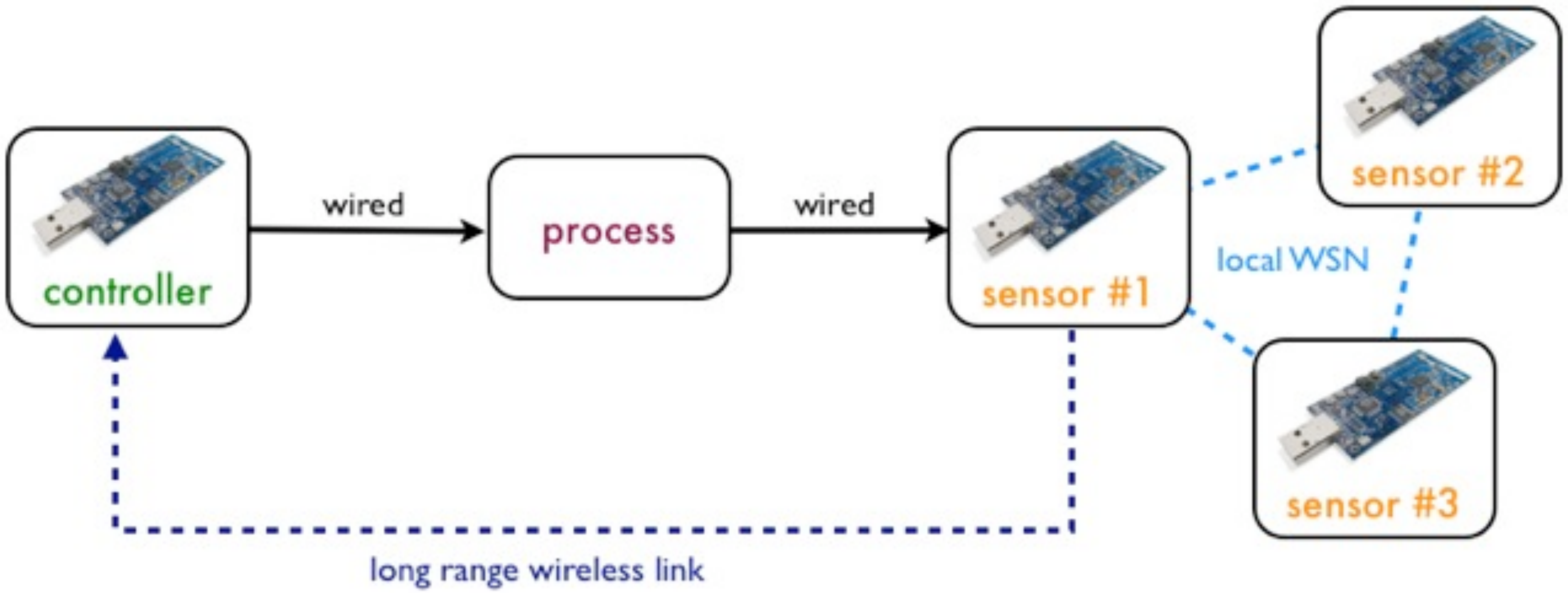
→ Transmitting a (small) set of measurements costs (almost) as transmitting a single measurement !

- Idea can be extended to **multiple sensors** and **noisy measurements**
- Asymptotic stability proofs are available (Bernardini, Bemporad, *Automatica*, to appear)

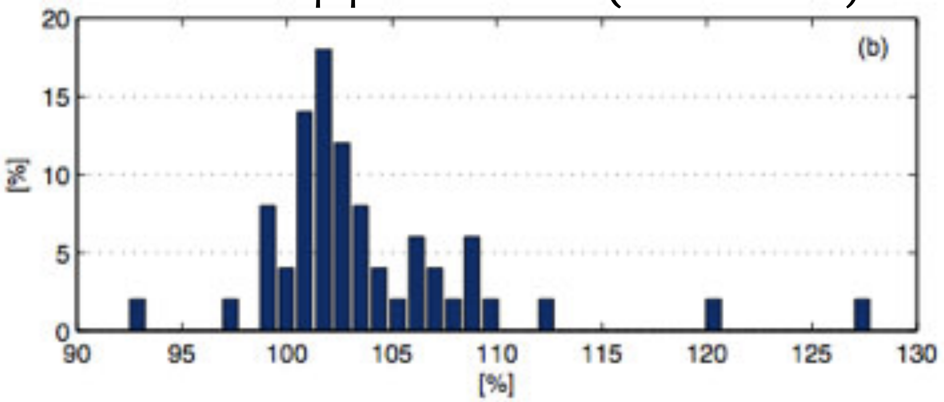
Energy-aware MPC over wireless networks

- Uncertain open-loop unstable system, 3 wireless sensor nodes

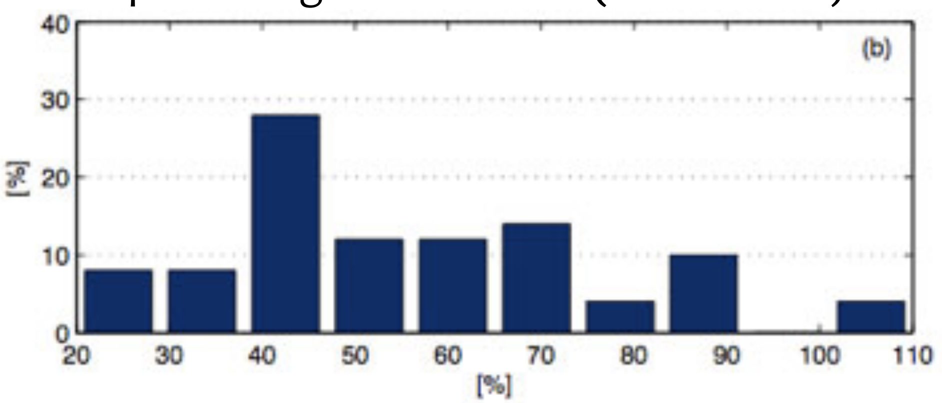
$$x(k+1) = \begin{bmatrix} 0.21 & -0.39 \\ -0.39 & 0.82 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + w(k)$$



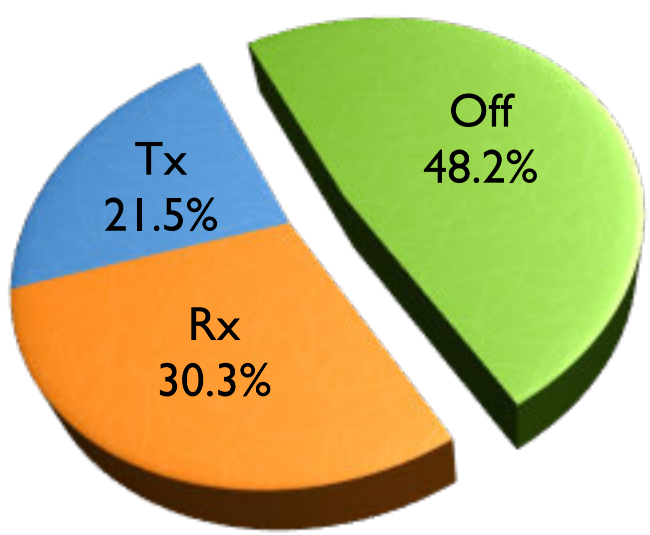
closed-loop performance (distribution)



percentage of radio use (distribution)



Controller	Performance	Tx Rate
Standard Robust MPC	2.5025	100%
Energy-Aware Robust MPC	2.5514	51.8%



Outline of this talk

✓ Quick review of MPC basics

✓ MPC that limits the use of network resources

→ Energy-aware MPC

- **MPC that tames model complexity**

→ **Decentralized MPC**

- MPC that takes into account communication imperfections

Centralized vs decentralized/distributed control

Centralized control:

- Need a **global model** of the overall networked system (and its maintenance)
- Complexity **not scalable** with model size
- Computation complexity may become prohibitive
- Control design hard to commission, start-up, and **maintain** (many tuning "knobs")
- High **risk**: a single controller is running the whole plant
- Good **theoretical** properties (e.g. closed-loop stability)

Centralized vs decentralized/distributed control

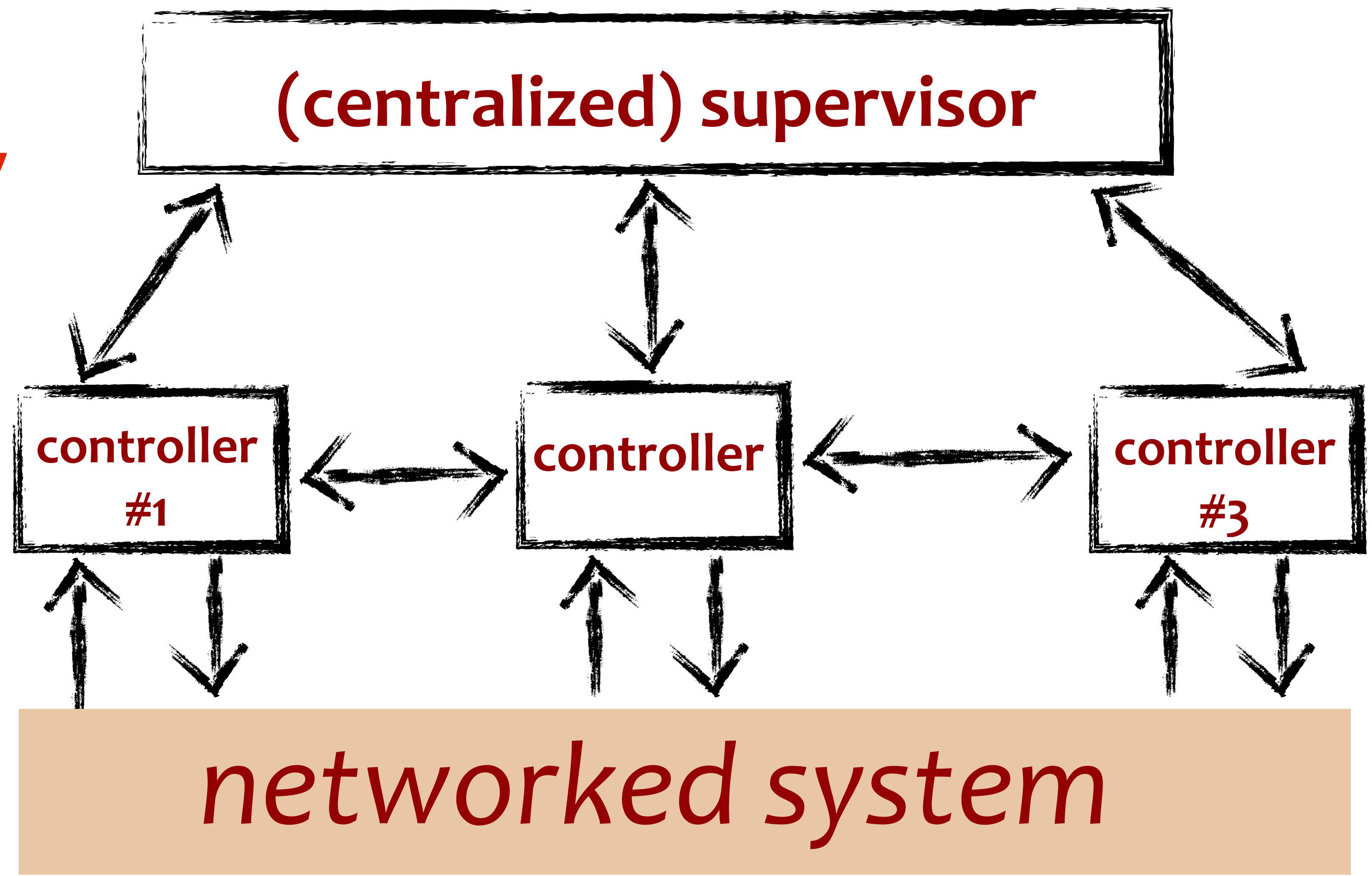
Decentralized control:

- **Local models** of networked components are enough
- Computational tasks are **parallelized**, each task is simple
- Data gathering is simpler (**local measurements** used only locally)
- Commissioning, start-up, and **maintenance** more practical (controller updates do not require a whole system shutdown)
- **Global properties** (stability, performance) hard to assess, especially in the presence of input/state constraints

⇒ Careful cooperation of controllers is needed to ensure global properties (such as stability and constraint fulfillment)

Typical decentralized approach

- Measure/estimate **local** states
- Compute control actions **locally**
- **Exchange** decisions with neighbors, possibly reiterate local computations
- Apply the current command input to local actuator(s)
- Possibly interact with upper level of decision making (**hierarchical control**) (Barcelli, Bemporad, Ripaccioli, IFAC'11)



Main issues: Global closed-loop stability ? Feasibility of global constraints ? Loss of performance w.r.t. centralized control ?

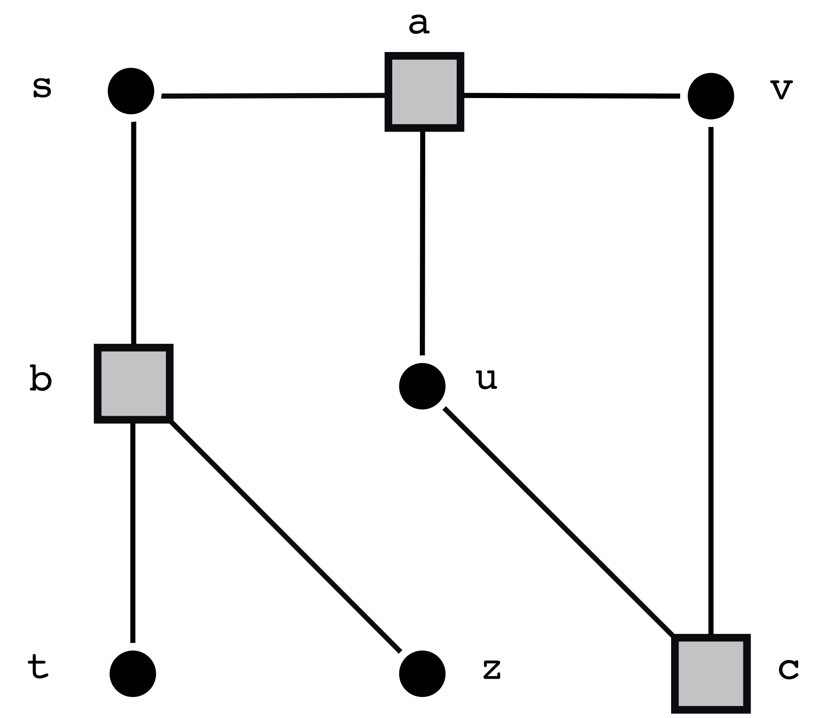
Decentralized linear control: Synthesis is easy !

global model

$$x(t+1) = Ax(t) + Bu(t)$$

constraints

$$\|x(t)\|_2 \le x_{\max}$$
$$\|u(t)\|_2 \le u_{\max}$$

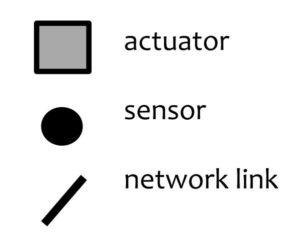


How to take into account network topology in synthesizing the control law $u(t) = Kx(t)$?

adjacency matrix Λ

$$\lambda_{ij} = \begin{cases} 1 & \text{if sensor } s_j \text{ is linked to actuator } a_i, \\ 0 & \text{otherwise.} \end{cases}$$

$$\Lambda = \begin{array}{c} \\ \\ \end{array}\begin{array}{cc} \begin{array}{ccccc} t & z & v & u & s \end{array} & \\ \left[\begin{array}{ccccc} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{array}\right] & \begin{array}{c} a \\ b \\ c \end{array} \end{array}$$



impose on controller K the same structure of Λ

$$\lambda_{ij} = 0 \rightarrow k_{ij} = 0$$

Decentralized linear control: Synthesis is easy !

(Crusius, Trofino, IEEE TAC, 1999)

Main idea to **impose decentralized structure** on K

- let $K = YQ^{-1}$, Y and Q are the unknowns

- impose on Y same structure of adjacency matrix Λ

Example:

$$\Lambda = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \Longrightarrow \quad Y = \begin{bmatrix} 0 & 0 & * & * & * \\ * & * & 0 & 0 & * \\ 0 & 0 & * & * & 0 \end{bmatrix}$$

- impose block-diagonal structure on Q

$$Q = \begin{bmatrix} * & * & 0 & 0 & 0 \\ * & * & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * \end{bmatrix} \quad \Longrightarrow \quad K = YQ^{-1} = \begin{bmatrix} 0 & 0 & * & * & * \\ * & * & 0 & 0 & * \\ 0 & 0 & * & * & 0 \end{bmatrix}$$

Decentralized linear control: Synthesis is easy !

Theorem: Given network topology Λ, a *decentralized* **linear** *controller* $K = YQ^{-1}$ stabilizing the system under *state and input constraints* is obtained by solving the following ***semidefinite program***:

$$\min_{\gamma,Q,Y} \quad \gamma$$

$$\text{s.t.} \quad \begin{bmatrix} Q & * & * & * \\ AQ+BY & Q & * & * \\ Q_x^{1/2}Q & \mathbf{0} & \gamma I_n & * \\ Q_u^{1/2}Y & \mathbf{0} & \mathbf{0} & \gamma I_m \end{bmatrix} \succeq 0,$$

decreasing condition on Lyapunov function

$$\begin{bmatrix} Q & (AQ+BY)' \\ AQ+BY & x_{\max}^2 I_n \end{bmatrix} \succeq 0,$$

state constraints satisfaction

$$\begin{bmatrix} u_{\max}^2 I_m & Y \\ Y' & Q \end{bmatrix} \succeq 0,$$

input constraints satisfaction

$$\begin{bmatrix} 1 & {v_i}' \\ v_i & Q \end{bmatrix} \succeq 0, \; i = 1, \ldots, n_v,$$

initial condition $x(0) \in \mathcal{X}_0$ $\quad v = \text{vertex}(\mathcal{X}_0)$

$$\left.\begin{array}{l} (\lambda_{ij} = 0) \Rightarrow y_{ij} = 0 \\ (\lambda_{ij} = 0) \wedge (\lambda_{ih} = 1) \;\Rightarrow\; q_{hj} = 0, \; q_{jh} = 0 \end{array}\right\} \begin{array}{l} i = 1, \ldots, m, \\ j = 1, \ldots, n, \\ h = 1, \ldots, n \end{array}$$

decentralized structure

The idea can be extended to **robust** and **stochastic** case

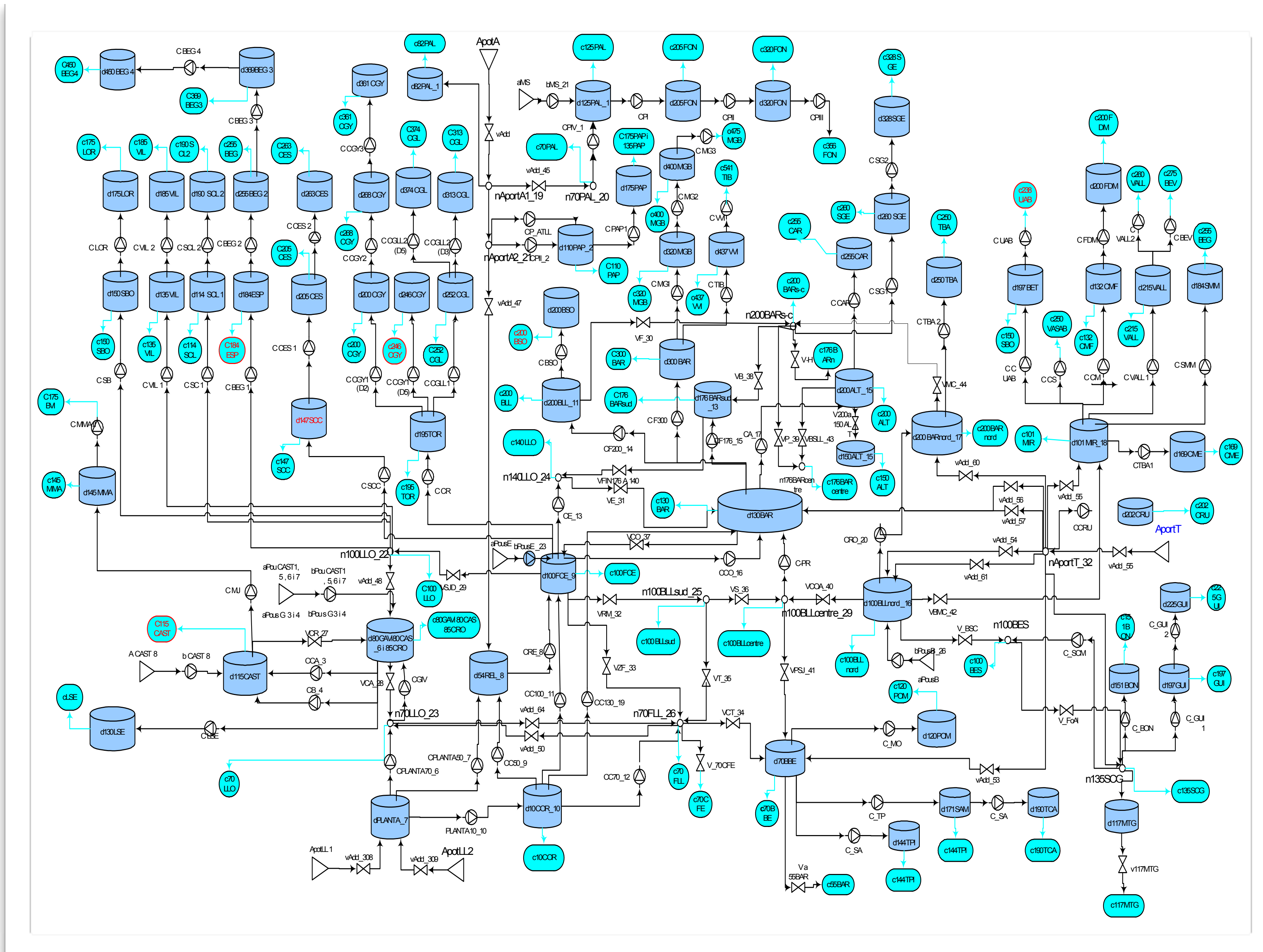
(Barcelli, Bernardini, Bemporad, CDC 2010)

Decentralized/distributed MPC

submodels	constraints	intersampling iterations	broadcast prediction	state constraints	stability constraints	authors
coupled	local inputs	no	no	no	none	Alessio, Barcelli, Bemporad
coupled	local inputs	yes	no	no	none	Venkat, Rawlings, Wright
coupled	local inputs	yes	yes	no	none	Mercangöz, Doyle
decoupled	local inputs	no	yes	yes	compatibility	Dunbar, Murray
decoupled		no	yes	yes	none	Keviczy, Borrelli, Balas
coupled	local states	no	yes	yes	contractive	Jia, Krogh
coupled/NL	local inputs	no	no	no	contractive	Magni, Scattolini

Alternative approach: ***distribute the optimization*** problem associated with the centralized MPC formulation, instead of distributing the problem formulation

Distributed MPC of Barcelona water network



- General overview:

Municipalities supplied	23
Supply area	424 km^2
Population supplied	2.922.773
Average demand	7 m^3/s

- Network parameters:

Pipes length	4.645 km
Pressure floors	113
Sectors	218

- Facilities

Remote stations	98
Water storage tanks	81
Valves	64
Flow meters	92
Pumps / Pumping stations	180 / 84
Chlorine dosing devices	23
Chlorine analyzers	74

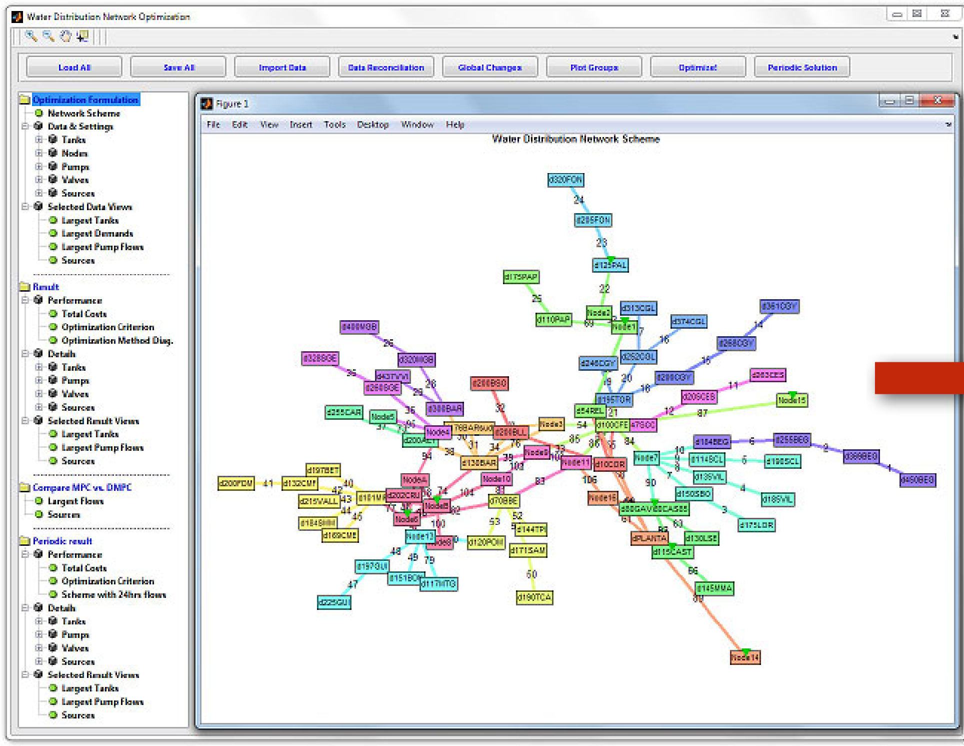
European FP7-ICT project WIDE
"DEcentralized and WIreless Control of Large-Scale Systems"

UNITN

DMPC of Barcelona water network

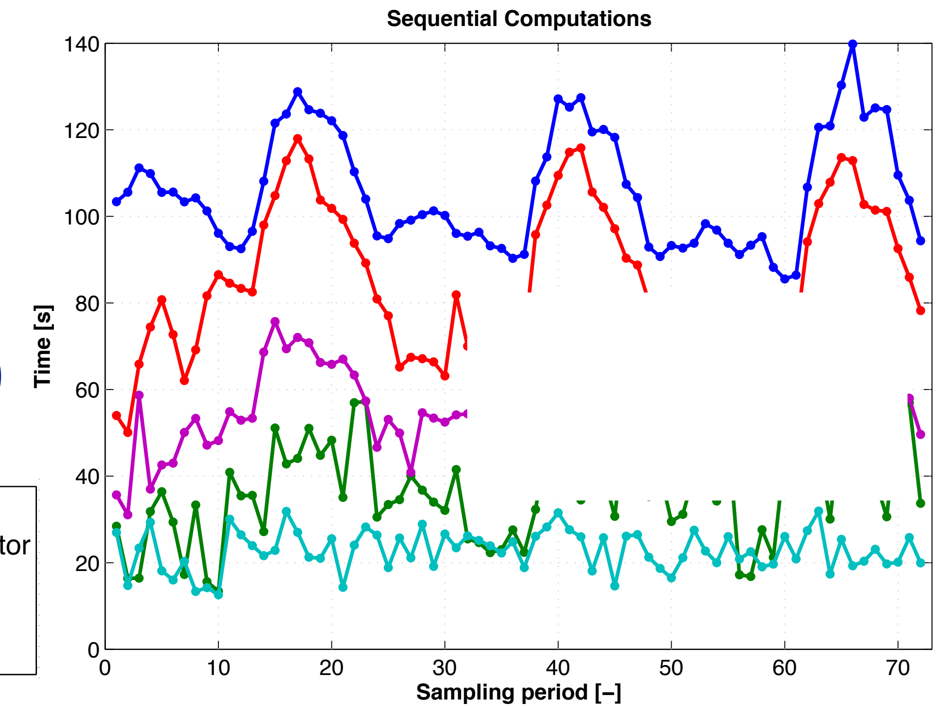
(Trnka, Pekar, Havlena, IFAC 2011)

Honeywell

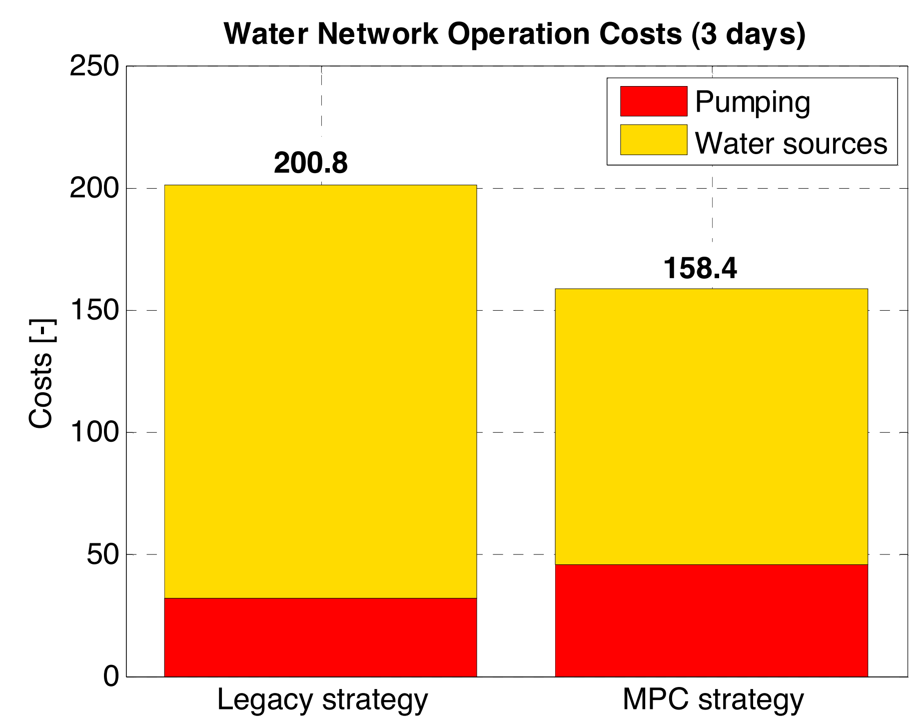


Network separation to groups

distributed MPC algorithm (optimal solution)



- Benefits evaluated on 3 days historic data set
- Benefits evaluation is complicated by water accumulation (different final accumulation for different control strategies)
- To get comparable data MPC was forced to fill tanks to the same final levels as in historical data (sub-optimal)
- ~20% direct cost savings (pumping and water sources)
- Indirect savings by smooth MV's operation -> leakage prevention by small pressure surges and reduced equipment tear & wear



A decentralized MPC approach

Consider a decentralization of the following MPC problem (with horizon $N=1$)

(Alessio, Barcelli, Bemporad, 2011)

$$V(x(t)) = \min_{\{u_k\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} x_k' Q x_k + u_k' R u_k$$

$$= \min_{u_0} x_1' P x_1 + x(t)' Q x(t) + u_0' R u_0$$

$$\text{s.t. } x_1 = Ax(t) + Bu_0$$

$$u_{\min} \le u_0 \le u_{\max}$$

$$u_k = 0, \ \forall k \ge 1$$

global weights

$$Q = Q' \ge 0$$
$$R = R' > 0$$

terminal weight

$$P = A'PA + Q$$

Assumption 1: open-loop system is asymptotically stable

Assumption 2: only input constraints $u_{\min} \le u(t) \le u_{\max}, \ u(t) \in \mathbb{R}^m$

A decentralized MPC approach

Main idea: replace a centralized MPC algorithm with $M=m$ simpler decentralized MPC algorithms, one for each actuator (*generalization*: one per group of actuators)

Prediction models: assuming that certain components matrices A,B are negligible, form m local submodels

$$x^i(t+1) = A_i x^i(t) + B_i u^i(t), \;\; i = 1, \ldots, m$$

where x^i and u^i collect local states and inputs of submodel #i

$$x^i = \begin{bmatrix} x_1 \\ \cancel{x_2} \\ \vdots \\ \\ x_{n-1} \\ \cancel{x_n} \end{bmatrix} \in \mathbb{R}^{n_i} \qquad\qquad u^i = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \\ \cancel{u_{m-1}} \\ \cancel{u_m} \end{bmatrix} \in \mathbb{R}^{m_i}$$

$$x^i = W_i' x \in \mathbb{R}^{n_i} \qquad\qquad u^i = Z_i' u \in \mathbb{R}^{m_i}$$

Note: The choice of decentralization scheme is a tuning knob of DMPC !

from fully centralized ($x^i\!=\!x,\ u^i\!=\!u$) to fully decentralized ($x^i\!=\!x_i,\ u^i\!=\!u_i$)

Local DMPC problem

- At time t, each DMPC agent $\#i$ solves the local MPC problem

$$
\begin{aligned}
V_i(x(t)) &= \min_{\{u_k^i\}_{k=0}^{\infty}} \sum_{k=0}^{\infty} (x_k^i)' Q_i x_k^i + (u_k^i)' R_i u_k^i \\
&= \min_{u_0^i} (x_1^i)' P_i x_1^i + x^i(t)' Q_i x^i(t) + (u_0^i)' R_i u_0^i \\
&\quad \text{s.t. } x_1^i = A_i x^i(t) + B_i u_0^i \\
&\qquad x_0^i = W_i' x(t) = x^i(t) \\
&\qquad u_{\min}^i \leq u_0^i \leq u_{\max}^i \\
&\qquad u_k^i = 0, \ \forall k \geq 1
\end{aligned}
$$

local weights

$$
\begin{aligned}
Q_i &\triangleq W_i' Q W_i \\
R_i &\triangleq Z_i' R Z_i
\end{aligned}
$$

local states

local optimizer: $u_0^{i*} = \begin{bmatrix} u_0^{i1*} \\ \vdots \\ u_0^{ii*} \\ \vdots \\ u_0^{im_i*} \end{bmatrix}$

this is the only input applied to the system: $u_i(t) = u^{ii*}(t)$

these are not applied, but provide an estimate of neighbors' optimal moves (depending on amount of model mismatch)

The commanded $u(t)$ is the collection of all optimal inputs $u^{11*}(t)$, … , $u^{mm*}(t)$

Stability issues in DMPC

Stability of each MPC sub-problem does not guarantee plant-wide stability !

Trajectory of state $x_j(t+k)$ predicted by controller MPC #i at time t

$\cancel{\Updownarrow}$

Trajectory of state $x_j(t+k)$ predicted by controller MPC #j at time t

Because of **prediction mismatch**, in general $u^{ij*} \neq u^{jj*},\ \ j \neq i$

that is, the move u_0^{ji*} optimized by local MPC controller #i for input $u_j(t)$ is not exactly the control move $u_j(t)=u_0^{jj*}$ actually applied to the process

Stability result

input mismatch

state mismatch

Theorem Assume A and all matrices A_i open-loop as. stable. Let $P_i = A_i' P A_i + Q_i$, $\forall i = 1, \dots, M$. Define

$$\Delta u^i(t) \triangleq u(t) - Z_i u_0^{*i}(t), \qquad \Delta x^i(t) \triangleq (I - W_i W_i')x(t)$$
$$\Delta A^i \triangleq (I - W_i W_i')A, \qquad \Delta B^i \triangleq B - W_i W_i' B Z_i Z_i'$$

model mismatch

Also, let

$$\Delta Y^i(x(t)) \triangleq W_i W_i'(A\Delta x^i(t) + BZ_i Z_i' \Delta u^i(t)) + \Delta A^i x(t) + \Delta B^i u(t)$$

and

$$\Delta S^i(x(t)) \triangleq \left(2(A_i W_i' x(t) + B_i u_0^{*i}(t))' + \Delta Y^i(x(t))' W_i\right) P_i W_i' \Delta Y^i(x(t))$$

If the condition

$$x' \left(\sum_{i=1}^{M} W_i W_i' Q W_i W_i' \right) x - \sum_{i=1}^{M} \Delta S^i(x) \geq 0, \;\; \forall x \in \mathbb{R}^n$$

is satisfied then the decentralized MPC scheme is globally asymptotically stabilizing.

(Alessio, Barcelli, Bemporad, 2011)

overall mismatch

$\Delta S_i(t)$ depends on amount of mismatch between global and local models.

$\Delta S_i(t)=0$ if no mismatch, i.e. all local models are also global (=full feedback)

Stability result

- Proof based on showing that $V(x(t)) = \sum_{i=1}^{M} V_i(x(t))$ is a Lyapunov function:

$$V_i(x(t+1)) \leq V_i(x(t)) - x'(t)W_iQ_iW_i'x(t) - u_0^{*i}(t)'R_iu_0^{*i}(t) + \Delta S^i(x(t))$$

quadratic fcn
of x around
the origin

- **Local asymptotic stability:** check eigenvalues of nxn matrix

- **Global asymptotic stability:** compute explicit solution of local MPC controllers & test LMI relaxation of resulting PWA closed-loop

- Extension to *open-loop unstable* (unconstrained) systems: use terminal weight = Riccati matrix for local submodel.

A simple DMPC example

- Global model $x(t+1) = \begin{bmatrix} -0.0283 & -0.2125 & 0.9251 \\ -0.7008 & -0.6440 & -0.1483 \\ 0.6401 & -0.6841 & -0.1736 \end{bmatrix} x(t) + \begin{bmatrix} 1.4316 & 0.3223 & 0.1843 \\ 0.3281 & 1.1318 & 0.2688 \\ 0.1423 & 0.0874 & 1.4632 \end{bmatrix} u(t)$

- Local submodels:

$$A_1 = \begin{bmatrix} -0.0283 & 0.9251 \\ 0.6401 & -0.1736 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1.4316 & 0.1843 \\ 0.1423 & 1.4632 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -0.0283 & -0.2125 \\ -0.7008 & -0.6440 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1.4316 & 0.3223 \\ 0.3281 & 1.1318 \end{bmatrix}$$

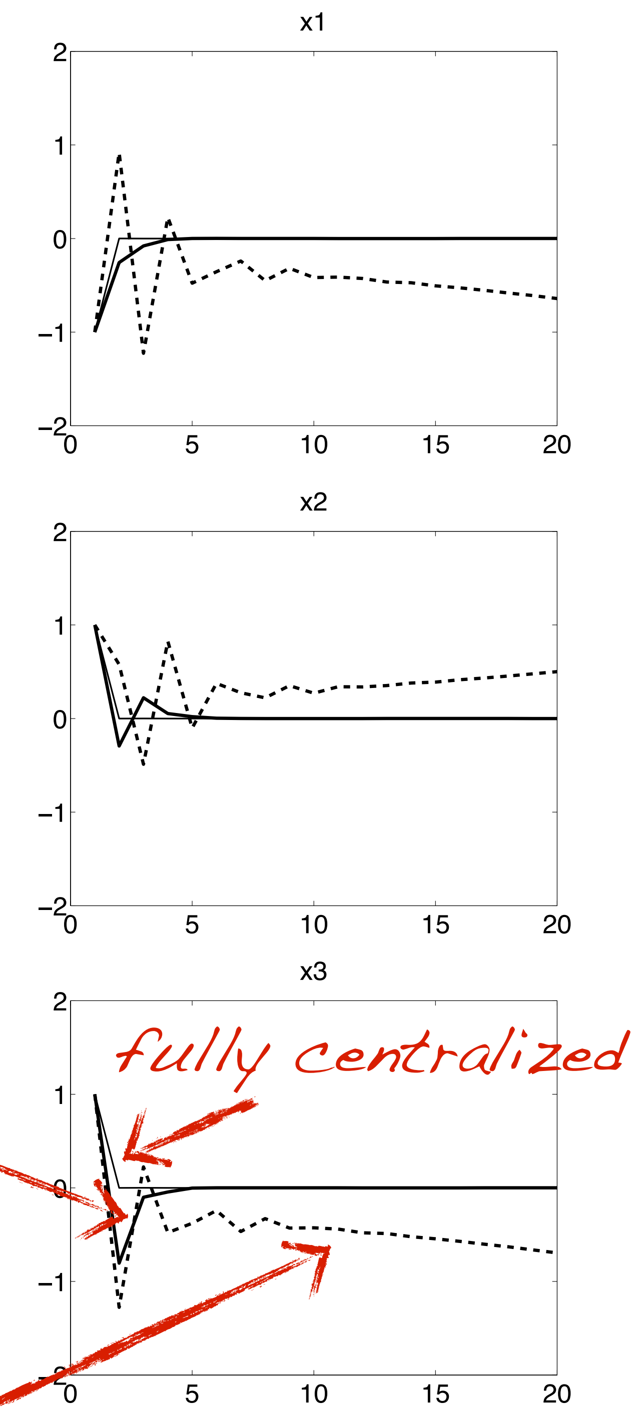
$$A_3 = \begin{bmatrix} -0.6440 & -0.1483 \\ -0.6841 & -0.1736 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 1.1318 & 0.2688 \\ 0.0874 & 1.4632 \end{bmatrix}$$

- Global weights:

$$Q = \begin{bmatrix} 1 & 0.3 & 0.3 \\ 0.3 & 1 & 0.3 \\ 0.3 & 0.3 & 1 \end{bmatrix}, \quad R = 10^{-6} \begin{bmatrix} 1 & 0.3 & 0.3 \\ 0.3 & 1 & 0.3 \\ 0.3 & 0.3 & 1 \end{bmatrix}$$

- DMPC law:

$$u(t) = \begin{bmatrix} 0.0771 & 0 & -0.6699 \\ 0.6563 & 0.5627 & 0 \\ 0 & 0.4398 & 0.1124 \end{bmatrix} x(t)$$

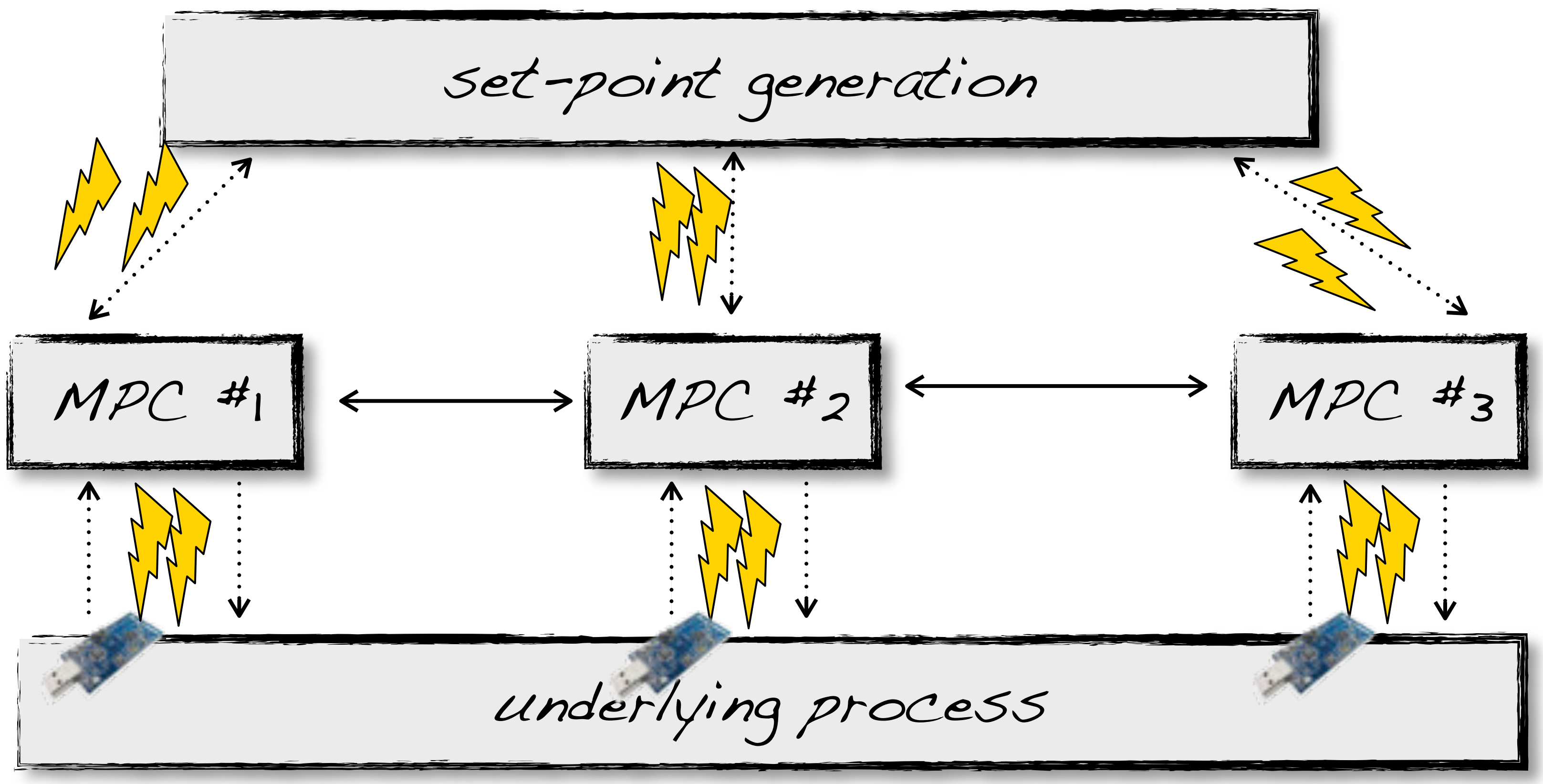


DMPC

fully decentralized

Extension to intermittent measurements

- Assume all data are exchanged on a wireless network
- The network may be congested and packets dropped out
- **Assumption:** when packets are lost at time t, by default $u_i(t)=0$
- **Assumption:** at most N packets can be lost consecutively
- Model mismatch grows with the number of consecutive packet losses

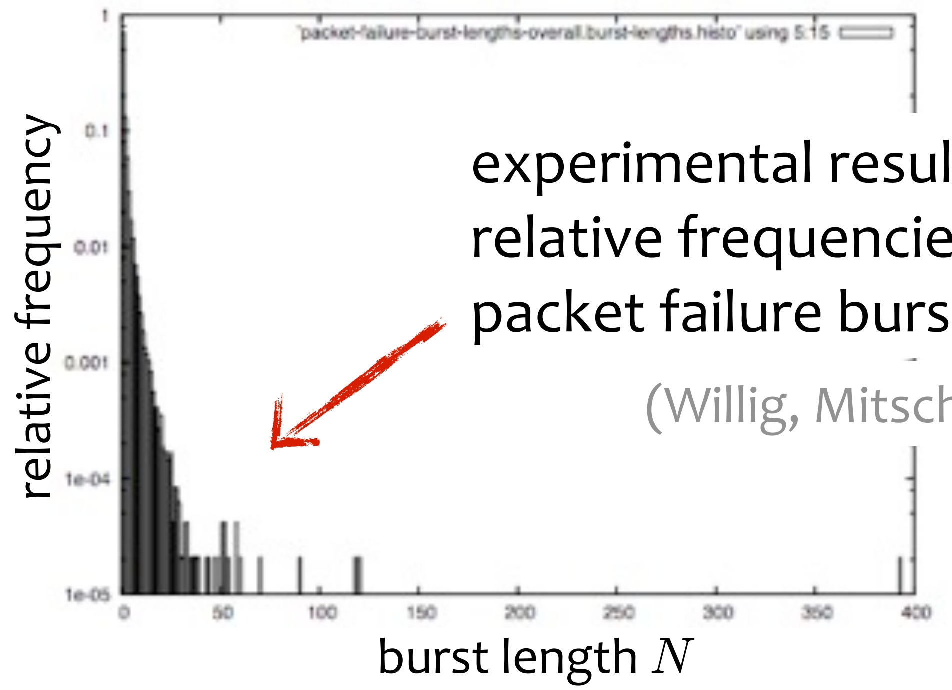


- Stability under packet loss can be proved similarly to deterministic case

(Alessio, Barcelli, Bemporad, 2011)

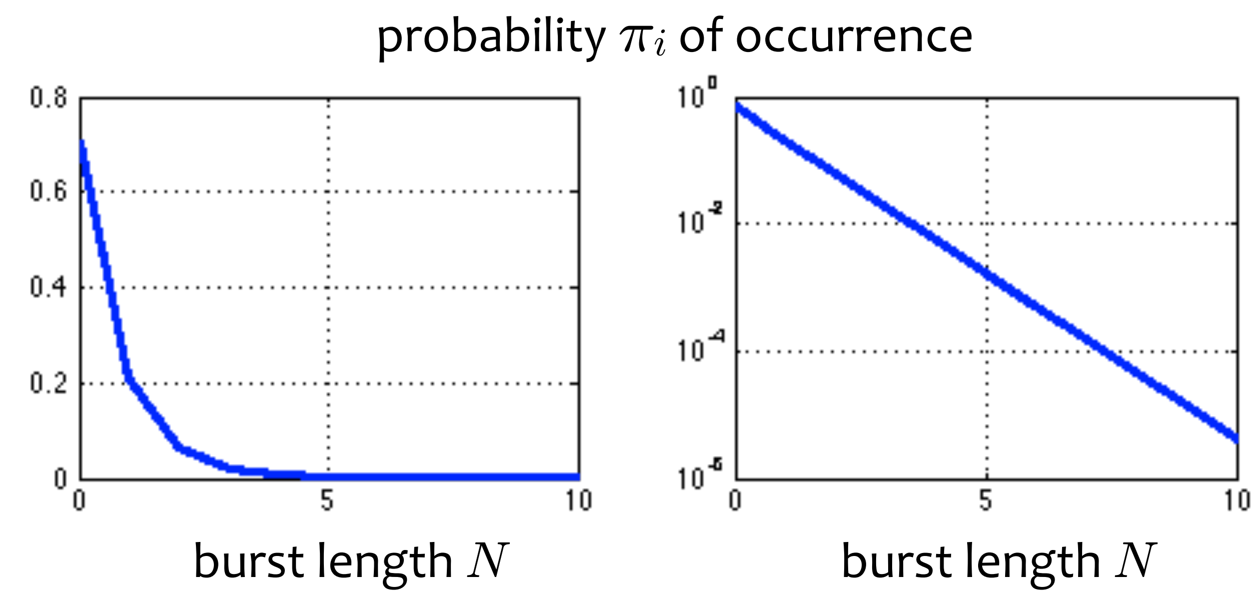
- Note: Proof does **not** depend on ***probability model*** for packet loss !

Packet-loss probability model

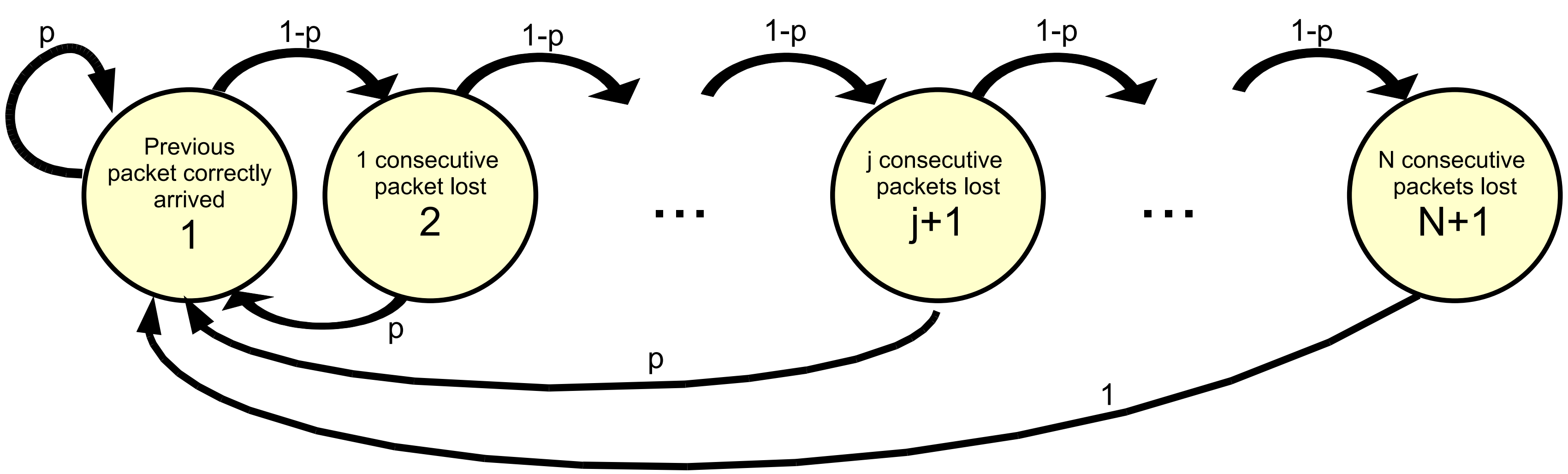


experimental results on relative frequencies of packet failure burst length

(Willig, Mitschke, 2006)



Proposed probability model: Markov chain (Barcelli, Bemporad, 2009)



$$P = \begin{bmatrix} p & 1-p & 0 & \cdots & 0 \\ p & 0 & 1-p & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p & 0 & 0 & \cdots & 1-p \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

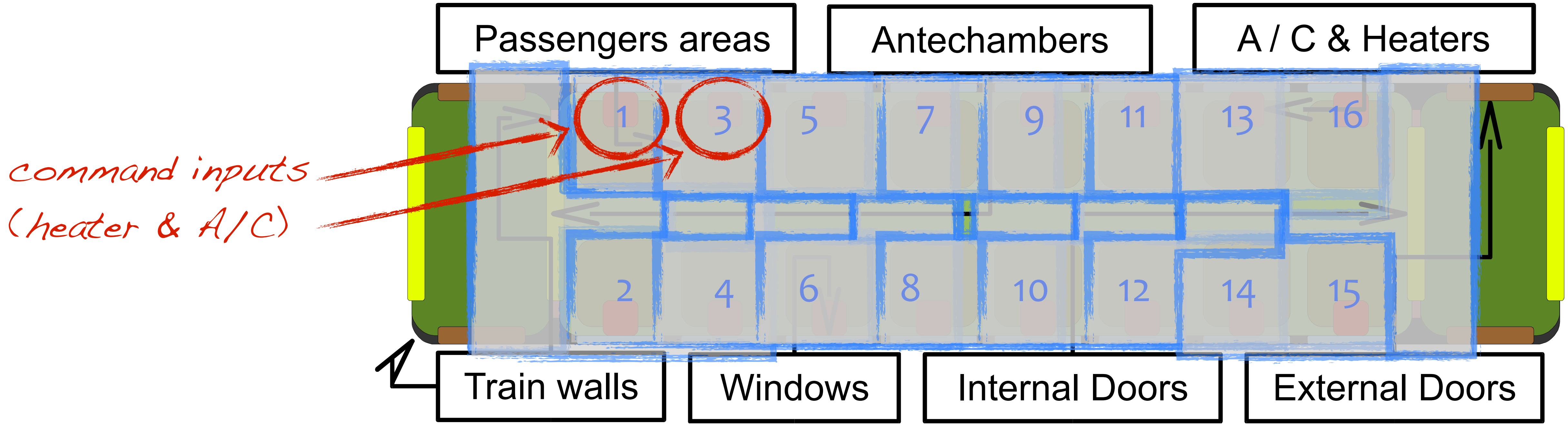
one-step probability matrix

$$\begin{cases} \pi' = \pi' P \\ \sum_{i=1}^{N} \pi_i = 1 \end{cases}$$

$\pi_i =$ steady-state probability of losing consecutively exactly i-1 packets

Decentralized temperature control example

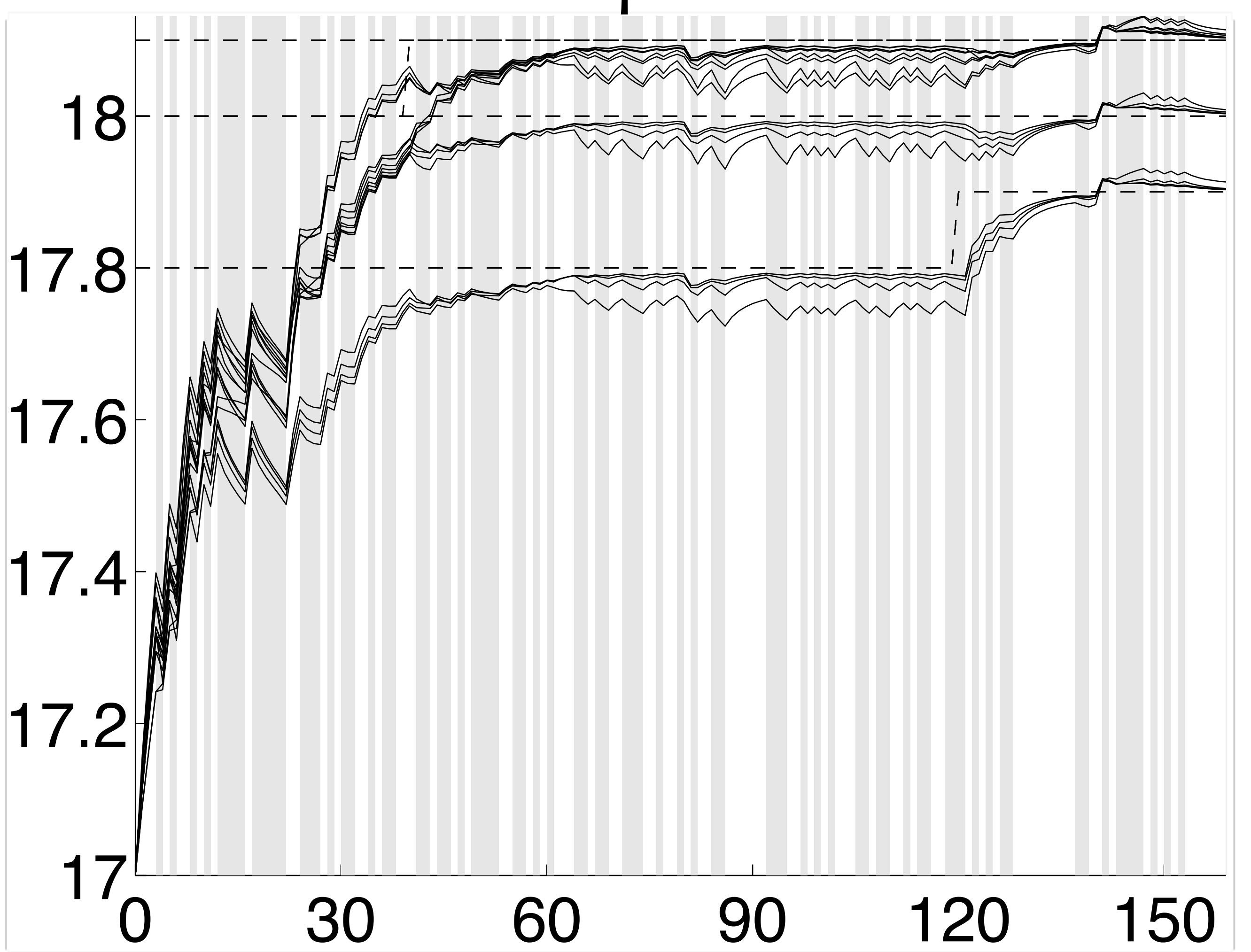
Temperature control in different passenger areas in a railcar



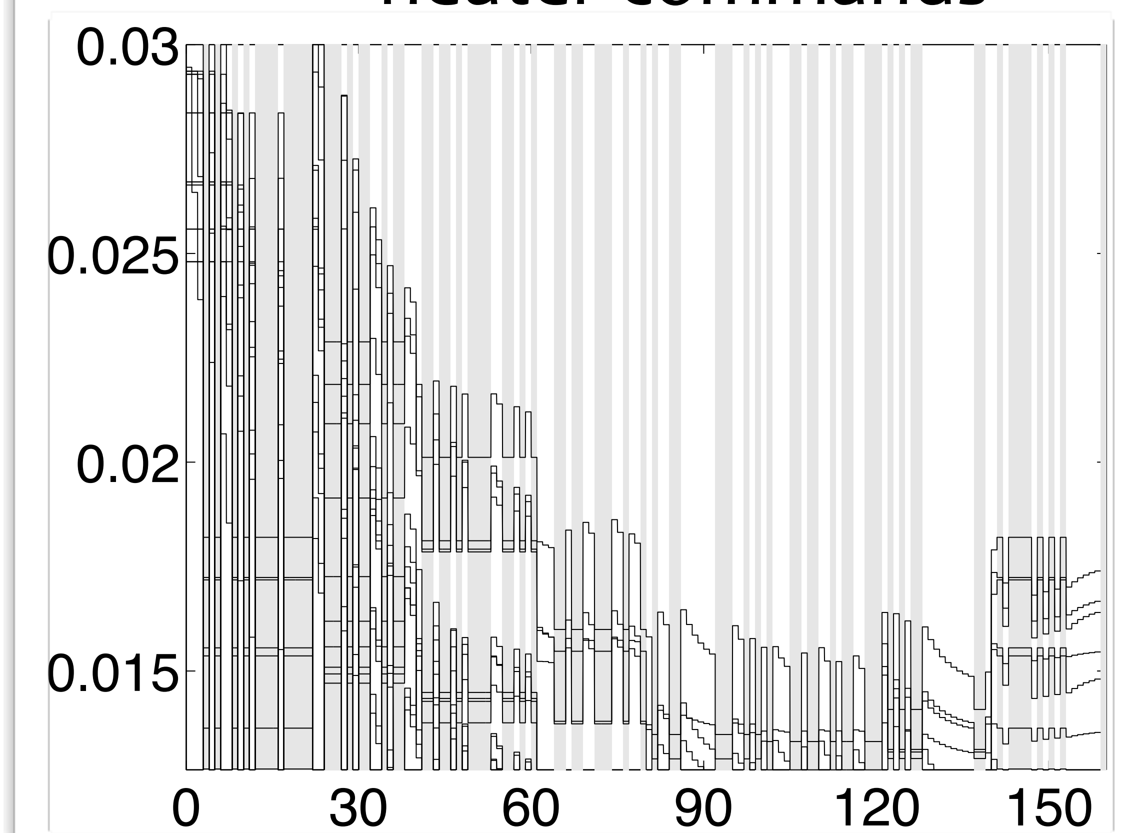
- Global model: 26 states, 16 inputs
- Decompose model into 16 submodels
- Each model has 5 states, 2 or 3 command inputs

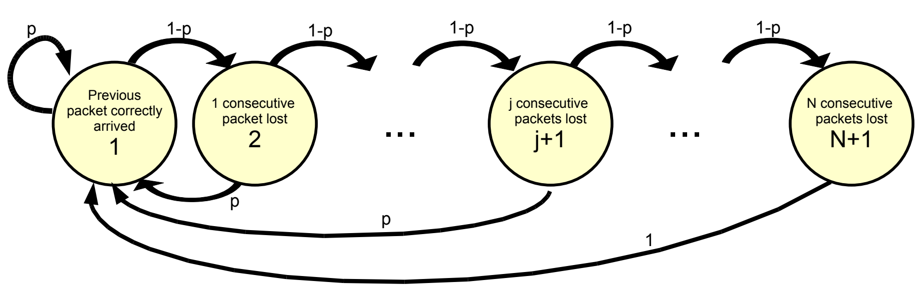
DMPC - Simulation results (with packet loss)

temperatures



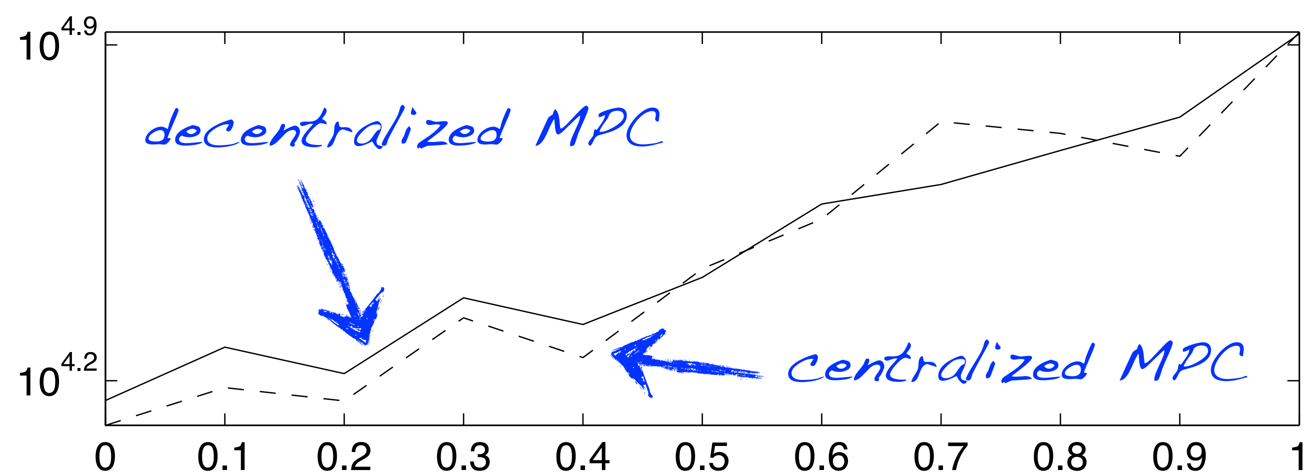
heater commands





Packet loss generated by Markov chain

centralized vs. decentralized performance



probability of packet dropout

Outline of this talk

✓ Quick review of MPC basics

✓ MPC that limits the use of network resources

→ Energy-aware MPC

✓ MPC that tames model complexity

→ Decentralized MPC

- **MPC that takes into account communication imperfections**

→ **Stochastic MPC**

Stochastic models of networked systems

- Wireless communication infrastructure introduces **uncertainty**:
  - packet dropout
  - delays (induced for example by buffers)

- **Robust** control approaches do not model uncertainty (only assume that is bounded) and pessimistically consider the worst case

- **Stochastic models** provide additional information about uncertainty.

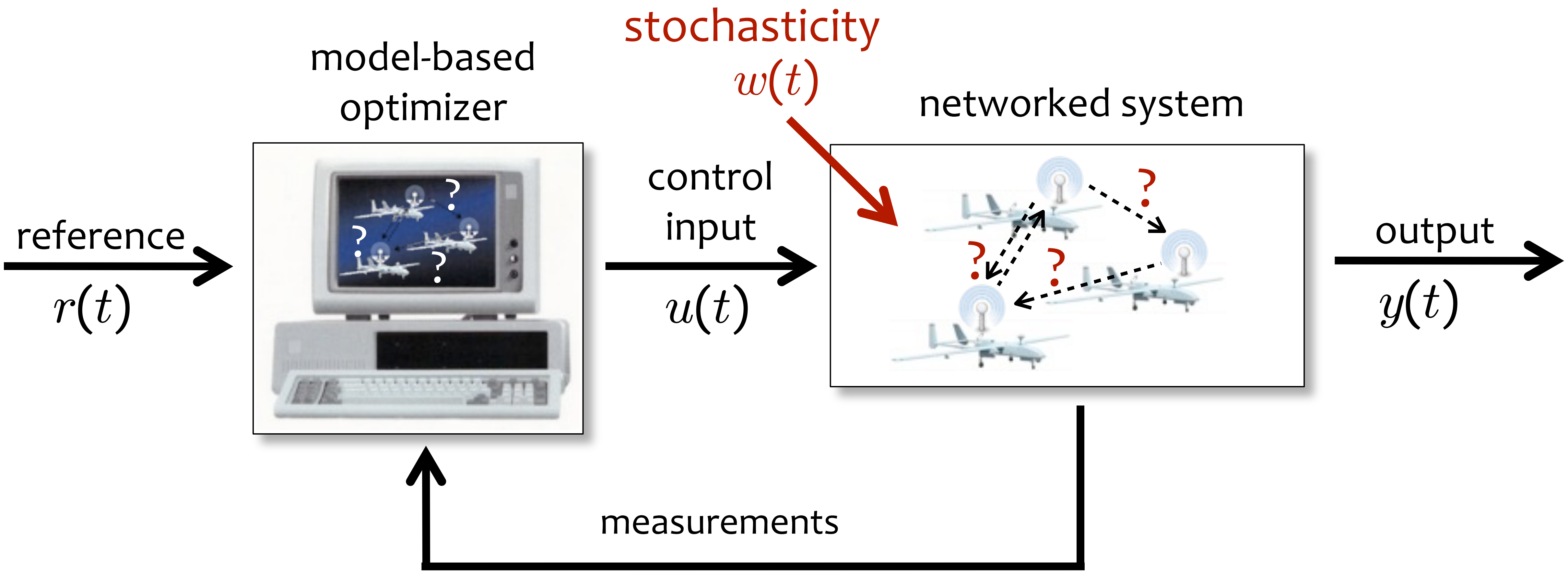
  Examples:

  - Network congestion dynamics modeled as a Markov chain, whose state determines the probability of losing a packet

  - Transmission delay described by probability distribution function

Need to include **stochastic models** in control problem formulation

Stochastic Model Predictive Control (SMPC)



*Use a **stochastic** dynamical **model** of the process to **predict** its possible future evolutions and choose the "best" **control** action*

Stochastic Model Predictive Control

- At time t: solve a **stochastic optimal control** problem over a finite future horizon of N steps:

$$
\begin{array}{ll}
\min\limits_{u} & E_w\left[\sum\limits_{k=0}^{N-1} \ell(y_k - r(t+k), u_k)\right] \\
\text{s.t.} & x_{k+1} = f(x_k, u_k, w_k) \\
& y_k = g(x_k, u_k, w_k) \\
& u_{\min} \leq u_{t+k} \leq u_{\max} \\
& y_{\min} \leq y_k \leq y_{\max}, \quad \forall w \\
& x_0 = x(t)
\end{array}
$$

$x(t)$ = process state

$u(t)$ = manipulated vars

$y(t)$ = controlled output

$w(t)$ = stochastic disturbances

- Only apply the first optimal move $u^*(t)$, discard $u^*(t+1)$, $u^*(t+2)$, ...

- At time t+1: Get new measurement $x(t+1)$, repeat the optimization. And so on ...

Linear stochastic MPC w/ discrete disturbance

- Linear stochastic prediction model

$$x(t+1) = A(w(t))x(t) + B(w(t))u(t) + E(w(t))$$

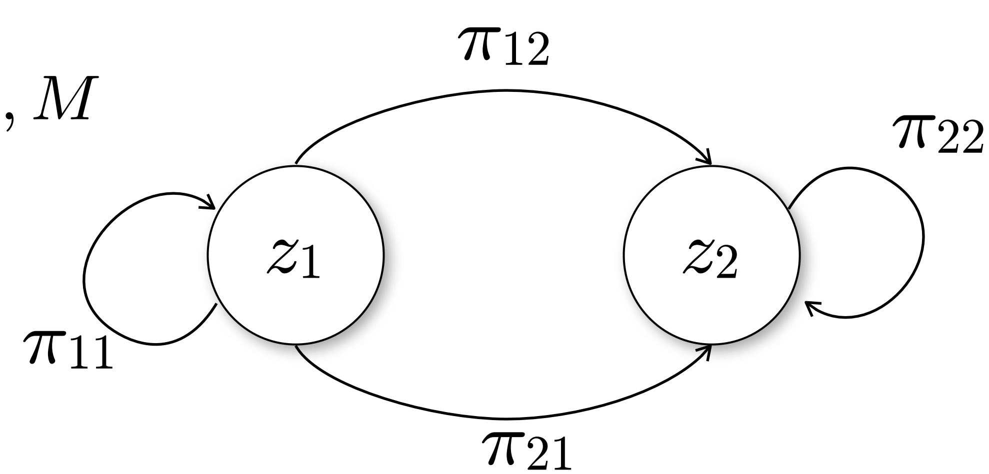
- Discrete disturbance

$$w(t) \in \{w_1, \ldots, w_s\} \qquad p_j(t) = \mathbf{Pr}[w(t) = w_j] \qquad \sum_{j=1}^{s} p_j(t) = 1$$

- Probabilities $p_j(k)$ can have their own dynamics. Example: Markov chain

$$\pi_{ih} = \mathbf{Pr}[z(t+1) = z_h \mid z(t) = z_i],\ \ i,h = 1,\ldots,M$$

$$p_j(t) = \begin{cases} e_{1j} & \text{if} \quad z(t) = z_1 \\ \vdots & \qquad \vdots \\ e_{Mj} & \text{if} \quad z(t) = z_M \end{cases}$$



- Discrete distributions can be estimated from historical data (and adapted on-line)

Linear stochastic MPC formulation

- Performance index $\quad \min E_w \left[x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k \right]$

- Goal: ensure mean-square convergence $E[x'(t)x(t)] = 0 \qquad$ (for $H=0$)

- The existence of a stochastic Lyapunov function $V(x) = x'Px$

$$\boxed{E_{w(t)}[V(x(t+1)] - V(x(t)) \leq -x(t)'Lx(t), \quad \forall t \geq 0}$$

(Morozan, 1983)

ensures mean-square stability

- Existing SMPC approaches:

(Schwarme & Nikolaou, 1999)
(Wendt & Wozny, 2000)
(Batina, Stoorvogel, Weiland, 2002)
(van Hessem & Bosgra 2002)
(Munoz de la Pena, Bemporad, Alamo, 2005)
(Couchman, Cannon, Kouvaritakis, 2006)
(Primbs, 2007)
(Bemporad, Di Cairano, 2005)
(Ono, Williams, 2008)
(Oldewurtel, Jones, Morari, 2008)
(Bernardini & Bemporad, 2009)

Stochastic program

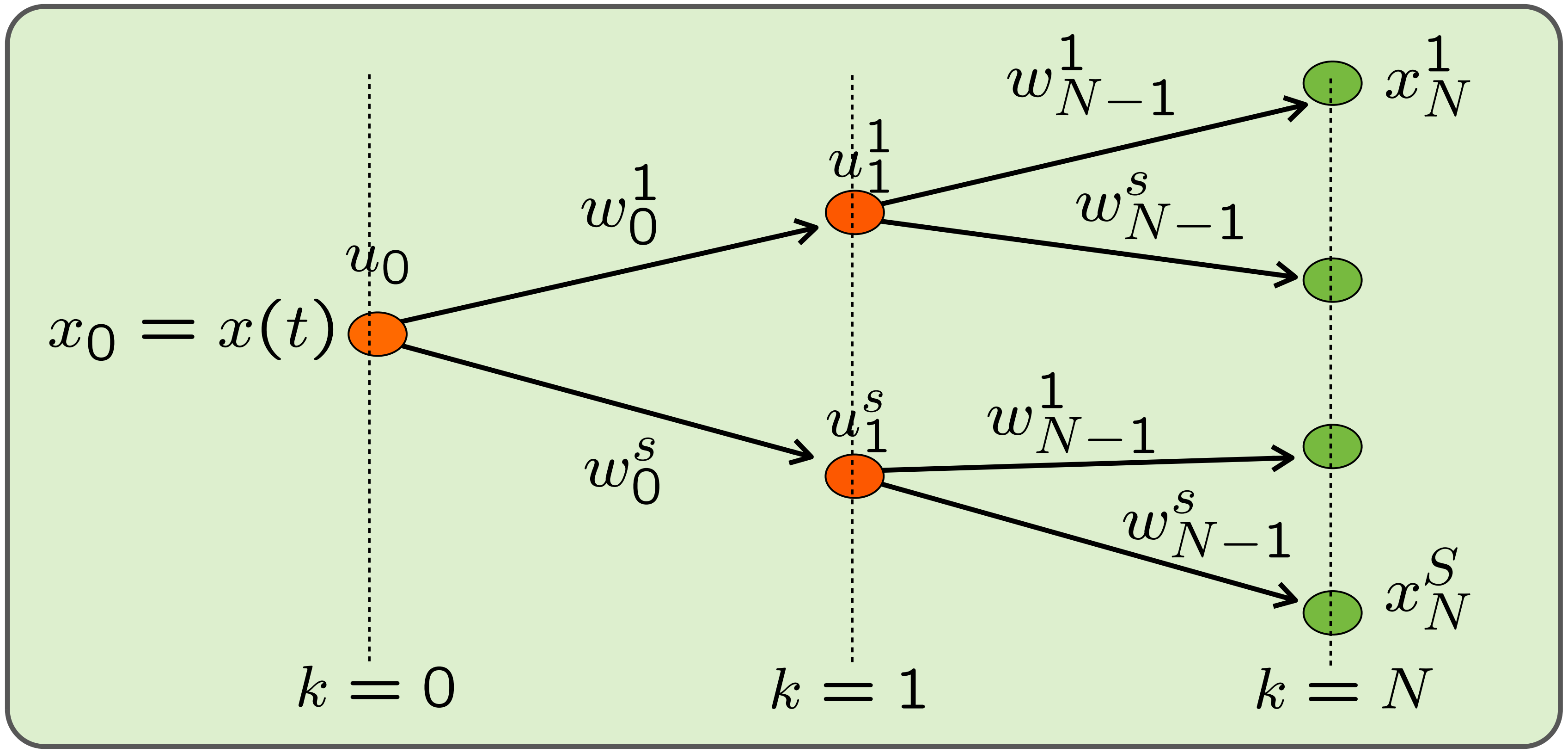
- Enumerate all possible scenarios $\{w_0^j, w_1^j, \ldots, w_{N-1}^j\},\ j = 1, \ldots, S$
- Each scenario has probability $p^j = \prod_{k=0}^{n-1} \mathbf{Pr}[w_k = w_k^j]$
- Each scenario has its own evolution $x_{k+1}^j = A(w_k^j)x_k^j + B(w_k^j)u_k^j$
- Expectations become simple sums

$$\min E_w \left[x'_N P x_N + \sum_{k=0}^{N-1} x'_k Q x_k + u'_k R u_k\right]$$

$$\min \sum_{j=1}^{S} p^j \left((x_N^j)' P x_N^j + \sum_{k=0}^{N-1} (x_k^j)' Q x_k^j + (u_k^j)' R u_k^j\right)$$

This is again a (large & sparse) QP

Scenario tree



- Scenario = path on the tree
- Number S of scenarios = number of leaf nodes

$$\boxed{\begin{array}{l}\min \ldots + p^j (x_k^j)' Q x_k^j + \ldots \\ y_{\min} \leq y_k \leq y_{\max}, \ \forall w\end{array}}$$

- Some paths can be removed if their probability is very small (at your own risk)
- **Causality constraint:** $u_k^j = u_k^h$ when scenarios j and h share the same node at prediction time k (for example: $u_0^j \equiv u_0^h$ at root node $k=0$)

Linear stochastic stabilization

- Assume $w(t) \in \{w_1, \ldots, w_s\}$ and **constant** probability $p(t) \equiv p, \ \forall t$

- The **stochastic convergence** condition $E_{w(t)}[V(x(t+1)] - V(x(t)) \leq -x(t)' L x(t)$ can be recast as the **LMI** condition

$$\begin{bmatrix} Q & Q & \sqrt{p_1}(A_1Q+B_1Y)' & \cdots & \sqrt{p_s}(A_sQ+B_sY)' \\ Q & W & 0 & \cdots & 0 \\ \sqrt{p_1}(A_1Q+B_1Y) & 0 & Q & & \\ \vdots & \vdots & & \ddots & \\ \sqrt{p_s}(A_sQ+B_sY) & 0 & & & Q \end{bmatrix} \succeq 0$$

- The Lyapunov function is $V(x) = x' Q^{-1} x$

- Mean-square stability guaranteed by linear feedback $u(k) = Kx(k), \ K = YQ^{-1}$

- A minimum decrease rate L can be imposed

- The approach can be extended to uncertain probabilities $p(t) \in \mathcal{P}$ (for example time-varying probabilities)

Stabilizing SMPC

(Bernardini, Bemporad, CDC'09)

- Impose stochastic stability constraint in SMPC problem (=quadratic constraint w.r.t. u_0)

$$\begin{array}{ll} \min\limits_{u} & E_w\left[\sum\limits_{k=0}^{N-1} \ell(x_k, u_k)\right] \\ \text{s.t.} & x_{k+1} = A(w_k)x_k + B(w_k)u_k \\ & E\left[V(A(w_0)x_0 + B(w_0)u_0)\right] \leq x_0'(Q^{-1} - L)x_0 \\ & x_0 = x(t) \end{array}$$

performance and stability are decoupled

- SMPC approach:
  1. Solve LMI problem off-line to find stochastic Lyapunov fcn $V(x) = x'Q^{-1}x$
  2. Optimize stochastic performance based on scenario tree

Theorem: The closed-loop SMPC system is as. stable in the mean-square sense

- SMPC can be generalized to handle **input and state constraints**

Note: *recursive feasibility* guaranteed by backup solution $u(k) = Kx(k)$

A few sample applications of SMPC

- **Financial engineering:** dynamic hedging of portfolios replicating synthetic options

  (Bemporad, Bellucci, Gabbriellini, 2009)
  (Bemporad, Gabbriellini, Puglia, Bellucci, 2010)
  (Bemporad,Puglia, Gabbriellini, 2011)

- **Energy systems:** power dispatch in smart grids, optimal bidding on electricity markets

  (Patrinos, Trimboli, Bemporad, CDC 2011)
  (Puglia, Bernardini, Bemporad, CDC 2011)

- **Automotive control:** energy management in HEVs, adaptive cruise control (human-machine interaction)

  (Bichi, Ripaccioli, Di Cairano, Bernardini, Bemporad, Kolmanovsky, CDC 2010)

- **Networked control:** improve robustness against communication imperfections

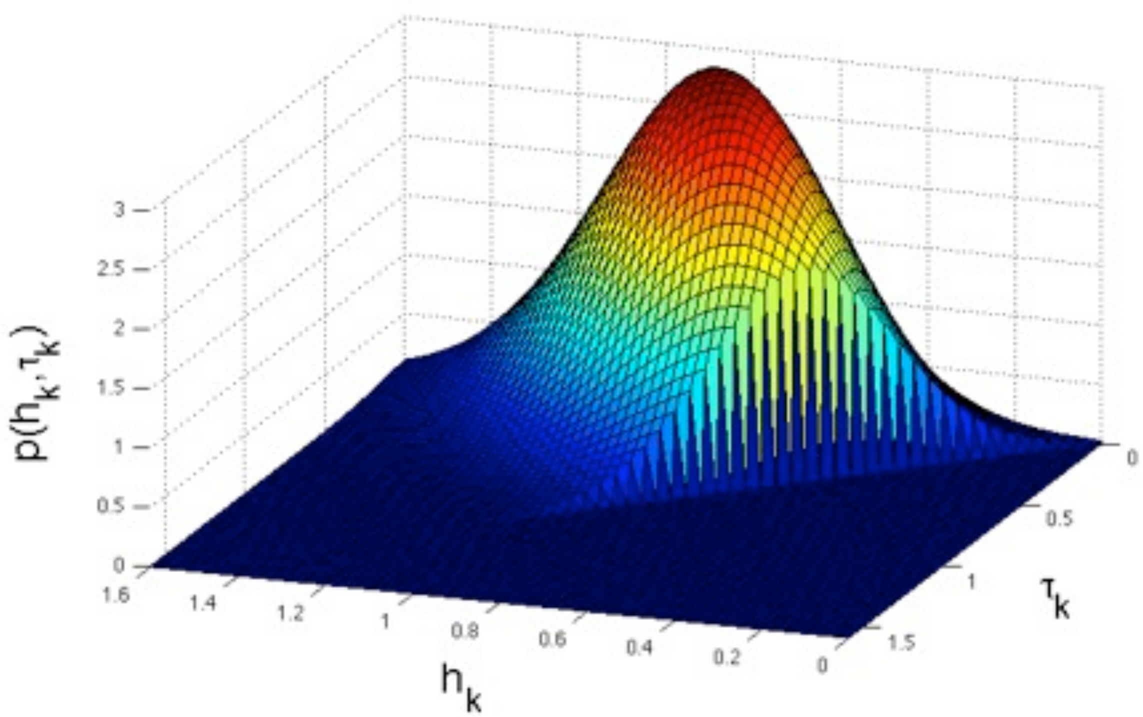
  (Bernardini, Donkers, Bemporad, Heemels, NECSYS 2010)

SMPC of networked systems

(Bernardini, Donkers, Bemporad, Heemels, NECSYS 2010)

- **Plant model:** $\dot{x}(t) = Ax(t) + Bu(t)$
- **Disturbance models** for transmission intervals h_k and delays τ_k
  - Robust models: $(h,\tau) \in [h_{min},\, h_{max}] \times [\tau_{min},\, \tau_{max}]$
  - Discrete stochastic models: $(h,\tau) \in \{\tilde{h}_1,\, \tilde{h}_2,\, \ldots,\, \tilde{h}_n\} \times \{\tilde{\tau}_1,\, \tilde{\tau}_2,\, \ldots,\, \tilde{\tau}_m\}$
  - Stochastic models with continuous distribution:

$$\Theta = \{(h,\tau) \in \mathbb{R}^2 \ \text{ such that } \ h \in (0,\, h_{max}] \ \wedge \ \tau \in [0,\, h]\}$$



- **Packet dropouts** in the sensor-to-controller channel can be modeled as prolongations of the sampling interval, by modifying the p.d.f. $p(h,\tau)$

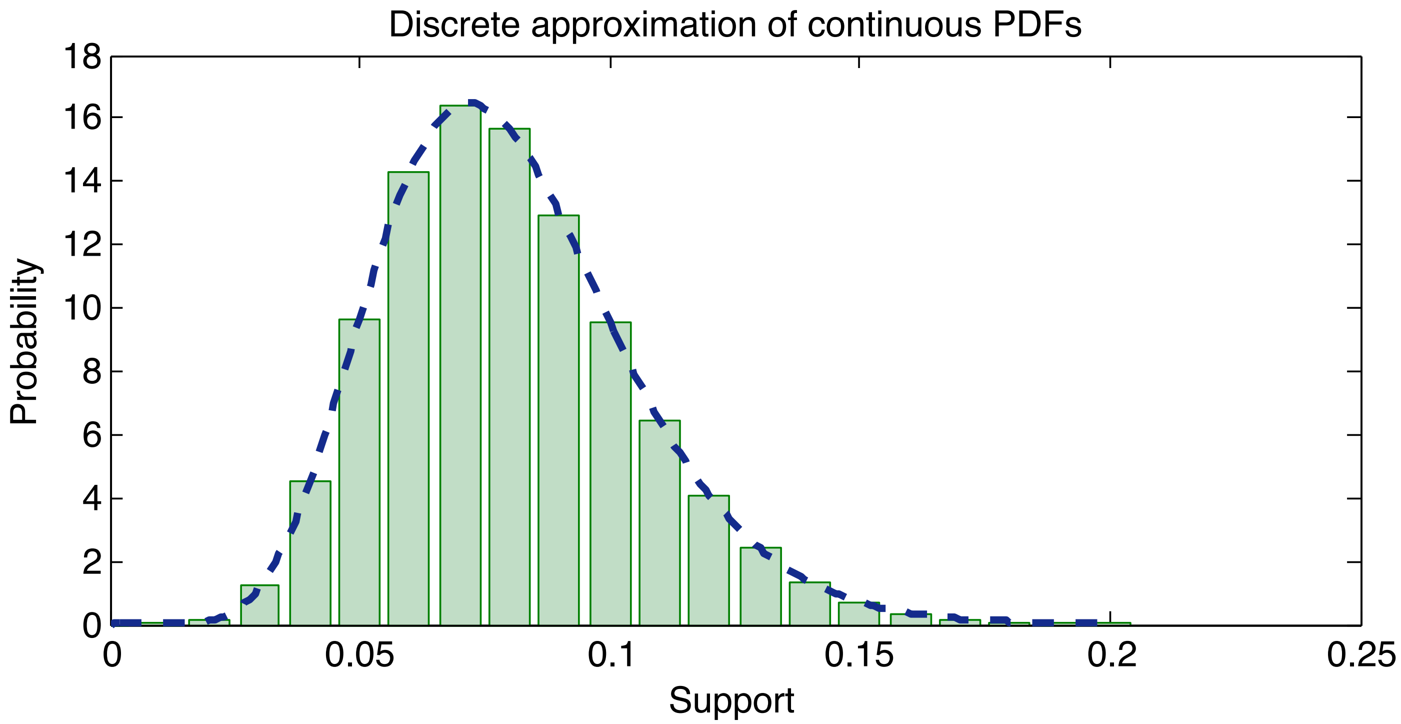
SMPC of networked systems

- **Problem:** given a plant model and a distribution $p(h, \tau)$, design a controller to optimize closed-loop performance while providing stability in the mean-square sense

- **Approach:**
  - Build a **discrete-time model** of the overall NCS by convex over-approximation techniques (Heemels, Bemporad , CDC 2011)
  - Derive LMI **conditions for stability** based on quadratic Lyapunov functions
  - Formulate the SMPC **control problem** incorporating performance optimization and stability constraints

SMPC of networked systems

- **SMPC-based control scheme:**
  - A **new partition** is defined by discretizing the parameter space
  - A **realization probability** is associated to every region



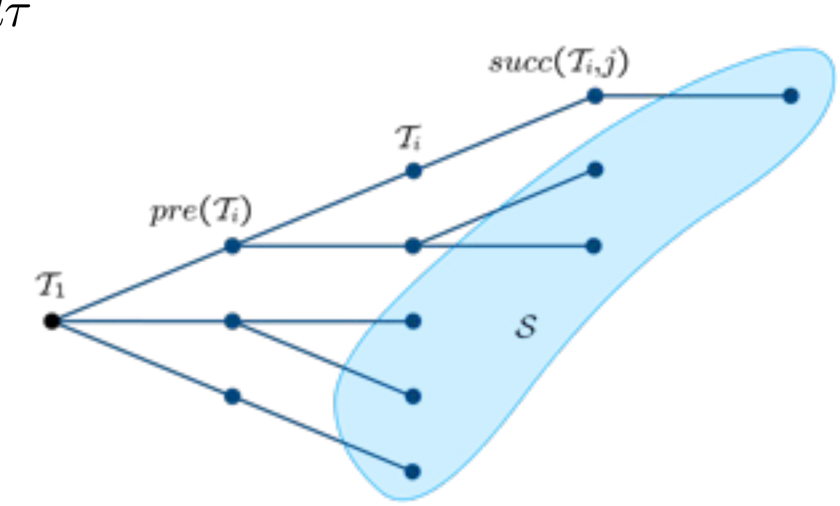
  - A **prediction model** based on the new partition is used by SMPC (*decoupling* between performance and stability)

$$\xi_{k+1} = \begin{cases} \bar{A}_1\xi_k + \bar{B}_1 u_k & \text{if } (h_k, \tau_k) \in \phi_1, \\ \bar{A}_2\xi_k + \bar{B}_2 u_k & \text{if } (h_k, \tau_k) \in \phi_2, \\ \vdots & \vdots \\ \bar{A}_s\xi_k + \bar{B}_s u_k & \text{if } (h_k, \tau_k) \in \phi_s, \end{cases}$$

$$\bar{A}_n = \iint_{\phi_n} \tilde{A}_{h,\tau} p(h,\tau) dh d\tau$$

$$\bar{B}_n = \iint_{\phi_n} \tilde{B}_{h,\tau} p(h,\tau) dh d\tau$$

  - A **scenario-based control problem** is solved at every k incorporating the minimization of a performance objective and constraints for mean-square stability



SMPC of networked systems - An example

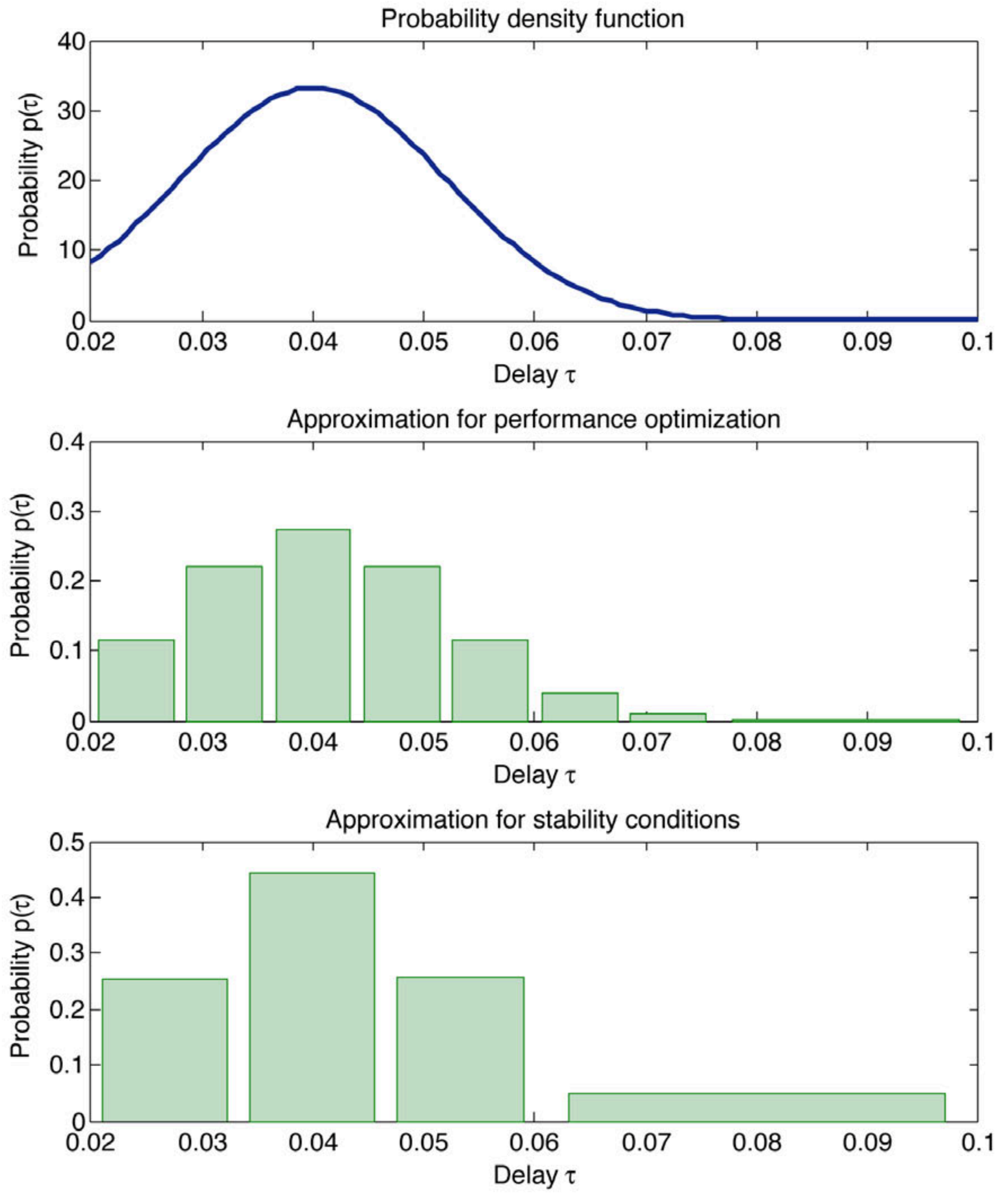
- **System model:** 2nd-order linear system

$$\dot{x}(t) = \begin{bmatrix} 1 & 15 \\ -15 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} u(t)$$

- **Sampling intervals:** constant $h_{nom} = 0.1\text{s}$

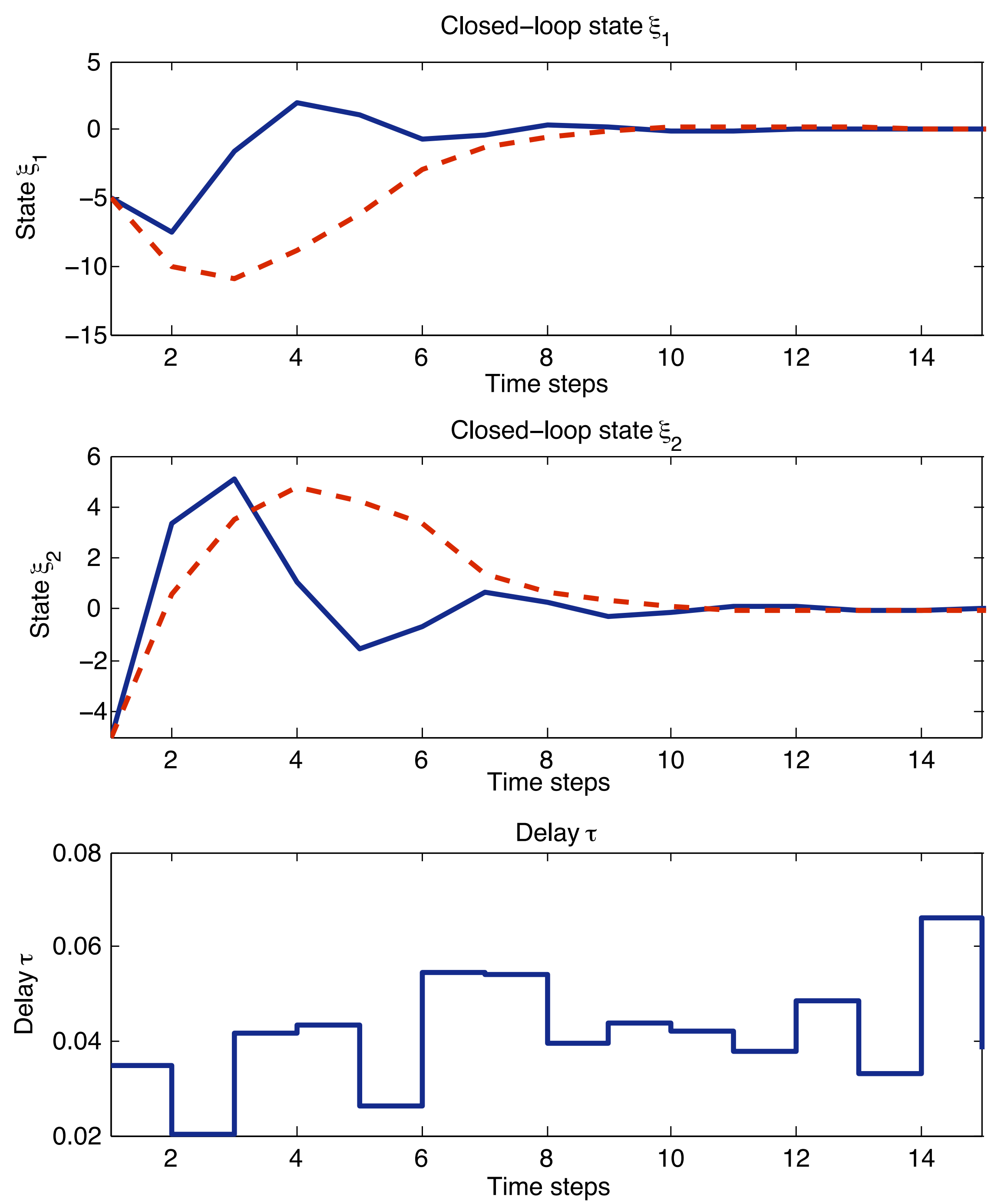
- **Delays:** truncated normal distribution with $\tau \in [0.02,\ 0.1]$

- **Approximation:**
  - pdf partitioned in **8 regions** for performance optimization
  - pdf partitioned in **4 partitions** for stability conditions



SMPC of networked systems - An example

- Comparison with a constant state-feedback controller that provides robust stability (no exploitation of stochastic disturbance model)



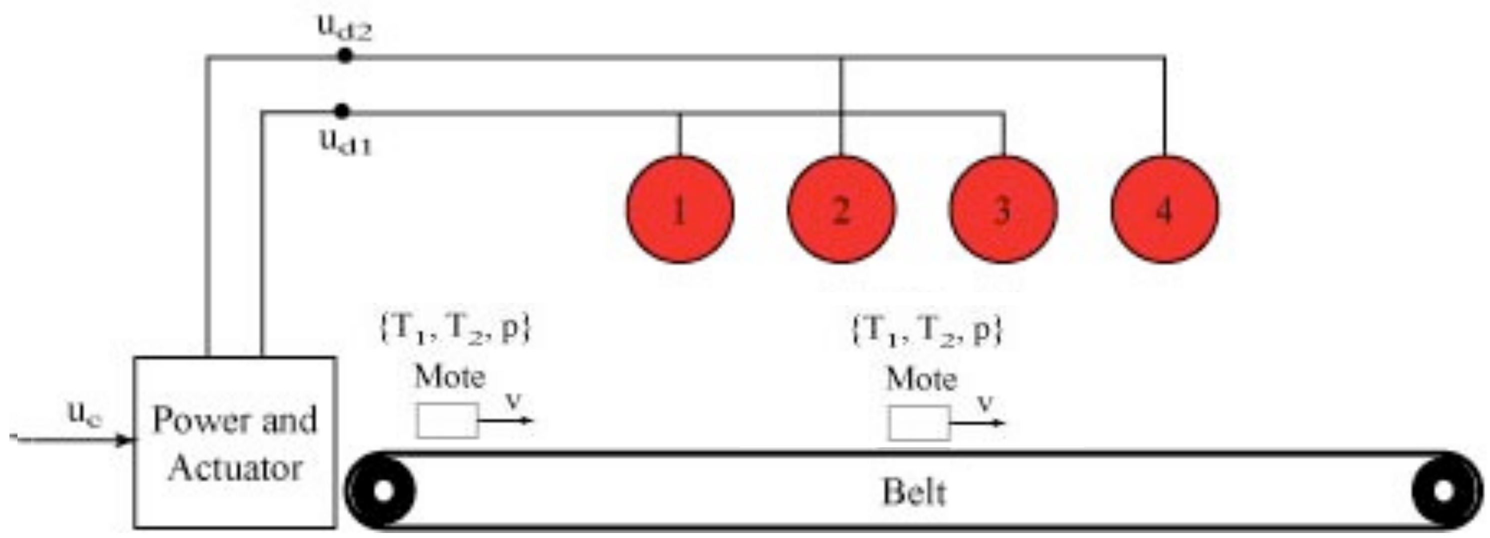
Controller	average performance	standard deviation
Robust state-feedback	884.34	382.19
Stochastic MPC	678.01	134.74

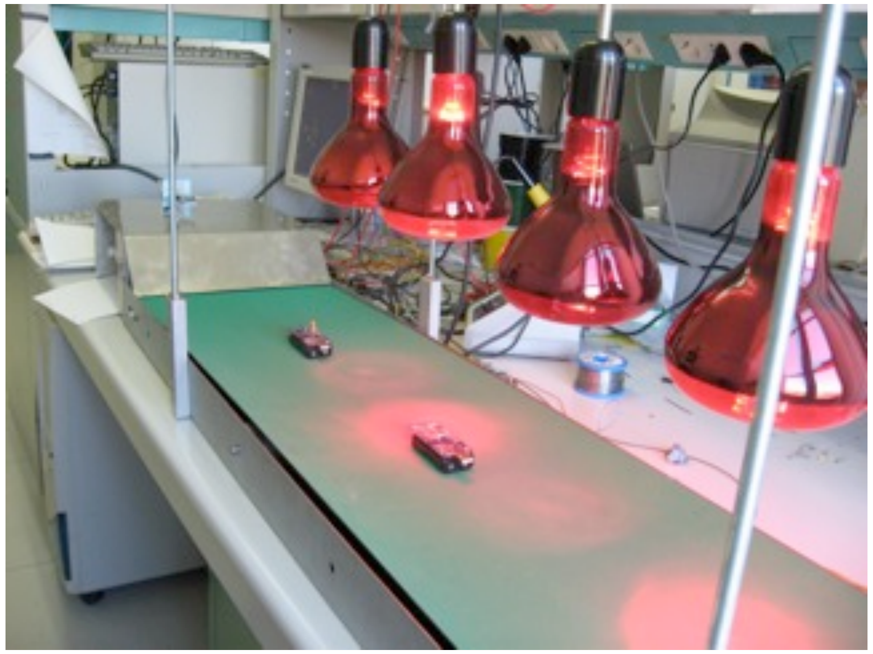
Computation time per step:

- average: 29 ms
- maximum: 41 ms

Demo Application in Wireless Automation

(Bemporad, Di Cairano, Henriksson, Johansson, IJRNC, 2010)





- Telos motes provide wireless temperature feedback in MATLAB/Simulink
- xPC-Target link used for wired communication
- MPC algorithm adjusts belt speed and turns lamps on/off (HybTBX+CPLEX)
- Objective: track **position** and **temperature** references while enforcing safety constraints

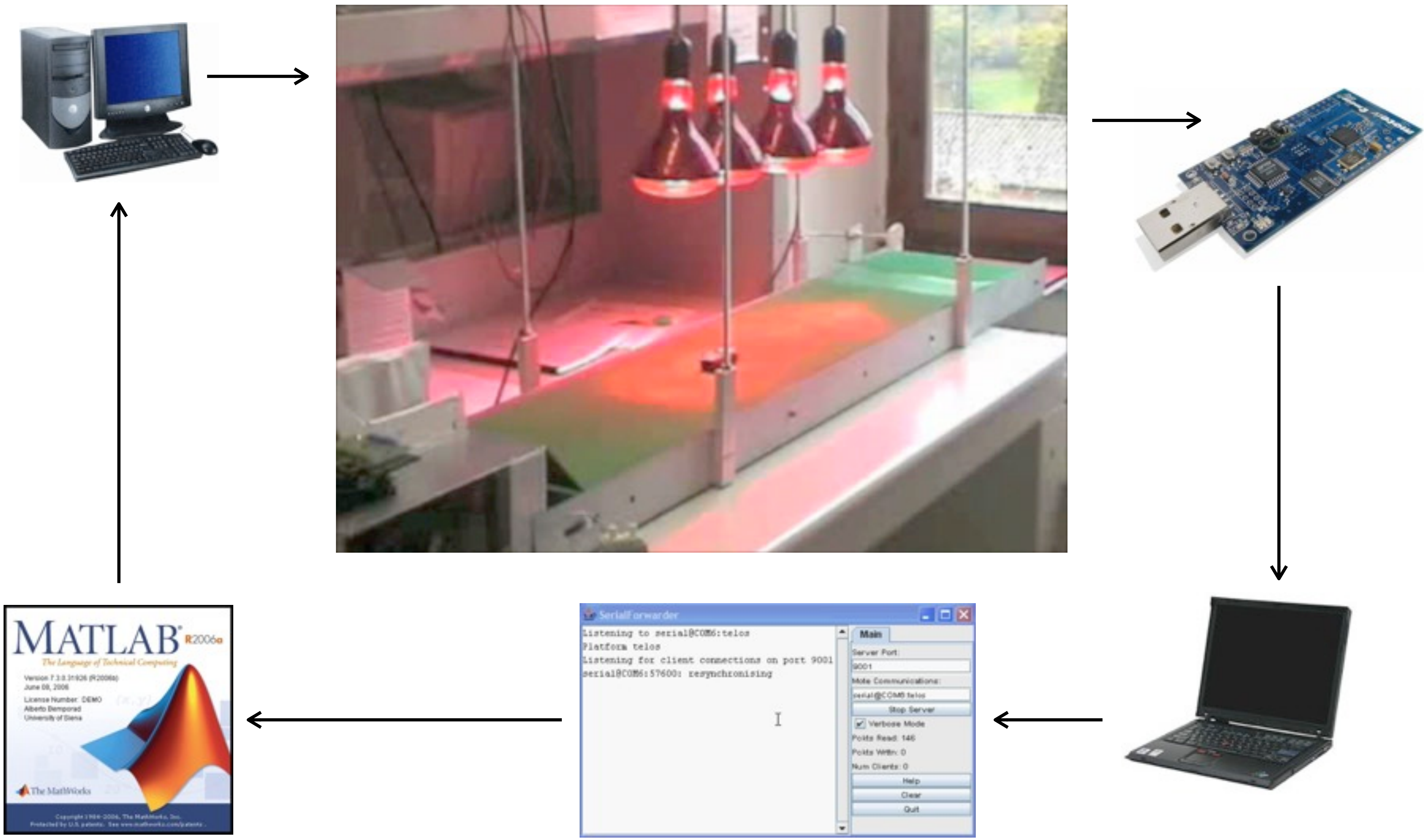
Motivation

Laminating plastics

Shrink wrapping

Ink drying in printing

Experimental results



Hybrid MPC problem with 2 binary inputs (lamps), 1 continuous input (speed), piecewise linear state function (heating), 2 outputs (temperature and position)

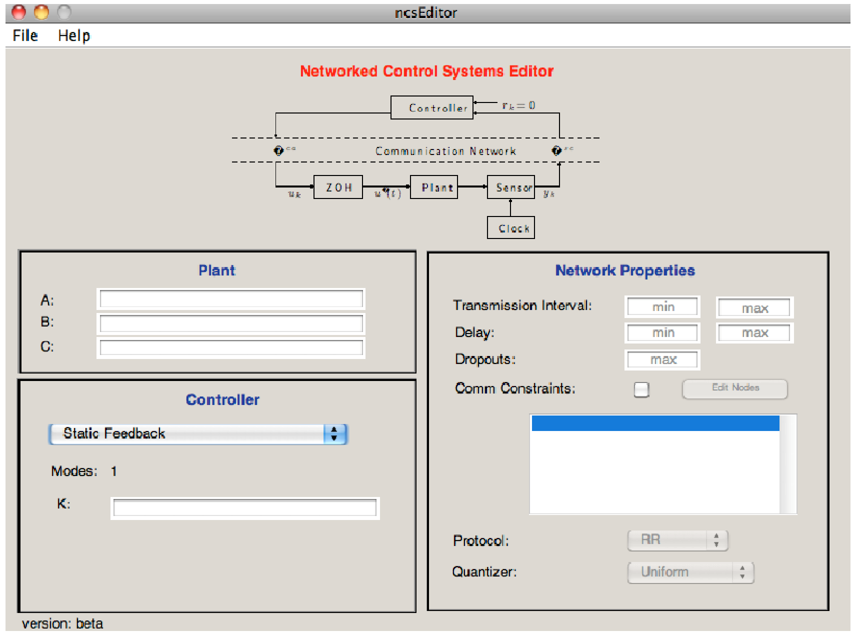
Conclusions and open research issues

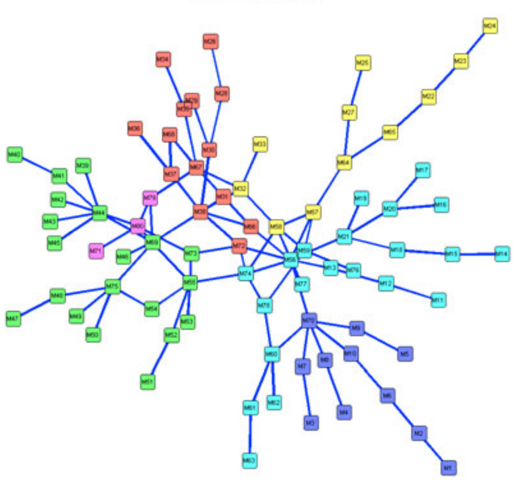
dynamical model	**+**	optimization algorithm	**=**	MPC

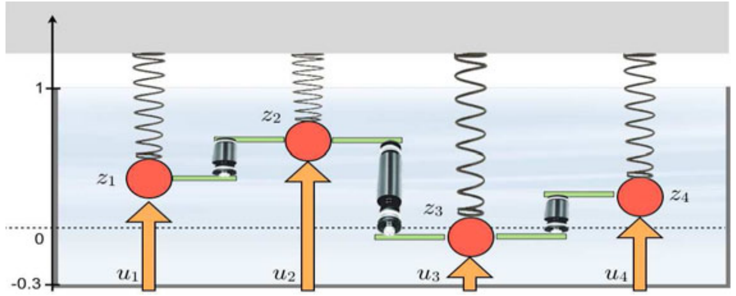
- **MPC** is a very rich control methodology, with many possible variants to handle different issues in control of networked systems, such as:
  - energy consumption of wireless sensors
  - large number of variables (states/inputs/outputs)
  - stochastic effects of network
- **Decentralized and distributed MPC of networked systems**:
  - some contributions exist, but theory not yet mature
  - a lot to gain from distributed optimization theory
- **Stochastic MPC of networked systems**:
  - some theoretical contributions exists
  - many ways to setup the stochastic MPC problem (=large space for new theories)

WIDE Toolbox for MATLAB

- **Networked control systems:** modeling, stability analysis, linear control synthesis
- **Model management** of large-scale systems (model reduction, create submodels, …)
- **Data acquisition** from physical WSN (Telos motes, E-Senza’s nodes)
- **WSN simulation** (generate TrueTime code)
- **MPC control:** decentralized, hierarchical, network-aware







- Available for public download on September 1st 2011
<u>`http://ist-wide.dii.unisi.it`</u>

TU/e

IMT

Honeywell