ORTEGA: An Efficient and Flexible Online Fault Tolerance Architecture for Real-Time Control Systems

Xue Liu, Qixin Wang, Sathish Gopalakrishnan, Wenbo He, Lui Sha, Hui Ding, Kihwal Lee

Outline

- Motivation and related work
- ORTEGA goals
- ORTEGA architecture
- Details of ORTEGA designs
- Implementation and evaluation
- Demo

Motivations

- Cyber-Physical Systems
  - Real-world systems involves not only computer science, but knowledge related to various disciplines.
  - Not only the computer system becomes more complex, the complexity of integrated system (i.e. the cyber-physical system) grows even faster.
  - Major challenge: how to let engineers of drastically different backgrounds collaborate with each other?

Motivations

- Control Systems
  - Conventional analog control systems
  - Digital control systems
- Computer Systems
  - Real-time scheduling
  - Fault tolerance
  - Reliable/online software upgrade
- We need to design a framework so that computer engineers and control engineers can easily collaborate and integrate their knowledge

$$\dot{x} = Ax + Bu$$

$$u = -Kx$$

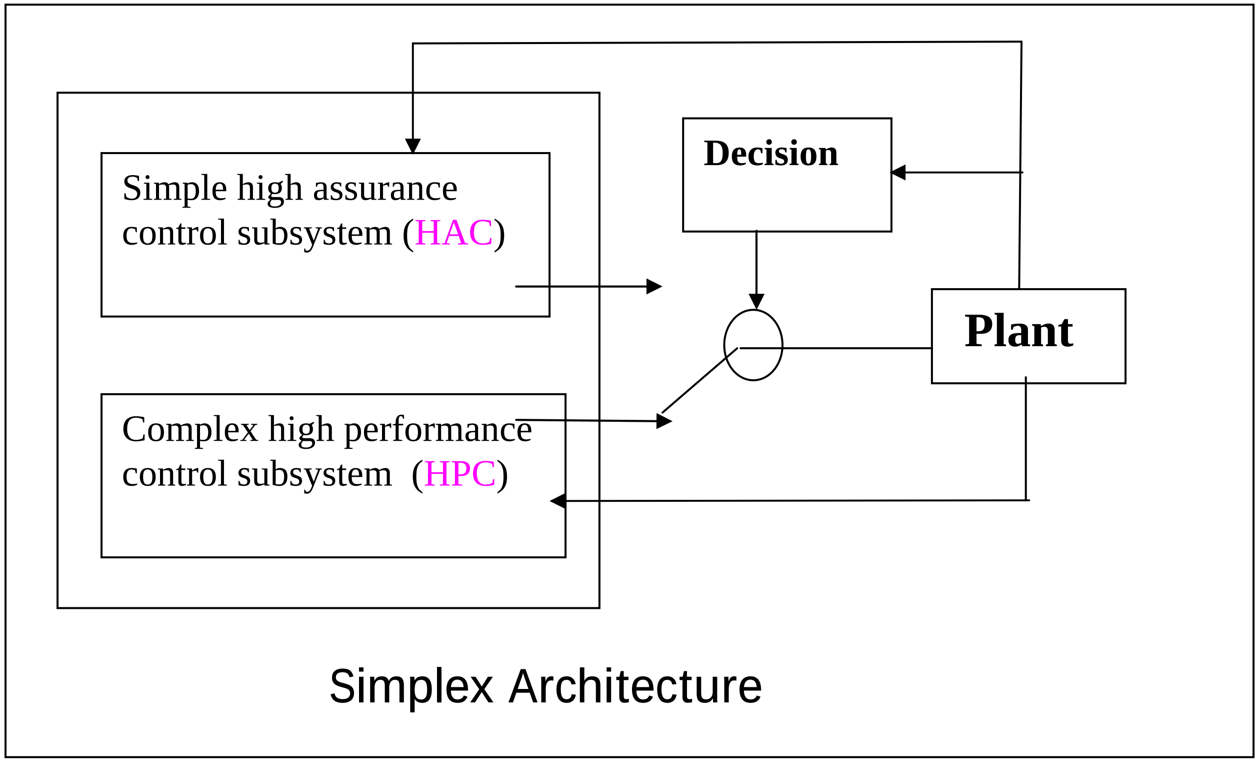
$$x(kh+h) = e^{Ah}x(kh) + \left(\int_0^h e^{As}ds\right)Bu(kh)$$

$$u(kh) = -Kx(kh)$$

Related work: Simplex architecture

- Demand:
  - Low cost development of upgraded control systems for mission critical control applications
    - instead of multi-versioning, just develop one version
    - Focus on the control theories
  - Runtime upgrade/testing of the single version buggy new system.
- Applications:
  - Aircraft control (F-16, Seto et. al, 2000)
  - Submarine control (NSSN, new attack submarine program at US navy)

Simplex for real-time control


Simple high assurance control subsystem (HAC)
Complex high performance control subsystem (HPC)
Decision
Plant

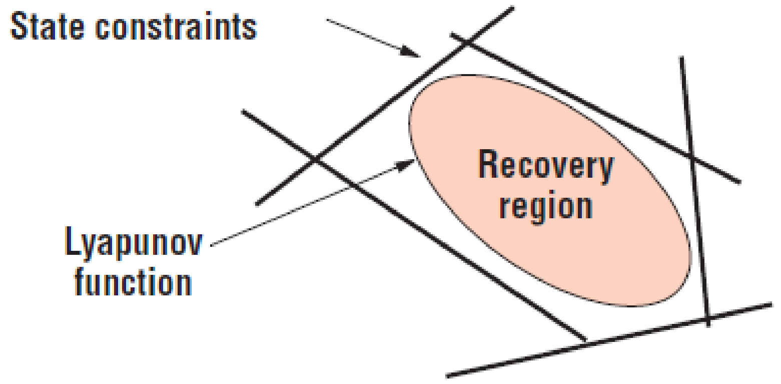
Simplex Architecture

Simplex for real-time control

Given LTI control system:

$$\dot{x} = \overline{A}x + Bu$$

$$= \overline{A}x - BKx = Ax$$

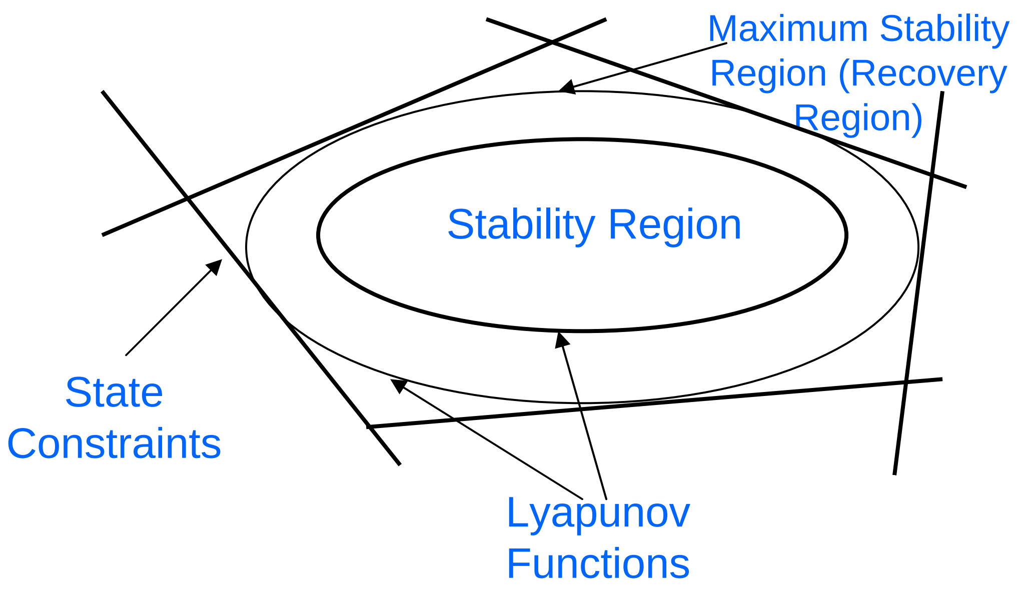


The above LTI control system is stable iff there exists a P>0, such that the Lyapunov function

$$x^T(A^T P + PA)x < 0$$

The solution ellipsoid is maximized by minimizing $\log\det P^{-1}$

Simplex for real-time control



We can choose smaller solution ellipsoid (i.e. $x^T P x < x^T P^{max} x$) to leave margins to guard against model/actuator/measurement errors.

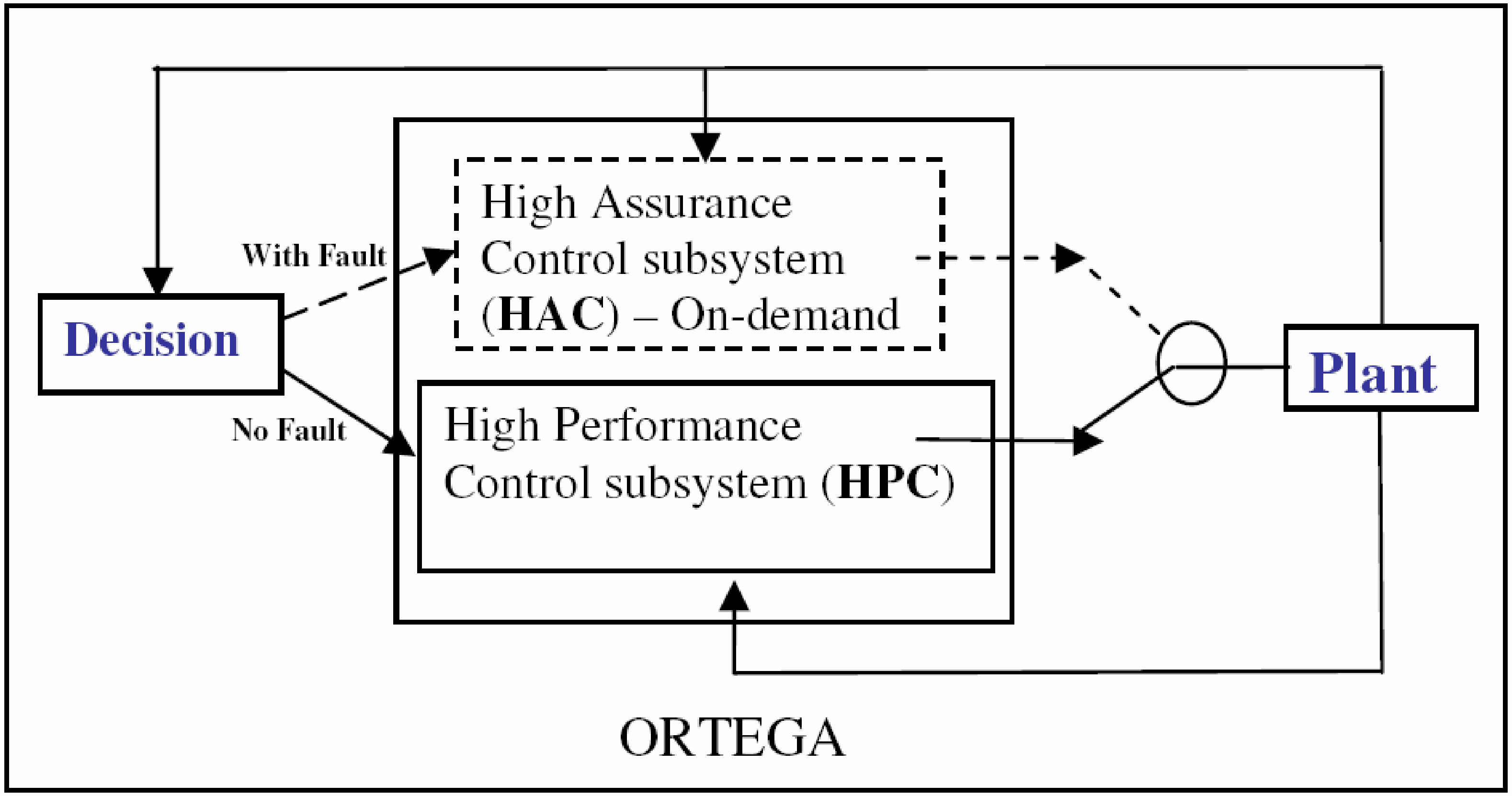
Drawbacks of Simplex

- P1: Lack of Efficiency
  - Analytically redundant high assurance controller (HAC) runs in parallel with complex controller (HPC)
    - Lowers system performance, increase operating costs
    - Limits the application of Simplex in only safety-critical domains
- P2: Lack of Flexibility
  - Enforces the same execution period on HAC and HPC
    - In practice, different controllers may use different periods for different performance considerations
    - For example: fast HAC recovery

Design goals of ORTEGA

- On-demand Real-TimE GuArd (ORTEGA)
  - A new efficient fault tolerance software architecture designed for real-time control systems
- More efficient resource usage (P1)
  - Through on-demand real-time recovery
- Flexible design (P2)
  - Allows HAC and HPC to run at different rates
  - Through new design and schedulability analysis
- Applicable to a wider range of real-time control systems

ORTEGA Architecture



With Fault
Decision
No Fault
High Assurance Control subsystem (HAC) – On-demand
High Performance Control subsystem (HPC)
Plant
ORTEGA

On-demand execution of HAC

- At any time, only one of the HAC or HPC is running to control the plant
- Decision module (DM) uses a mutex semaphore to control which of the HAC and HPC is running
  - When the HPC is running well, the HAC blocks on the semaphore;
  - Only when a fault is detected in the HPC, the DM releases the semaphore to allow HAC to take over
- Decision logic is based on stability regions
  - Determined through Linear Matrix Inequality theory
  - Details later

CPU savings of ORTEGA

HPC's timing parameters: $\{C^p, T^p\}$; HAC's timing parameters: $\{C^a, T^a\}$;

Pr: the percentage of time for recovery (HAC) during a total time of *T*

- Total CPU resource usage under Simplex

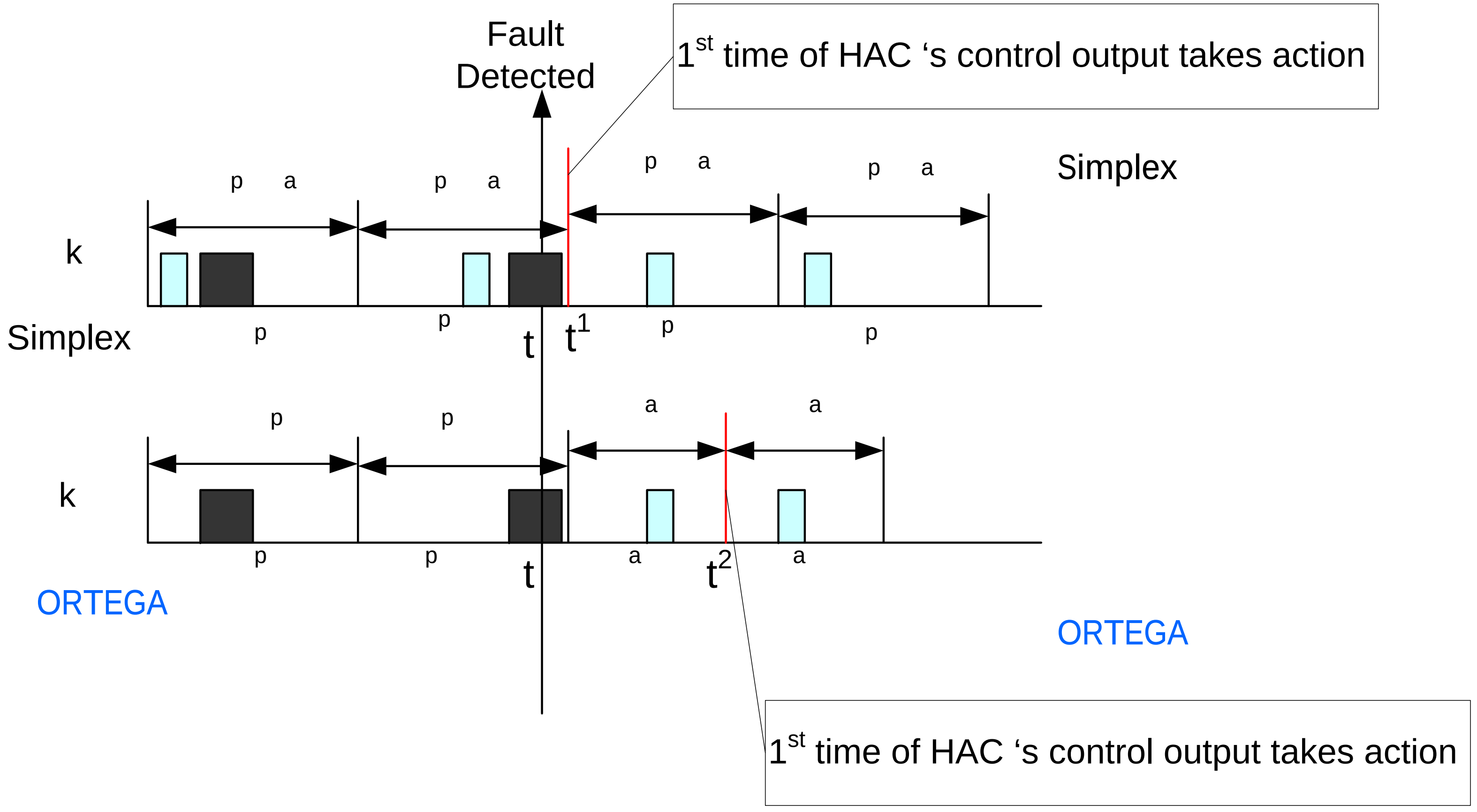
$$R_{Simplex} = (1-P_r)\cdot\left(C^a\cdot\left\lceil\frac{T}{T^a}\right\rceil + C^p\cdot\left\lceil\frac{T}{T^p}\right\rceil\right) + P_r\cdot C^a\cdot\left\lceil\frac{T}{T^a}\right\rceil$$

- Total CPU resource usage under ORTEGA

$$R_{ORTEGA} = (1-P_r)\cdot C^p\cdot\left\lceil\frac{T}{T^p}\right\rceil + P_r\cdot C^a\cdot\left\lceil\frac{T}{T^a}\right\rceil$$

- CPU resource usage savings: $(1-P_r)\cdot C^a\cdot\left\lceil\frac{T}{T^a}\right\rceil$

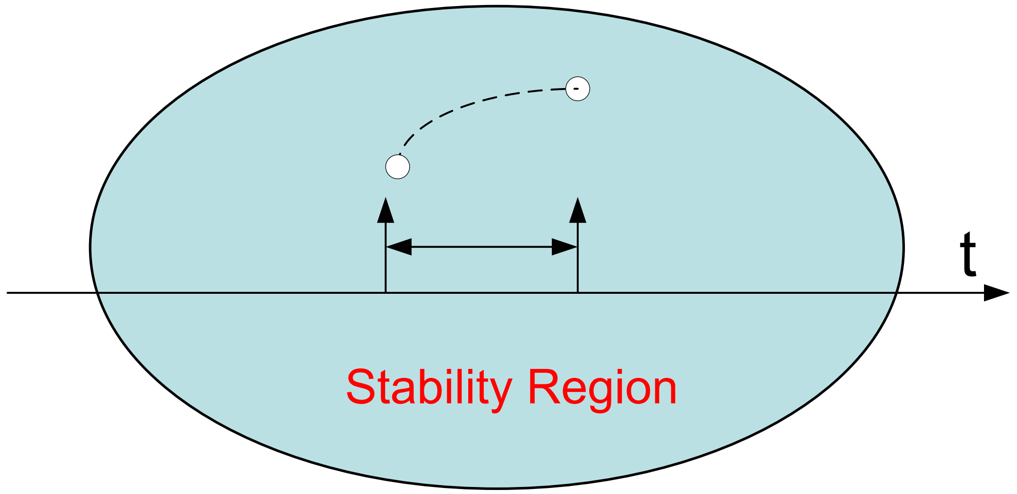
No Free Lunch: An extra period of delay



Fault Detected
1st time of HAC 's control output takes action
Simplex
p a
p a
p a
p a
k
Simplex
p
p
t
t1
p
p
p
p
a
a
k
p
p
t
a
t2
a
ORTEGA
ORTEGA
1st time of HAC 's control output takes action

up to T^a incurred due to the on-demand execution of HAC

Handle the extra delay by state projections

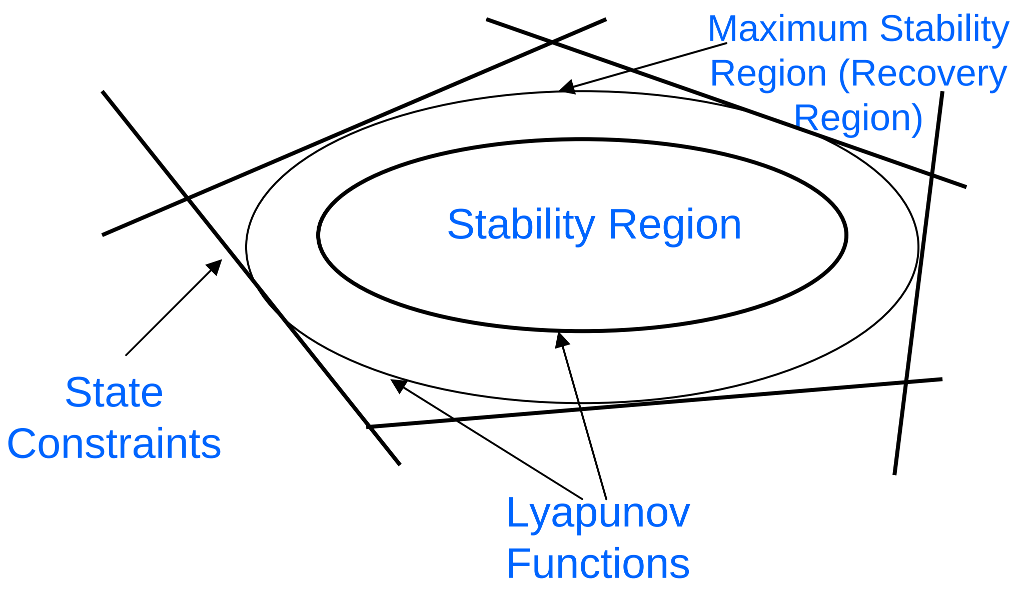


Resource usage reduction v.s. extra delay :

(1) Extra delay causes disturbances when fault occurs (infrequent)

(2) But the gain in resource usage is large.

Recovery region design



- The decision module uses recovery region to determine when to switch to HAC
- Recovery region is defined as the maximum region in which the HAC can make the plant stable

Determine recovery region (1)

Digital controllers:

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t),$$

$$\Downarrow$$

$$x(k+1) = F(h)x(k) + G(h)u(k),$$

$$u(k) = -Kx(k)$$

$$\Downarrow$$

$$x(k+1) = \bar{F}x(k) \qquad (*) \qquad (\bar{F} = F - GK)$$

State constraints:

$$\alpha_m^T x \le 1, \quad m = 1, \cdots, q. \quad (1)$$

Stability region:

The discrete LTI control system is stable iff there exists a P>0, such that $\bar{F}^T P \bar{F} - P < 0$

Determine recovery region (1)

Digital controllers:

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t),$$

$$\Downarrow$$

$$x(k+1) = F(h)x(k) + G(h)u(k),$$

$$u(k) = -Kx(k)$$

$$\Downarrow$$

$$x(k+1) = \bar{F}x(k) \qquad (*) \qquad (\bar{F} = F - GK)$$

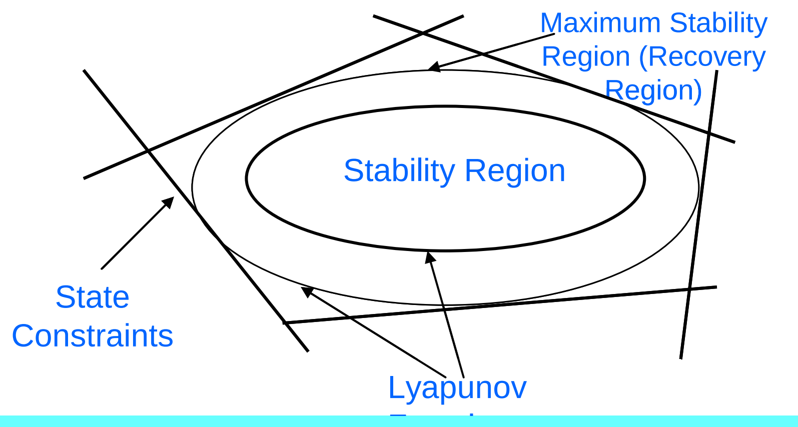
State constraints:

$$\alpha_m^T x \le 1, \quad m = 1, \cdots, q. \quad (1)$$

Stability region:

Stability region of the system with respect to *P* is defined as $\{x \mid x^T P x < 1\}.$

Determine recovery region (2)



Theorem: Determine the maximum stability region of digital implemented closed loop system with constraints (1) can be transformed to the following MAXDET (LMI) problem.

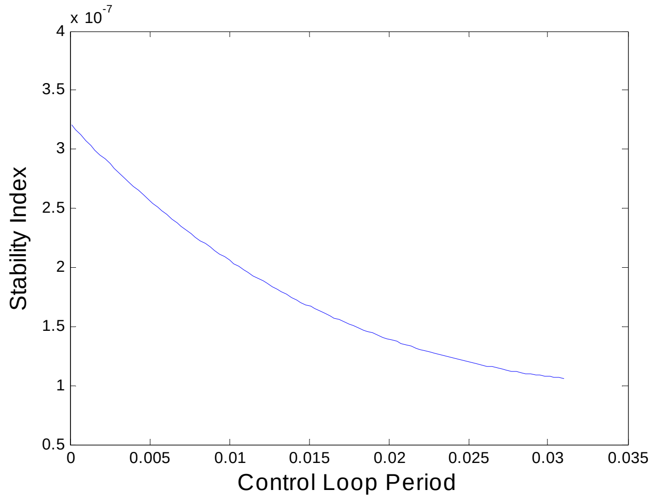
$$\begin{aligned} & Maximize \quad \underbrace{\text{logdet}\, P^{-1}}_{\text{Area of recovery region}} \\ & s.t.: \quad \underbrace{P > 0, \quad \bar{F}^T P \bar{F} - P < 0,}_{\text{Stability}} \\ & \qquad \underbrace{\alpha_m^T P^{-1} \alpha_m \le 1, \qquad m = 1, \cdots, q.}_{\text{State constraints}} \end{aligned}$$

Recovery region v.s. control loop period

Stability Index A(T): Area of the maximum stability region

- It is a function of the control loop period *T*. The smaller the controller loop period, the larger the maximum stability region.

Example: an inverted pendulum



System model

$$\dot{x} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2.7528 & -10.9526 & 0.0043 \\ 0 & 28.5812 & 24.9179 & -0.0441 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 1.9432 \\ -4.4385 \end{pmatrix} u$$

Controller

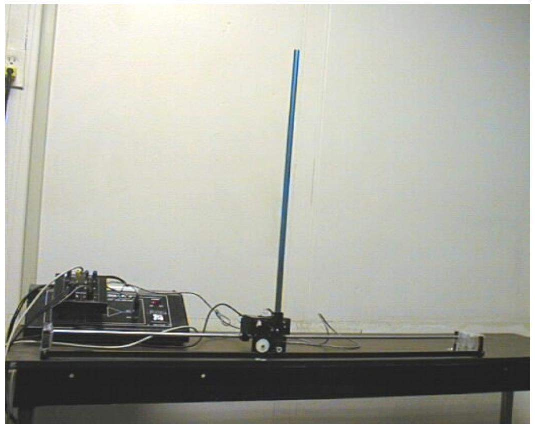
$$u(k) = -[5.7807, 42.2087, 14.0953, 8.6016]x(k)$$

The smaller the period, the larger the recovery region.

ORTEGA allows larger recovery region (more flexible)

Implementation and evaluation

- Inverted pendulum from Quanser
- CPU: Pentium II 350MHz
- OS: Linux kernel 2.4.18-3 with RMS
- HAC: field tested state feedback controller



Evaluation of CPU savings

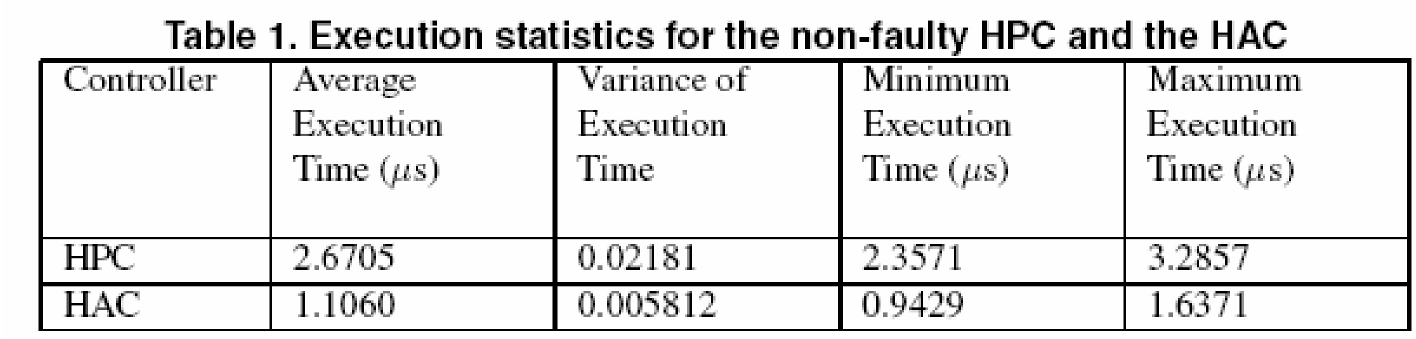


Table 1. Execution statistics for the non-faulty HPC and the HAC

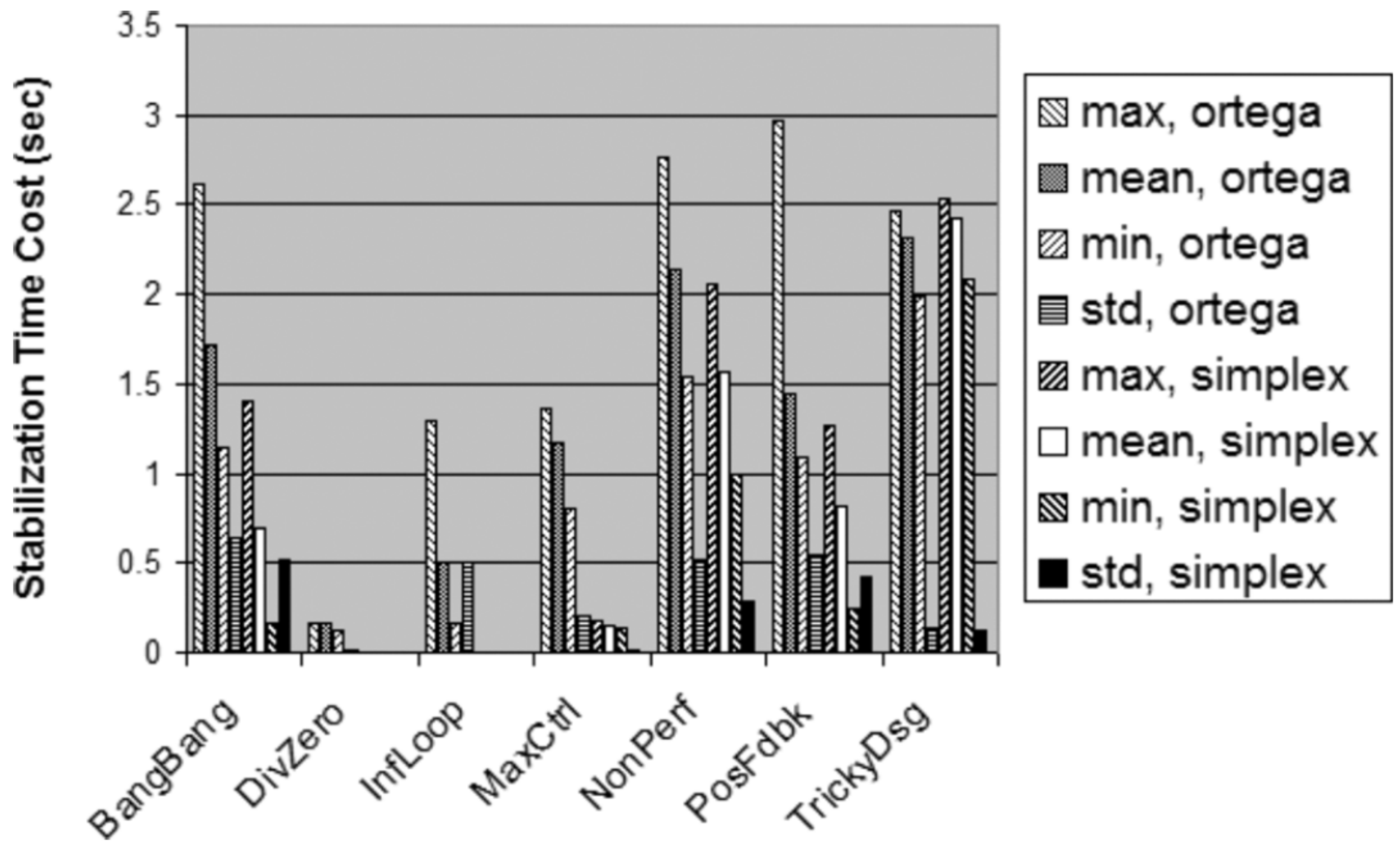
Controller	Average Execution Time (μs)	Variance of Execution Time	Minimum Execution Time (μs)	Maximum Execution Time (μs)
HPC	2.6705	0.02181	2.3571	3.2857
HAC	1.1060	0.005812	0.9429	1.6371

- If HAC and HPC both run at 50Hz, ORTEGA’s CPU saving is 29.29%
- If HAC runs at 50Hz, HPC runs at 20Hz, ORTEGA’s CPU saving is 50.87%

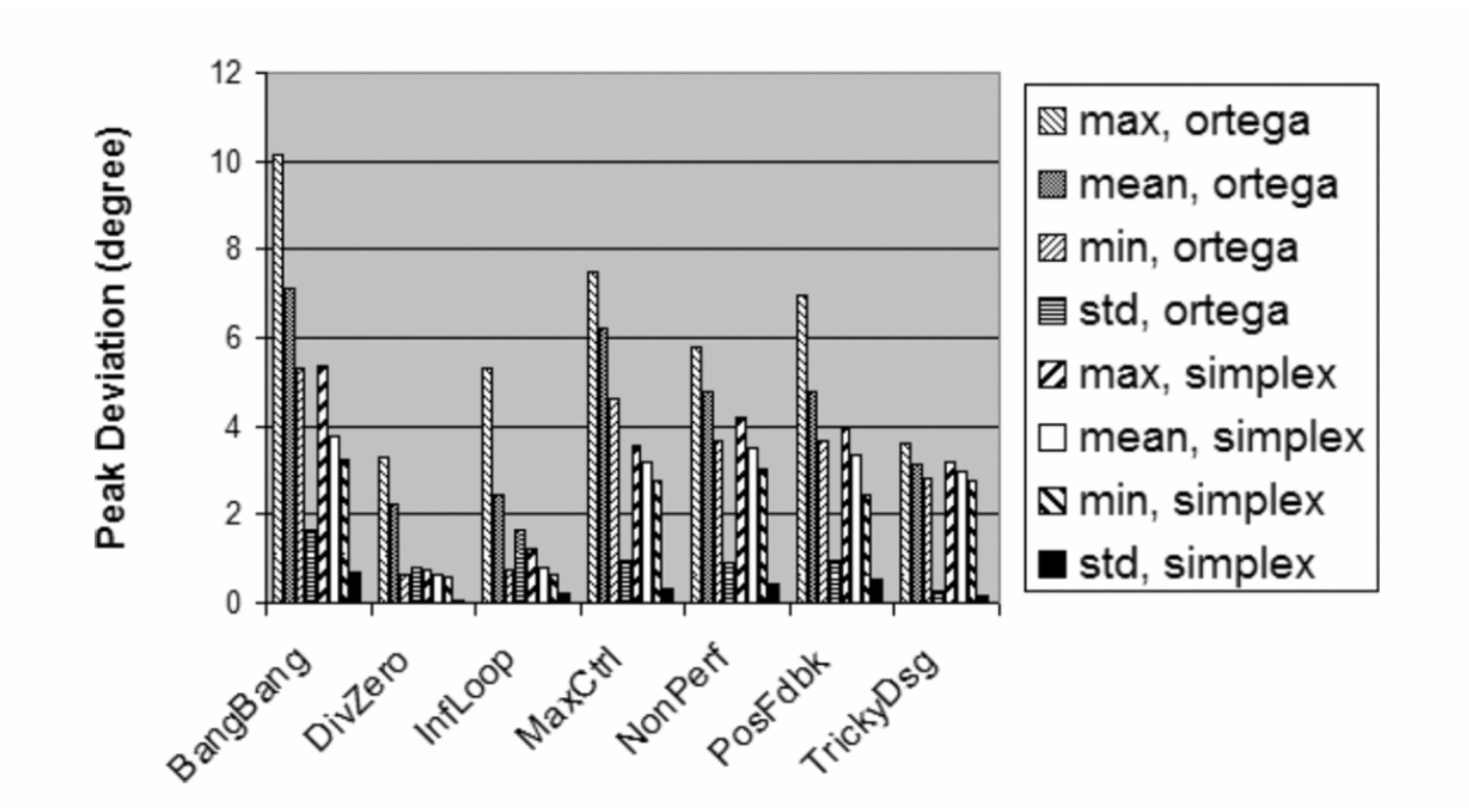
Evaluation of fault tolerance

- Infinite loop bug
- Non-performing bug
- Maximum control output bug
- Divided by zero bug
- Bang-Bang type bug
- Positive feedback bug
- Tricky design bug
- ...

Evaluation of fault tolerance



Evaluation of fault tolerance



Thank You

Q&A

Backup Slides

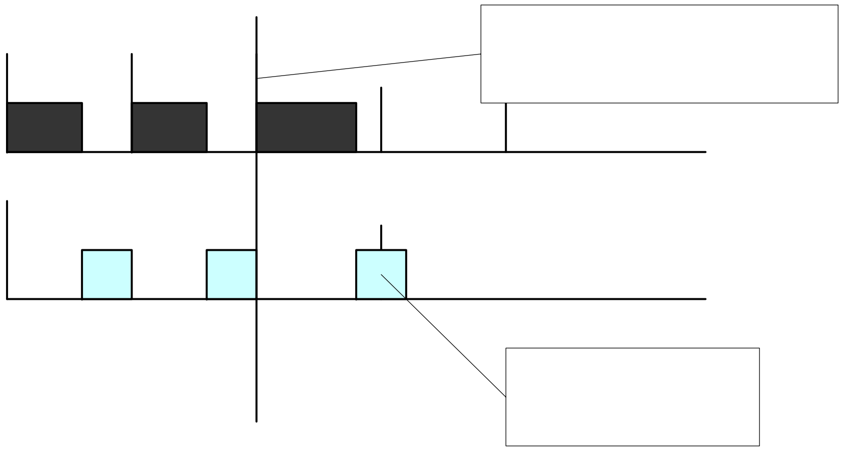
Schedulability analysis of ORTEGA

Mode-Change Problem Incurred by Recovery

Example: Suppose one plant τ_1^p : $(C_1^p, T_1^p) = (3,5)$; τ_1^a : $(C_1^a, T_1^a) = (4,10)$; with another real time task τ_2 : $(C_2, T_2) = (6,15)$.

- Before the recovery at t=10, $\{\tau_1^p , \tau_2\} = \{(3,5), \{6,15\}\}$ is schedulable;
- After the recovery transition, $\{\tau_1^a , \tau2\} = \{(4,10), \{6,15\}\}$ is also schedulable;

- However, during the transition of recovery, τ_2 misses its deadline at t=15!



Unschedulable of tasks due to the recovery

Mode-change in fixed priority scheduling is a well-recognized difficult problem by the real-time community

Schedulability Analysis

Schedulability Analysis: We adopt the work by Real and Crespo (2004)

Idea: Analyze the transitional scheduling overhead incurred by the recovery.

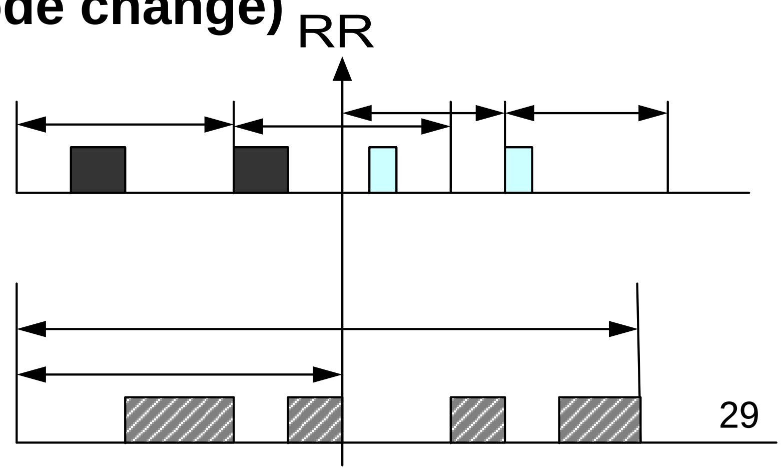
(I) Schedulability analysis of steady state task set

(II) Schedulability analysis of old-mode tasks with transitional scheduling overhead (due to the mode change)

$$w_i(x) = C_i + \left\lfloor \frac{x}{T_k^p} \right\rfloor C_k^p + \min\left(x - \left\lfloor \frac{x}{T_k^p} \right\rfloor T_k^p, C_k^p\right) + \left\lceil \frac{w_i(x) - x}{T_k^a} \right\rceil_0 C_k^a + \sum_{j<i, j\neq k} \left\lceil \frac{w_i(x)}{T_j} \right\rceil C_j \ .$$

(III) Schedulability analysis of new-mode tasks with transitional scheduling overhead (due to the mode change)

$$w_i = C_i + \left\lceil \frac{w_i}{T_k^a} \right\rceil C_k^a + \sum_{j<i, j\neq k} \left(\left\lceil \frac{w_i}{T_j} \right\rceil C_j \right) \ .$$



Fault Tolerance and Scheduling Co-design -- one FT-enabled task case

Maximize the recovery region subject to schedulability constraint

↓

Find the smallest (optimal) control loop period $T_k{}^{*a}$, s.t. the task set is schedulable under random recoveries

↓

Given the schedulability test, we can use binary search algorithm to find $T_k{}^{*a}$

Example: 3 tasks. $\tau_1 = (2, 4)$ and $\tau_3 = (3, 30)$ are ordinary real-time tasks. τ_2 is a FT-enabled task, with $\tau_2^p = (2, 8)$.

Numerical Solution:

(1) If $C_2^a = 2.0$, we have $T_2^{*a} = 6.5 < T_2^p$;
(2) If $C_2^a = 1.5$, we have $T_2^{*a} = 4.5 < T_2^p$;
(3) If $C_2^a = 1.0$, we have $T_2^{*a} = 3.0 < T_2^p$;
(4) If $C_2^a = 0.5$, we have $T_2^{*a} = 2.5 < T_2^p$.

P2: Recovery Region for Digital Controllers

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t), \Longrightarrow x(k+1) = F(h)x(k) + G(h)u(k),$$

Sampling time h,

Zero-order hold

$$F(h) = e^{Ah}, \quad G(h) = \int_0^h e^{As} ds B.$$

Controller $$u(k) = -Kx(k)$$

$$\Downarrow$$

$$x(k+1) = \bar{F}x(k) \qquad (\bar{F} = F - GK)$$

Theorem (Lyapunov): A discrete time LTI system shown above is stable iff there exists a matrix *P>0*, such that

$$\bar{F}^T P \bar{F} - P < 0.$$

Stability Region (Continued)

Stability region of the system $X(k+1) = \bar{F}x(k)$,

with respect to *P* is defined as: $\{x \mid x^T P x < 1\}$.

Stability Region with Constraints

State constraints $\quad a_i^T x \le 1, \qquad i = 1 \cdots l,$

Control input constraints $\quad b_j^T u \le 1, \qquad j = 1 \cdots r.$

Can be combined in the closed loop system as $\quad \alpha_m^T x \le 1, \quad m = 1, 刴 \;. \qquad (1)$

Lemma: The stability region defined above satisfy constraints (1) iff $\alpha_m^T P^{-1} \alpha_m \le 1, \qquad m = 1, 刴 \;.$