TOWARDS CONTROLLABLE DISTRIBUTED REAL-TIME SYSTEMS WITH FEASIBLE UTILIZATION CONTROL

X.WANG, Y. CHEN, C. LU, X. KOUTSOUKOS

Presented by: Aida Ehyaei

Outline

Motivation

- Traditional control approaches
  - rely on accurate knowledge about system workload
- Feedback control solutions
  - based on dynamic feedback
    - adapting to workload variations in unpredictable environments
- DRE*systems
  - Have unknown and varying workloads
    - Due to bursty users, cyber attack, (Aperidic tasks)

*Distributed Real-time Embedded (DRE)

Motivation

- Traditional control approaches
  - rely on accurate knowledge about system workload
- Feedback control solutions
  - based on dynamic feedback
    - adapting to workload variations in unpredictable environments
- DRE*systems
  - Have unknov
    - Due to burs

A suitable application for applying feedback control techniques

*Distributed Real-time Embedded (DRE)

Fundamental Problems

- Guaranteeing system controllability
  - Caused by lack of enough actuators in the system
- Guaranteeing system Feasibility
  - Caused by actuation constraints (e.g., rate constraints of periodic tasks in a DRE system)

Representative case study(1/3)

- <u>Multiprocessor utilization control problem</u>

- End-to-end utilization control
  - Guarantee the end-to-end deadline of all periodic tasks in a soft DRE system

- End-to-end scheduling: meeting the sub-deadline of each subtask
  - A well-known approach :
    - CPU utilization of the processor < schedulable utilization bound

Representative case study(2/3)

- Tasks with adjustable ranges may run slower to lower CPU utilization, keep it under schedulable bound

- While, a higher task rate
  - A higher value to the application
  - Better system QOS
  - Cost: higher CPU utilization

Representative case study(3/3)

- Desirable
  - Increase the system value by driving the processor utilizations close to the utilization bounds

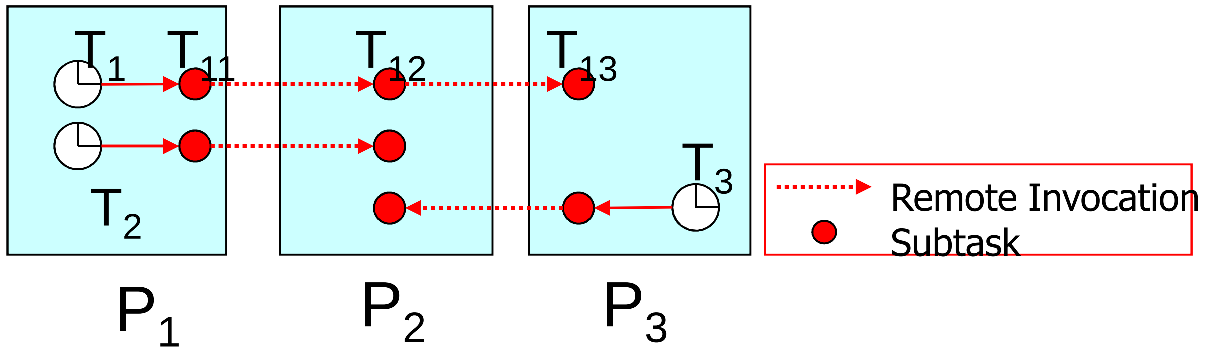
- With controllability and feasibility guarantees
  - maximize the system value
    - running all tasks at the highest possible rates without causing any deadline misses

Contributions of the paper

- Formulate the controllability and feasibility problem
  - As an end-to-end task allocation problem
- Design task allocation algorithms
- Develope runtime algorithm to reallocate tasks dynamically in response to workload variation
- Integrate the algorithms with a robust real-time middleware
- Present both empirical and numerical results

System Model

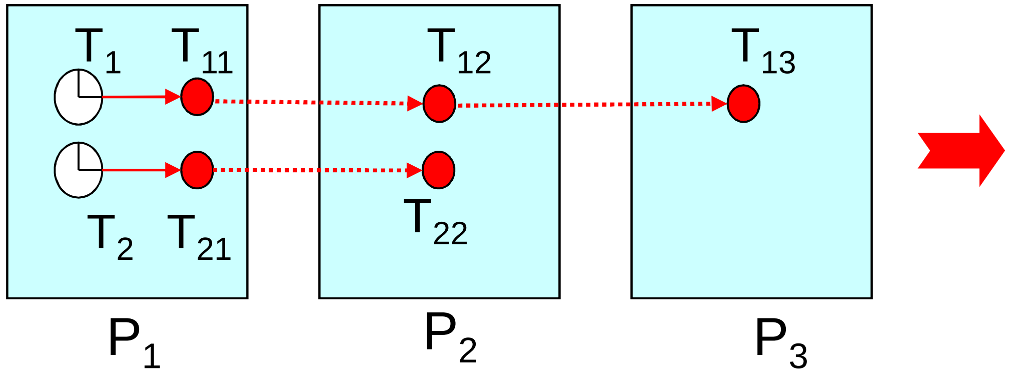
- *m* periodic tasks, *n* processors
- Each periodic task T_i = a chain of subtasks $\{T_{ij}\}$ located on different processors
  - Subtasks run at a same rate with precedence relation
- Each subtask T_{ij} has an estimated execution time c_{ij} available at design time
- Task rate may be dynamically adjusted within a range
  - $R_{min,i} \leq r_i(k) \leq R_{max,i}$ $(1 \leq i \leq m)$
- Subtasks may be moved to other processors at runtime



Dynamic Model

$$\boxed{\mathbf{u}(k+1) = \mathbf{u}(k) + \mathbf{GF}\Delta\mathbf{r}(k)}$$

- **u(k):** utilization control in k^{th} sampling point
- **G:** diagonal matrix of utilization gains
  - The ratio between the actual utilization change and its estimation
- **F**: subtask allocation matrix
  - models the coupling among processors
  - $f_{ij} = \Sigma c_{jl}$ for all subtasks T_{jl} of task T_j on processor P_i
  - $f_{ij} = 0$ if T_j has no subtask on P_i



$$\mathbf{u(k)} = \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} c_{11} & c_{21} \\ c_{12} & c_{22} \\ c_{13} & 0 \end{bmatrix}, \quad \mathbf{\Delta r(k)} = \begin{bmatrix} \Delta r_1(k) \\ \Delta r_2(k) \end{bmatrix}.$$

Control Matrix

- For $X(k+1)=PX(k)+QV(k)$ with
  - n control outputs X and
  - m control inputs V,

  - The controllability matrix is $C = [Q\ PQ\ \ldots\ P^{n-1}Q]$

Controllability Problem

- A Controllable DRE system: there exists a sequence of task rates that take the utilizations of all processors in the system to any desired utilization set-points

- According to the control theory
  - An MIMO System with *n* control outputs and *m* control inputs is controllable iff the rank of its controllability matrix is *n*, the order of the system

$\Rightarrow$ Guarantee:

The rank of the controllability matrix $(C=[FF\ldots F]_{n\times nm}) =$ *n* (the number of processors in the system)

Controllability Analysis

- **Theorem.** A DRE system is controllable if and only if the rank of its subtask allocation matrix F is n.
  - **Corollary.** A DRE system with n processors and m end-to-end tasks is uncontrollable if $m < n$ (a necessary but not sufficient condition)

- Structurally controllable: if there exists another system which is structurally equivalent to the system and is completely controllable
- structurally equivalent: there is a one-to-one correspondence between the locations of the fixed zeros and nonzero items in their controllability matrices

Feasibility Problem (1/2)

- A controllable DRE system is infeasible if it cannot get to the set points because the rates of one or more of its tasks saturate at the rate boundaries

- Utilization control for a DRE system is practically feasible if:

The utilizations of all processors $\leq$ The desired set points

- Instead of continuously monitoring feasibility and migrating subtasks (large runtime overhead)
  - Increase the likelihood of the system remaining feasible even under variations

Feasibility Problem (2/2)

- The minimum estimated utilization of a processor:

$$u_{min,i} = \sum_{T_{jl} \in S_i} c_{jl} R_{min,j}$$

- Feasibility margin $= B_i - u_{min,i}$
  - **B$_i$**: Utilization set point of processor P$_i$ ($1 \leq i \leq n$)

- Feasibility problem:

$$\max(\min_{1 \leq i \leq n}(|B_i - u_{min,i}|))$$

  - Subject to utilization constraint and resource constraint

Algorithm to Increase Feasibility Margin

```
(1) Enqueue all subtasks $T_{jl}$ in the order of decreasing $u_{min,jl}$;
(2) While there is at least one subtask in the queue,
        pop up the first subtask $T_{jl}$(which has the largest $u_{min,jl}$);
        For each  processor $P_q = cons\ (T_{jl}, q{+}{+})$,
            If $u_{current,q} + u_{min,jl} \leq B_q$
                $u_{new,q} = u_{current,q} + u_{min,jl}$;
                Feasibility margin of $P_q$: $B_q$- $u_{new,q}$;
            Endif;
        Endfor;
        Allocate $T_{jl}$ to provessor $P_i$ with the largest feasibility margin;
        If $T_{jl}$ cannot be allocated to any processor,
            Algorithm fails;
    Endwhile;
```

Ensuring controllability

- Dedicate task to each processor.
  - A task can only be dedicated to one processor
- Failed to find dedicated tasks for some processors?
  - Migrate subtasks of some non-dedicated tasks from other processors to them
- **Theorem:** If every processor in a system has a dedicated task, the system is controllable*

*both a sufficient and a necessary condition for controllability

Algorithm to ensuring controllability

(1) Initializes the two auxiliary matrices E and B and sorts all the processors based on their numbers of subtasks

(2) For every processor/task pair in the allocation matrix, search for a candidate subtask by assuming that the processor fails to find its dedicated task and needs a subtask of this task to be moved to the processor

(3) Sort all the existing subtasks of each processor in the E based on their minimum estimated utilizations

In B, sort the best candidate subtasks of each processor based on their minimum estimated utilizations

(4) Start the dedicating process

If no task can be dedicated to a processor, move the best candidate subtask of the first non-dedicated task to the processor

Pseudo Code For Controlability Algorithm (1/2)

```
(1) Create two $n \times m$ auxiliary matrices $\mathbf{E}$ and $\mathbf{B}$;
    $\mathbf{E}$ is used to calculate $u_{min,jl} = c_{jl} * R_{min,j}$ for every existing subtask $T_{jl}$ on each processor;
    $\mathbf{B}$ is used to find the best candidate subtask for each processor ;
    Initialize all elements in the auxiliary matrix $\mathbf{E}$ to be zero ;
    For every non−zero element $\mathbf{F}[i, j] = c_{jl}$ in the allocation matrix $\mathbf{F}$, $\mathbf{E}[i, j] = u_{min,jl}$;
    Initialize all elements in the auxiliary matrix $\mathbf{B}$ to be the maximum integer;
    Sort all the processors in the increasing order of number of subtasks ;

(2) For every column ( task ) in $\mathbf{E}$,
      Find the subtask $T_{jl}$ such that $u_{min,jl} = min(u_{min,jl'}\ (0 \leq l' \leq m_j))$;
      $T_{jl}$ is the subtask with the smallest minimum estimated utilization in task $T_j$;
      $T_{jl}$ is hence the best candidate if we need to move a subtask of task $T_j$ to a processor;
      For each processor $P_q$ that is allowed to execute $T_{jl}$ based on the constraints matrix $cons$,
         if $\mathbf{F}[q, j] = 0$, $\mathbf{B}[q, j] = u_{min,jl}$;
      Endfor;
    Endfor;
```

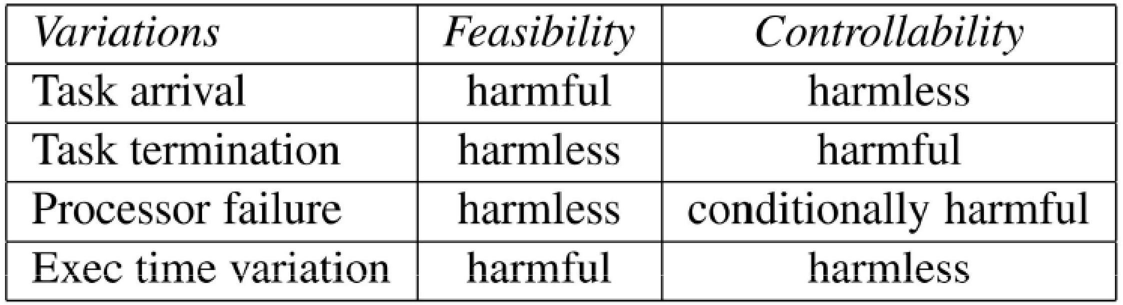
Pseudo Code For Controlability Algorithm (2/2)

```
(3) For each row of $\mathbf{E}$, sort this row in the decreasing order of $u_{min,jl}$;
    For each row of $\mathbf{B}$, sort this row in the increasing order of $u_{min,jl}$;

(4) For each row (processor $P_i$ in the increasing order of number of subtasks) in $\mathbf{E}$,
      For each subtask $T_{jl}$ that is sorted in Step 3,
        If task $T_j$ is not dedicated, dedicate $T_j$ to $P_i$ and exit the inner loop;
      Endfor;
      If all the current subtasks in the row of $P_i$ are already dedicated to other processors,
        In the corresponding row (processor $P_i$) in $\mathbf{B}$,
        For each subtask $T_{jl}$ that is sorted in Step 3,
          If task $T_j$ is not dedicated,
            Move the best candidate subtask $T_{jl}$ to $P_i$;
            Dedicate task $T_j$ to $P_i$;
            Adjust the allocation matrix $\mathbf{F}$ accordingly and exit the inner loop;
          Endif;
        Endfor;
        If cannot find a non-dedicated task, algorithm fails ;
      Endif;
    Endfor. // for each row
```

Runtime Analysis

- Impact of workload variations



Variations	*Feasibility*	*Controllability*
Task arrival	harmful	harmless
Task termination	harmless	harmful
Processor failure	harmless	conditionally harmful
Exec time variation	harmful	harmless

- **Theorem:** Processor failure is harmful to controllability if the failed processor has more than m-n+2
  - Conditionally harmful

- Any variation that increases system workload may cause the feasibility margin to decrease

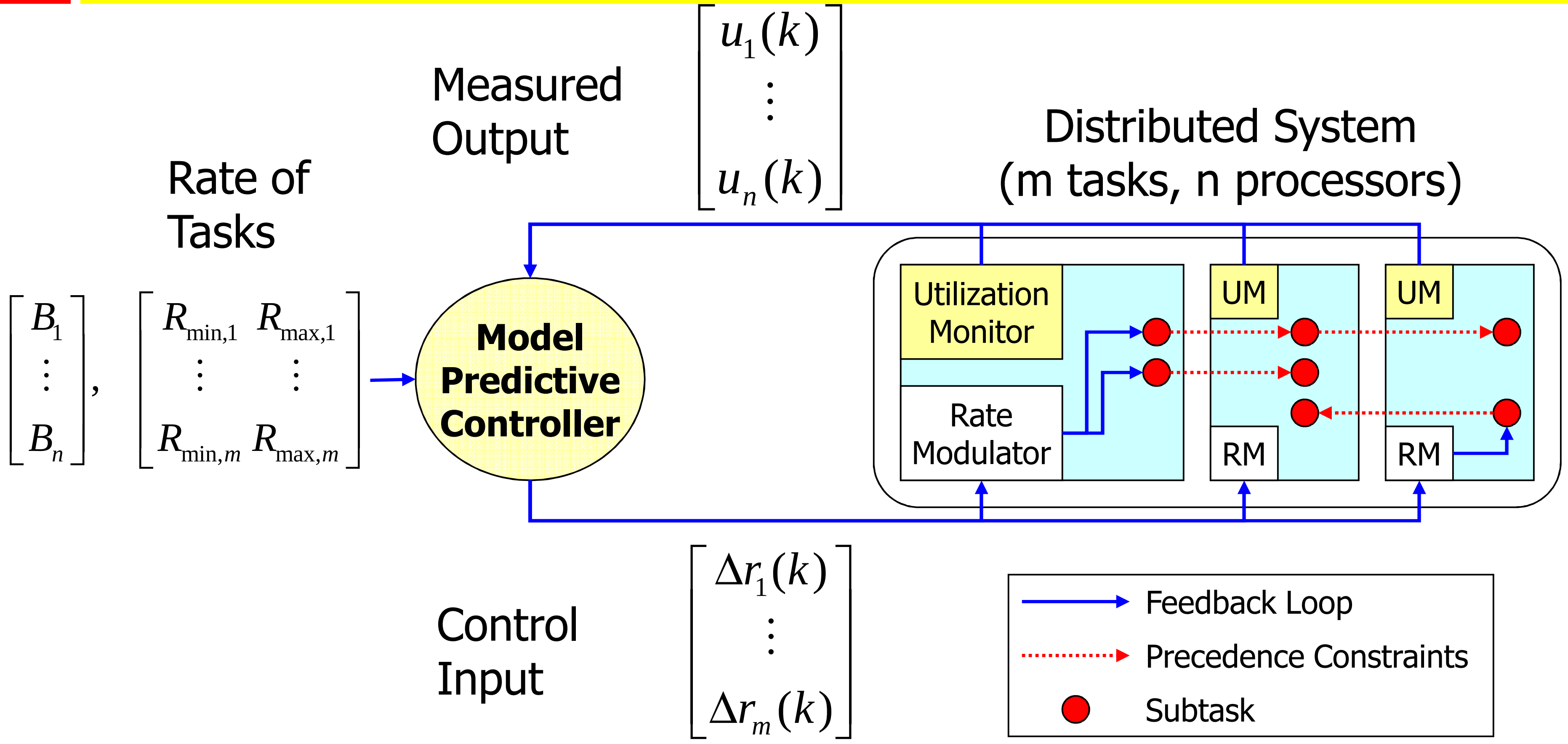
Runtime Adjustments

- For feasibility
  - Execution time variation: handled by feasibility margin
  - Task Arrivals: Sort and allocate only the new arriving tasks in Algorithm 1

- For controllability

```
(1)Remove the terminated task from the allocation matrix;
(2) If this task is not dedicated to a processor,
        Algorithm successfully ends;
(3) Else,
        For the processor that the terminated task was dedicated to,
            Run step 4 to find a dedicated task;
    Endif.
```

EUCON*: Multi-Input-Multi-Output Control

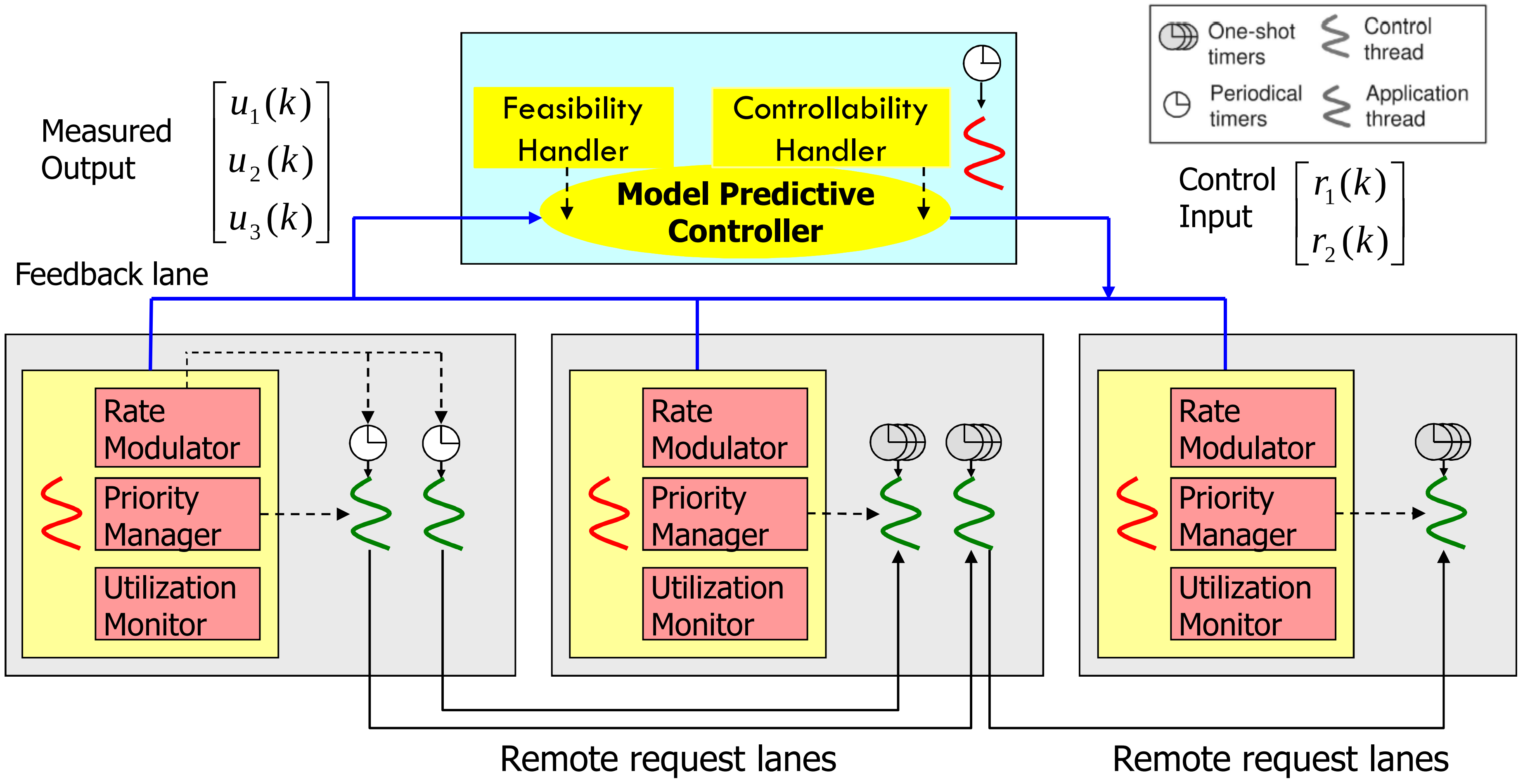


Measured Output
$\begin{bmatrix} u_1(k) \\ \vdots \\ u_n(k) \end{bmatrix}$
Distributed System (m tasks, n processors)
Rate of Tasks
$\begin{bmatrix} B_1 \\ \vdots \\ B_n \end{bmatrix}, \begin{bmatrix} R_{\min,1} & R_{\max,1} \\ \vdots & \vdots \\ R_{\min,m} & R_{\max,m} \end{bmatrix}$
Model Predictive Controller
Utilization Monitor
Rate Modulator
UM
UM
RM
RM
Control Input
$\begin{bmatrix} \Delta r_1(k) \\ \vdots \\ \Delta r_m(k) \end{bmatrix}$
Feedback Loop
Precedence Constraints
Subtask

*An end-to-end utilization control algorithm

C. Lu, X. Wang and X. Koutsoukos, Feedback Utilization Control in Distributed Real-Time Systems with End-to-End Tasks, IEEE Transactions on Parallel and Distributed Systems, 16(6): 550-561, June 2005.

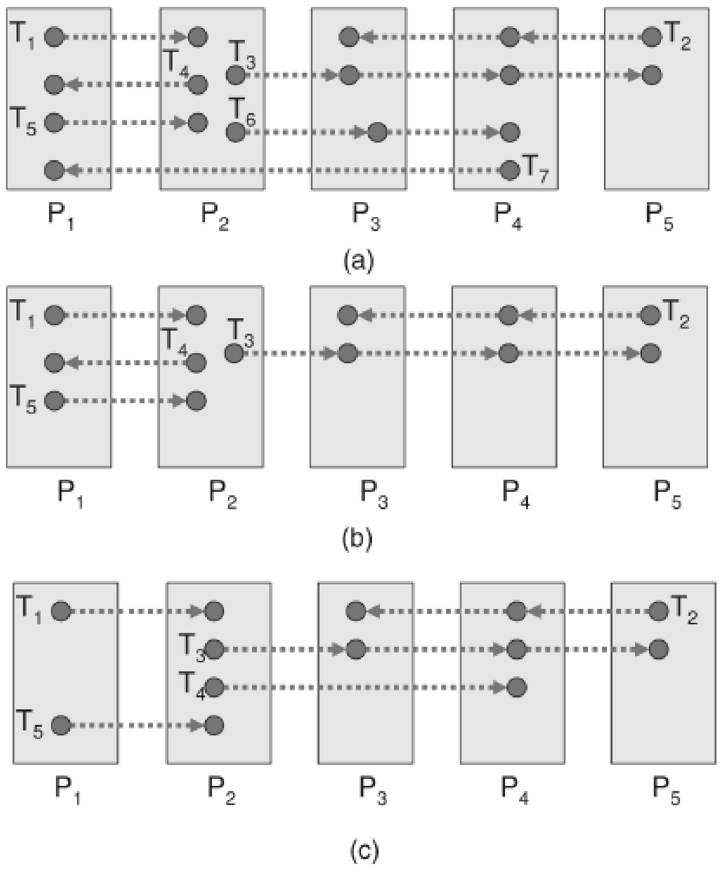
FC-ORB Middleware



Utilization set point of every processor : 0.7

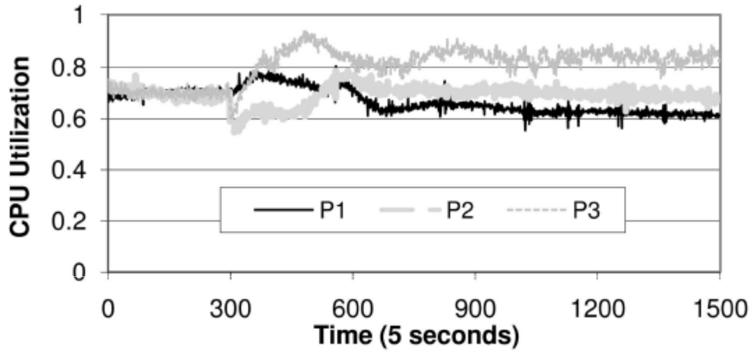
Controllability Experiments

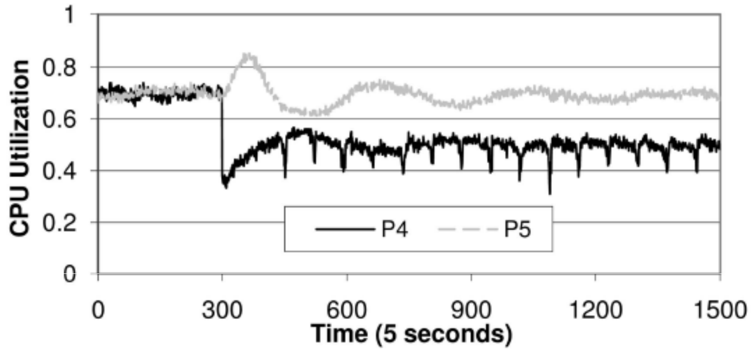
- Workload configuration and variations
  - (a) Initial task allocation
  - (b) Allocation after task termination (T_6 and T_7)
  - (c) Allocation after controllability maintenance



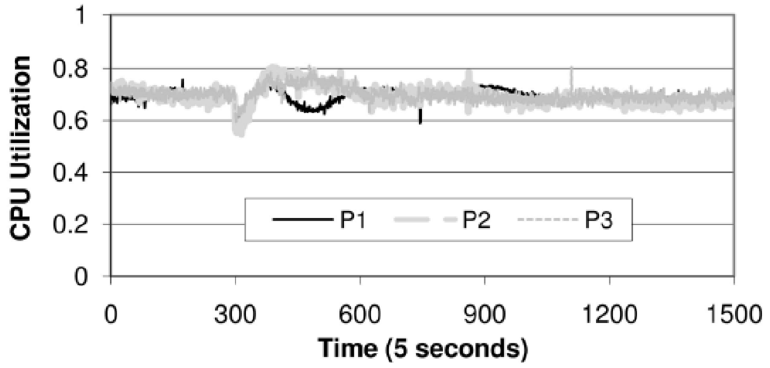
Results

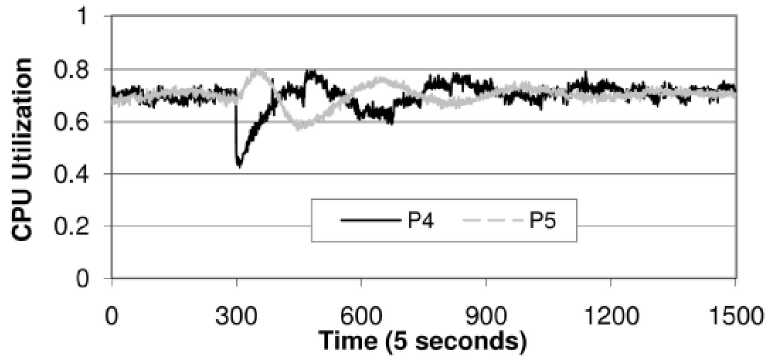
- After termination T6 and T7





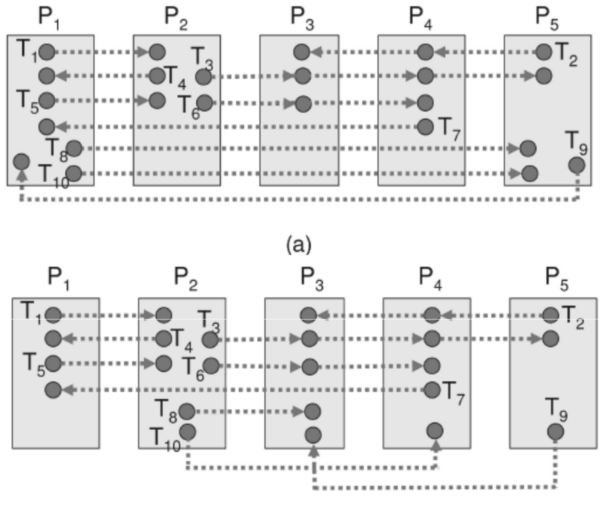
- After controllability maintenance





Feasibility Experiments

- T_8, T_9, and T_{10} are arrived at time 300×5
  - (a) Task allocation after naive solution
  - (b) Allocation after feasibility adjustment



(a)

(b)

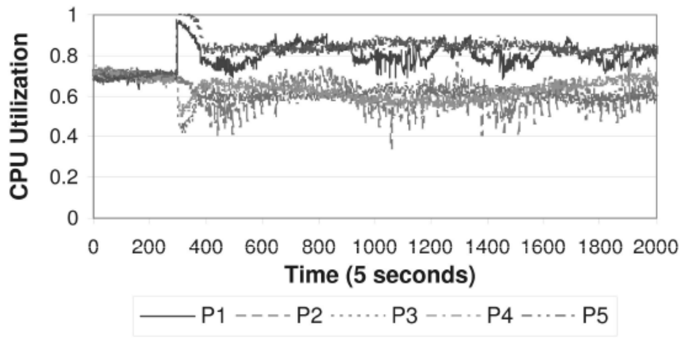
- Task Rates of All Tasks (S Means Rate Saturated)

	T_1	T_2	T_3	T_4	T_5
Naive	20 (S)	30.6569	5 (S)	29.9830	5.6836
Feasibility	43.1741	20.6556	19.3107	11.6857	5.0194
	T_6	T_7	T_8	T_9	T_{10}
Naive	20 (S)	50.4092	10 (S)	10 (S)	10 (S)
Feasibility	5.0004	50.5294	10.0018	11.1398	10.0008

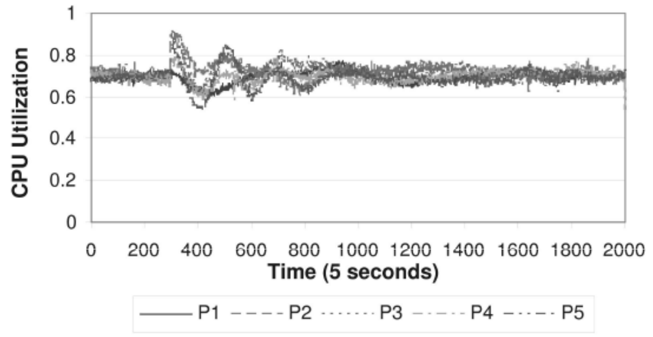
- Naive solution: simply keeping processor utilizations under their bounds.

Results

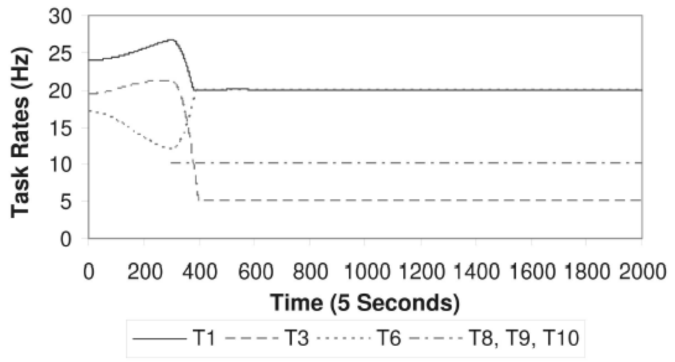
- System becomes infeasible after task arrivals



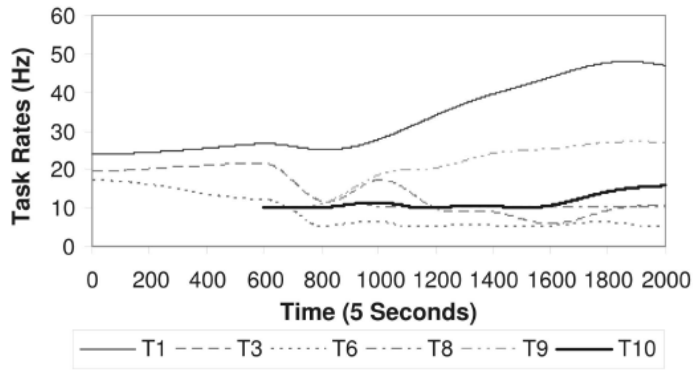
- System remains feasible after feasibility adjustment



- Task rates saturate at boundaries when the system is infeasible

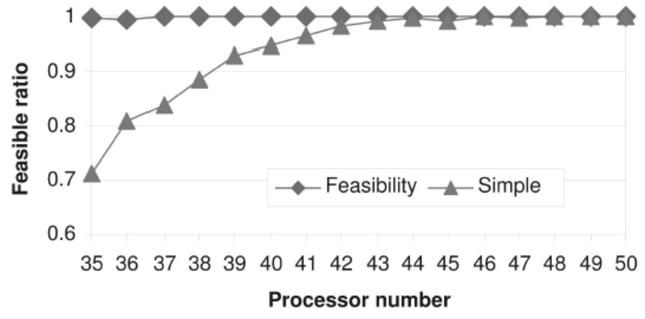


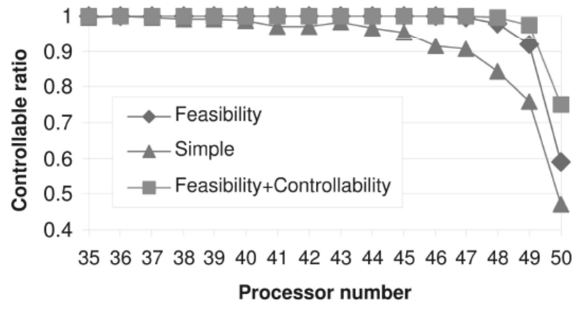
- Task rates no longer saturate after feasibility adjustment

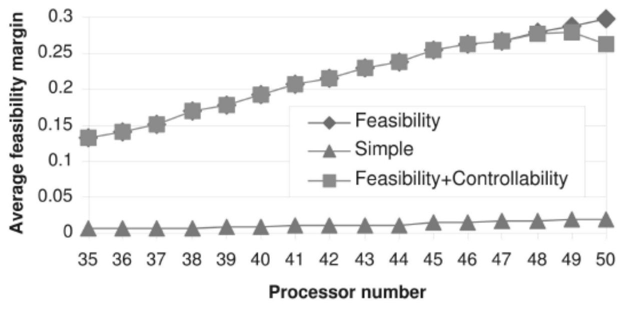


Numerical results

- Evaluate offline task allocation algorithms
  - Feasible ratio under different processor numbers
  - Controllable ratio under different processor numbers
  - Feasibility margin under different processor numbers







Simple algorithm: typical bin-packing-based allocation solution without the consideration of controlleability or feasability

Summary

- Controllability and feasibility
  - Fundamental properties of DRE systems
  - Crucial to the success of feedback control in such systems
  - Depend on end-to-end task allocations
  - Without them DRE systems often cause
    - Processor overload
    - Deadline misses
    - Undesired low task rates

- Offline and online task allocation algorithms are presented to ensure system controllability and feasibility
  - Meeting the end-to-end deadlines of all tasks is guaranteed
  - run all tasks at the highest possible rates
  - System value is increased

Questions?

Thank you for your attention!