Event based control: A way to reduce reconfigurations in autonomic computing ?

N. Marchand

gipsa, Control Systems Department, Grenoble, FRANCE

Outline

Motivation

to put control theory in computer science

- Dealing with the **dynamics**: **time is crucial**
- Mathematical tools to "control" a system
- By "control", we mean being able to
  - define a control objective
  - define control actions accordingly
  - **guarantee** performances of the controlled system
    - despite errors
    - despite perturbations
    - ➳ Facing everything that is unknown
    - ➳ Guarantee stability
  - Many other area of control theory are relevant to computer science
    - Fault tolerant control, fault detection, supervision, etc.
- Nowadays control theory is everywhere...
  - automotive, robotics, energy (grids, production, etc.), microelectronics (DVFS), etc.
- ...except maybe in computer science

Challenging difficulties

Let's start with the hardest things

- Languages difficulties

Words	Computer science	Control theory
Autonomic or Autonomous	Controlled	Uncontrolled
Time response	Queuing + processing time	time needed to reach x% of the final value
Parameter	variables you can change	constants
Cloud	Set of interconnected computers	Look at Lund's sky
Control	Parametrization	$\frac{dx}{dt} = f(x, u)$
...	...	...

- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

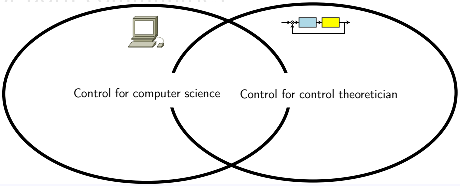
Challenging difficulties

Let's start with the hardest things

- Languages difficulties

Words	Computer science	Control theory
Autonomic or Autonomous	Controlled	Uncontrolled
Time response	Queuing + processing time	time needed to reach $x\%$ of the final value
Parameter	variables you can change	constants
Cloud	Set of interconnected computers	Look at Lund's sky
Control	Parametrization	$\frac{dx}{dt} = f(x, u)$
...	...	...

- Interest of both communities
- No phys
- "Let's d



Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
  - Computer science community wants control already operative
  - Control community don't care about computer sciences
    - Except in Lund !
    - IFAC technical commitee on computer for control but none on control for computers (one on on mining..., see IFAC website)
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
  - Computer science community wants control already operative
  - Control community don't care about computer sciences
    - Except in Lund !
    - IFAC technical commitee on computer for control but none on control for computers (one on on mining..., see IFAC website)
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

Challenging difficulties

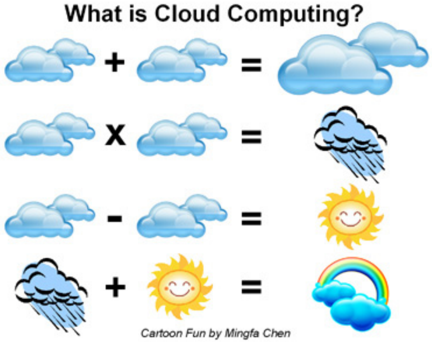
Let's start with the hardest things

- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
  - Building models is critical and unusual
    - How do i put the system in the control theory normal form $\frac{dx}{dt}x = f(x,u)$?
    - Frequency, poles, etc. are sometimes clear
    - Control, outputs, sensors, etc. can disappear with a system update
    - Evolution of a system can be discontinuous (robustness issue)
    - No "tiredness", only crashes
  - Model must capture main behavior BUT
    - if too precise ☞ too complex
    - if too complex ☞ inefficient for control (unrobust)
    - Model for control is not classical modeling
  - Requires much more interaction than usual sciences
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud **computing**" (Sara Bouchenak from LIG-lab)



Challenging difficulties

Let's start with the hardest things

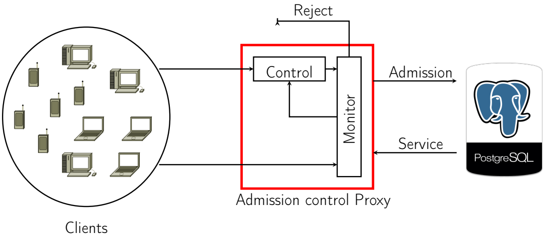
- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud **control**" (Sara Bouchenak from LIG-lab)

Menu

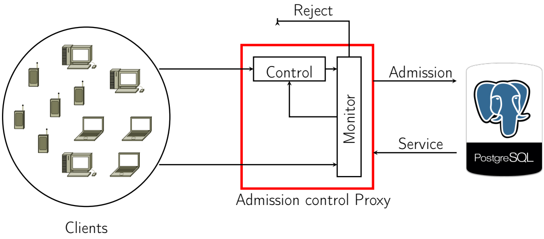
- *Starter*: An short example of how bad computer system can be for control theory:
  - Admission control for PostgreSQL database server
- *Main dish*:
  - Cloud control needs to
    - react to spikes (high frequency)
    - reconfigure as less as possible (low frequency)
    - Antinomic !
  - Focus on Event-Based control
    - More on event based PID
    - Short presentation of extensions
    - Assure SLA compliance in Hadoop Mapreduce
- *Dessert*: What need to be more efficient

Hope it will be not too indigestible !

A nonlinear system example



A nonlinear system example



- Model gives when saturated:

$$\left\{\begin{array}{rcl}\frac{dN}{dt} & = & (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} & = & -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} & = & -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right]\end{array}\right.$$

- N: number of concurrent request on the server
- α: abandon rate
- T_o: Throughput of served requests
- T_i: Throughput of incoming requests
- MPL: Multi Processing Level, it is the control variable
- a, b, c and Δ: parameters

A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} & = & (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} & = & -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} & = & -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model

- few variables/parameters
- easily identifiable
- fits well

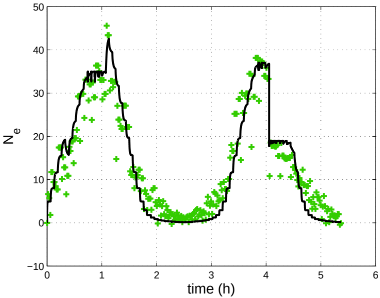
A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} = (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} = -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}\left(1-\frac{T_o}{T_i}\right)\right] \\ \frac{dT_o}{dt} = -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model

- few variables/parameters
- easily identifiable
- fits well



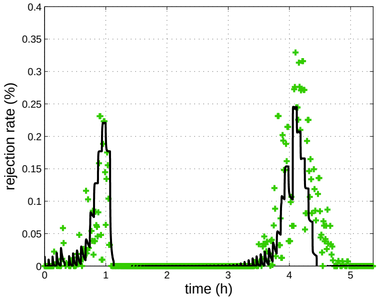
A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} & = & (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} & = & -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} & = & -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model

- few variables/parameters
- easily identifiable
- fits well



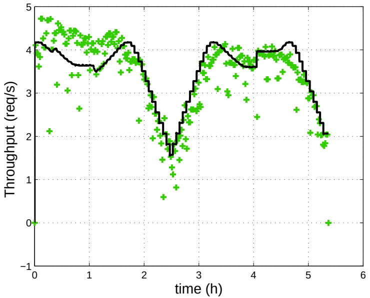
A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} = (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} = -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} = -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model

- few variables/parameters
- easily identifiable
- fits well



A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} & = & (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} & = & -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} & = & -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model

- few variables/parameters
- easily identifiable
- fits well

✗ Not an easy model

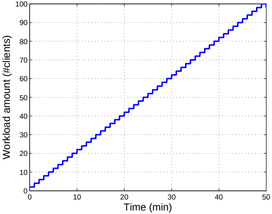
- Model is highly nonlinear
- Not in the standard form for control theory
- Control is the level saturation of an exogenous input

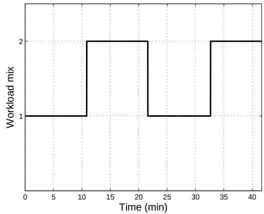
A nonlinear system example

- Controlled thanks nonlinear control theory (Lyapunov)
- Stability is guaranteed for any value of the parameters
  - No danger of miss-identification

A nonlinear system example

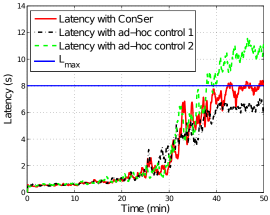
- Controlled thanks nonlinear control theory (Lyapunov)
- Stability is guaranteed for any value of the parameters
    - No danger of miss-identification

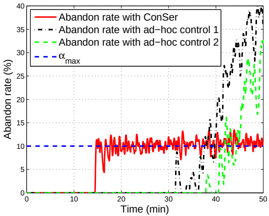




Latency control

Rejection rate control



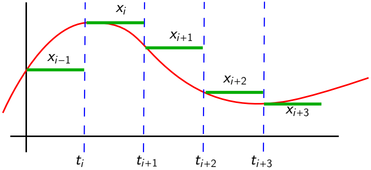


Menu

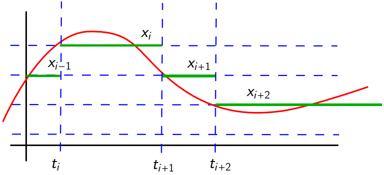
- *Main dish*:
  - Cloud control needs to
    - react to spikes (high frequency)
    - reconfigure as less as possible (low frequency)
    - Antinomic !
  - Focus on Event-Based control
    - More on event based PID
    - Short presentation of extensions
    - Assure SLA compliance in Hadoop Mapreduce

Event-based sampling vs. periodic sampling

- Periodic sampling
  - Sampling periodically on time
  - Analogical to Riemann's integral
  - Well known theory (Shannon, etc.)

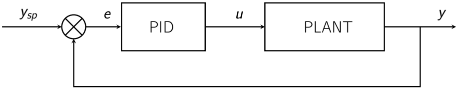


- Event-based sampling
  - Sampling on level's
  - At first glance close to Lebesgues integral
  - Different extension :
    - Outside event (event-triggered)
    - State/output dependent sampling (self-triggered)
  - Should reduce transmission/computation
  - Few theory



PID controller

Classical work



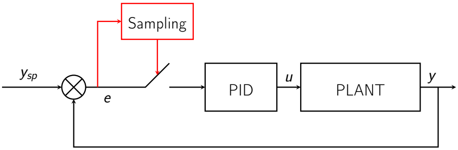
- Classical PID
  - In the frequency domain:

$$U(s) = K\left(E(s) + \frac{1}{T_i s}E(s) + T_d s E(s)\right)$$

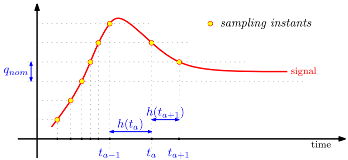
  - Discrete time version (h_{nom}: sampling period, N tunes the filter):

$$\begin{aligned}
u_p(t_k) &= Ke(t_k) \\
u_i(t_{k+1}) &= u_i(t_k) + K_i h_{nom} e(t_k) \\
u_d(t_k) &= \frac{T_d}{T_d + N h_{nom}} u_d(t_{k-1}) + \frac{K T_d N}{T_d + N h_{nom}}(e(t_k) - e(t_{k-1})) \\
u &= u_p + u_i + u_d
\end{aligned}$$

PID controller



- Idea:
  - do not update the control if y is close to y_{sp}, typically if $||e(t_a) - e(t_{a-1})|| \leq q_{nom}$
- No need to respect Shannon
- Bad behavior of the integral part
- q_{nom} linked to precision and noise



What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling $\Rightarrow$ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + (t_{i+1} - t_i) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{2}}$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling ⇒ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + \left(t_{i+1} - t_i\right) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x,\ e(x) = 0$ when $|x| = \exp(-\kappa),\ \kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{2}},\ e(x) = 0$ when $|x| = \frac{1}{\kappa},\ \kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling ⇒ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + (t_{i+1} - t_i) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{2}}$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

What can happen (2/3) ?

3. $k(x) = -x,\ e(x) = 0$ when $|x| = \frac{1}{\kappa},\ \kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{i+1}$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = 0$ when $\lim t_i = \infty$
   - Infinitely fast sampling at infinity
4. $k(x) = -x^3,\ e(x) = 0$ when $|x| = \exp(-\kappa),\ \kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \exp(2i) \cdot [1 - \exp(-1)]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$
   - Infinitely slow sampling at infinity

What can happen (2/3) ?

3. $k(x) = -x,\ e(x) = 0$ when $|x| = \frac{1}{\kappa},\ \kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{i+1}$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = 0$ when $\lim t_i = \infty$
   - Infinitely fast sampling at infinity
4. $k(x) = -x^3,\ e(x) = 0$ when $|x| = \exp(-\kappa),\ \kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \exp(2i) \cdot [1 - \exp(-1)]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$
   - Infinitely slow sampling at infinity

What can happen (3/3) ?

- **Unstable system**: $\frac{dx}{dt} = (x+u)^3$
- Solution is: $x_{i+1} = \frac{x_i+u}{\sqrt{1-2(t_{i+1}-t_i)\cdot(x_i+u)^2}} - u$

5. $k(x) = -2x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$ and initial condition $x_0 = 1$
   - $t_{i+1} - t_i = \frac{\exp(2i)}{2} \cdot \left[1 - \frac{1}{(2-\exp(-1))^2}\right]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0,\infty[$

What can happen (3/3) ?

- **Unstable system**: $\frac{dx}{dt} = (x+u)^3$
- Solution is: $x_{i+1} = \frac{x_i+u}{\sqrt{1-2(t_{i+1}-t_i)\cdot(x_i+u)^2}} - u$

5. $k(x) = -2x,\ e(x) = 0$ when $|x| = \exp(-\kappa),\ \kappa \in \mathbb{Z}$ and initial condition $x_0 = 1$
   - $t_{i+1} - t_i = \frac{\exp(2i)}{2} \cdot \left[1 - \frac{1}{(2-\exp(-1))^2}\right]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0,\infty[$

PID controller

- We focus on the integral part
  - What happens one waits too long before updating the control ?
  - The integral part grows because $h(t_a)$ grows:

$$u_i(t_a) = u_i(t_{a-1}) + K_i \underbrace{h(t_a)}_{\text{big}} \underbrace{e(t_a)}_{\text{small}}$$

  - Strong overshoot when the control is updated (similar to saturated PID without antiwindup)
- **Solution**: replace the product $h \cdot e$ by a bounded function he:

$$u_i(t_a) = u_i(t_{a-1}) + K_i \underbrace{he(t_a)}_{\text{limited}}$$

  - Saturation, Exponential forgetting factor, Hybrid, etc.

PID controller

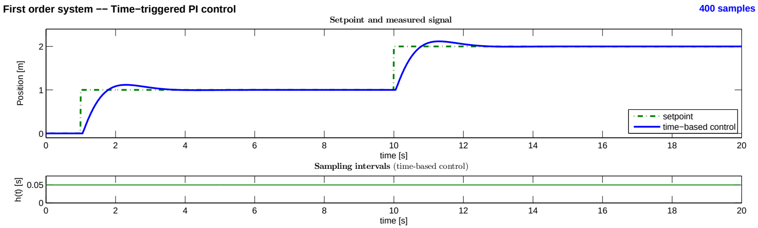
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Periodic sampling



PID controller

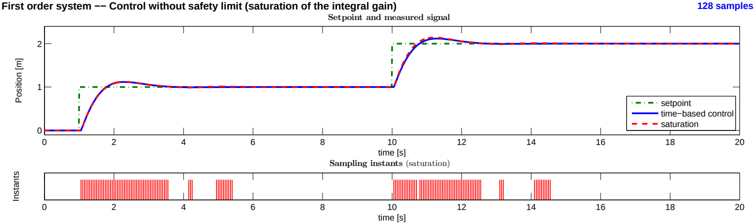
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Evend based PID with saturated $h \cdot e$



PID controller

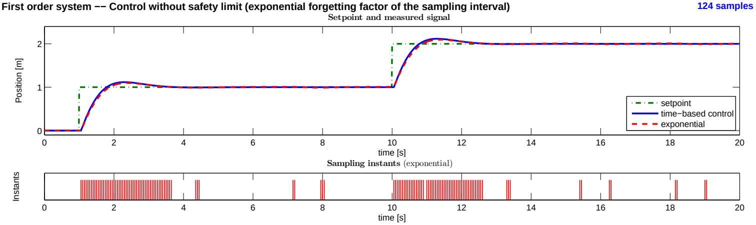
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Evend based PID with exponential forgetting



PID controller

Simulation result

LCCC workshop on Cloud Control

N. Marchand

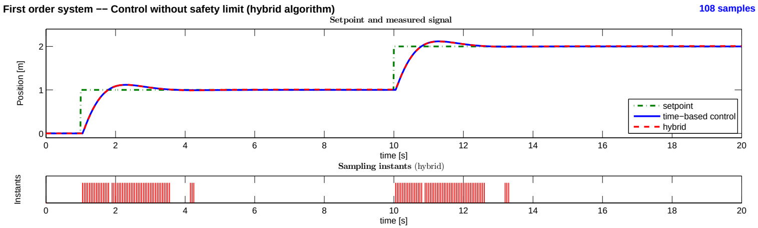
Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G=1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

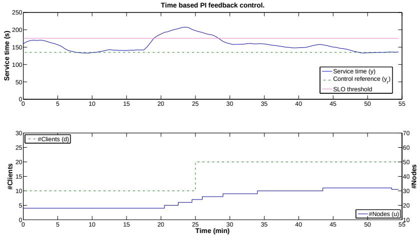
Evend based PID with hybrid $h \cdot e$

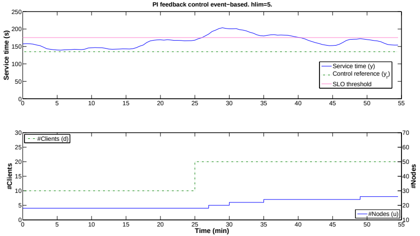


MapReduce control

Simple PI controller

- Time based (8 updates) vs. Event based (4 updates)

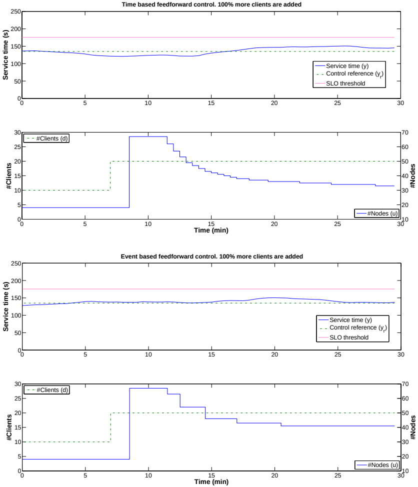




MapReduce control

Simple PI controller with feedforward

- Time based (18 updates) vs. Event based (6 updates)



More food for people in control theory

- Event-based control is really recent
- Now exist :
  - Almost basic linear control where carried in an event-based framework (PID, LQR, etc.)
  - Sontag's general formula has been extended
  - A lot of strategies based on Lyapunov theory exist
  - Many practical implementations even on noisy and unstable systems
  - Early results are appearing for time-delayed systems
- Remain to clarify
  - Real number of control updates
  - All what it brings (in good and bad) is not clear
  - Frequency analysis is less convenient

Conclusion

- People always adopt control theory ...
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.

What need to be improved

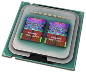
- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side



- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

What need to be improved

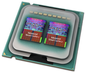
- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together

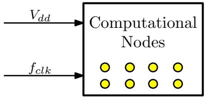


- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

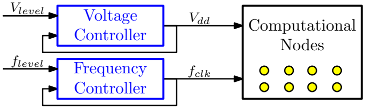
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

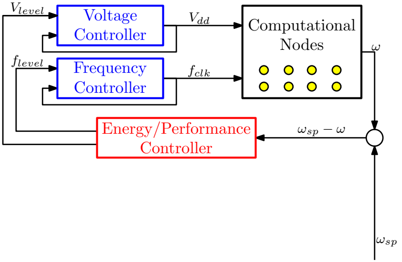
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

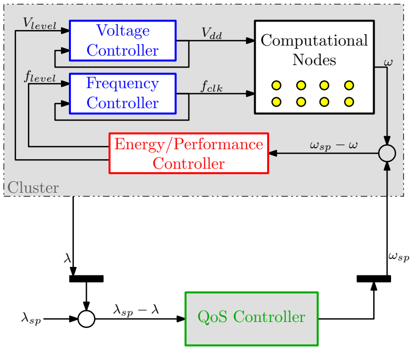
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

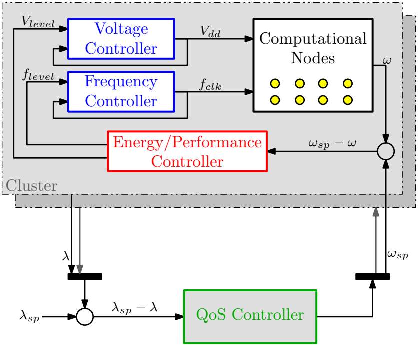
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

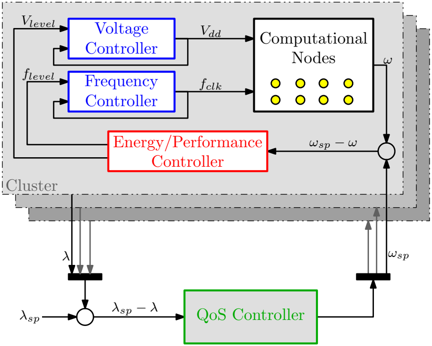
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)
Also the approach used in smart grids

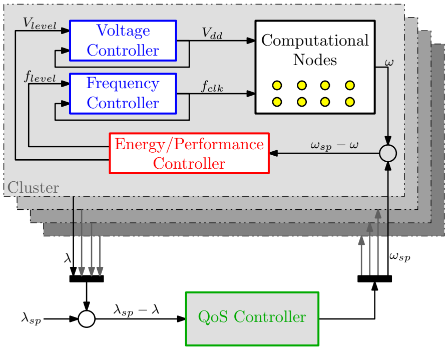
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)
Also the approach used in smart grids

1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Grenoble Workshop on Autonomic Computing and Control

- Date: 27 may 2014
- Location: Grenoble
- Organisation: Eric Rutten, INRIA and Stéphane Mocanu, Gipsa-lab
- Confirmed speakers:
  - Karl-Erik ARZEN (Lund, Sweden)
  - Alberto LEVA (Milano, Italy)
  - Ada DIACONESCU (Telecom Paris-Tech, France)
  - Suzanne LESECQ (CEA LETI)
  - Didier DONSEZ (LIG)
  - Bogdan ROBU (GIPSA)
  - Eric RUTTEN (INRIA)

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Confirmed lecturers:
  - Didier HENRION
  - Andrew TEEL
  - Laurent PRALY
  - Mirko FIACCHINI
  - Luca ZACCARIAN

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Con

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Co