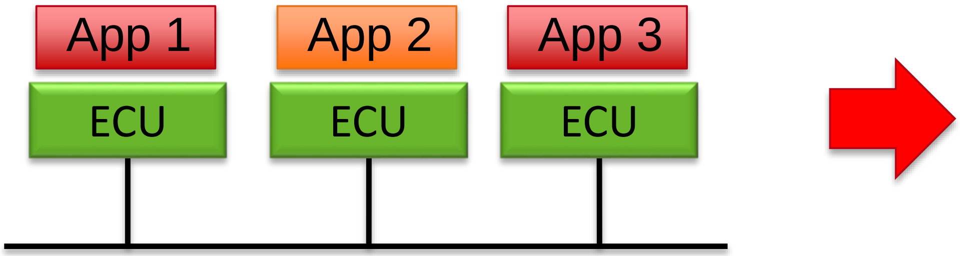
Adaptation of real-time scheduler RUN to Mixed-Criticality Systems

Romain GRATIA IRT-SystemX

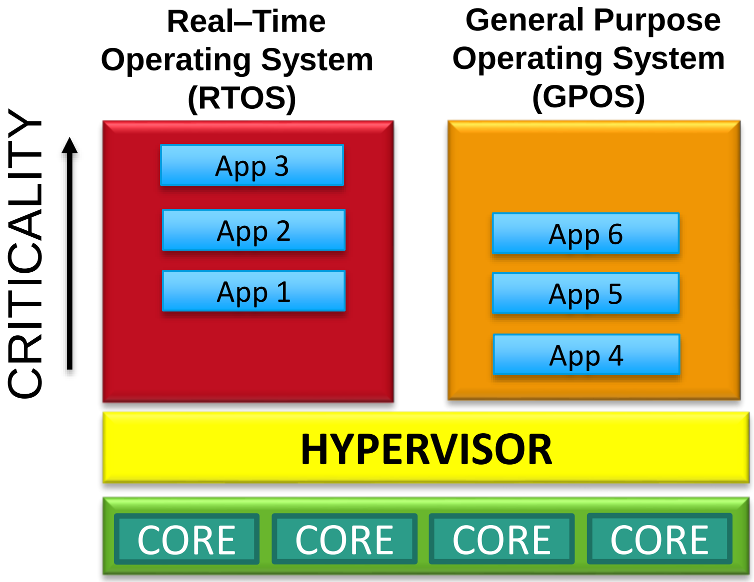
Thomas ROBERT Institut Mines-Telecom

Laurent PAUTET Institut Mines-Telecom

Current Architecture: Federated Architecture



Considered Architecture: Integrated Architecture



SO HOW TO PROPERLY SCHEDULE ALL THESE APPLICATIONS ?

- **Each application is modelled as a set of tasks, more precisely Real-Time tasks**

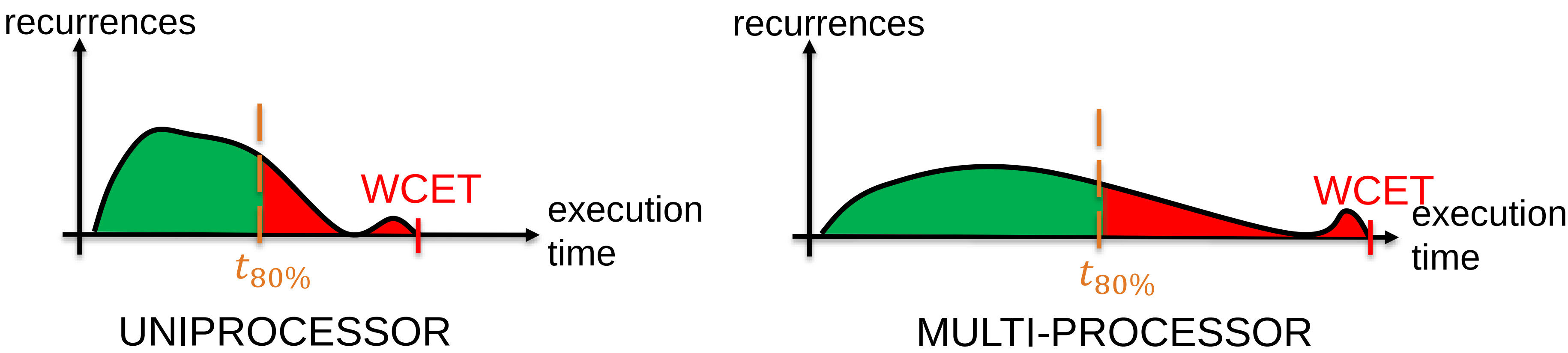
- **Each Real-Time task T_i follows the Periodic Task model:**
  - Period/Deadline P_i (implicit deadline)
  - Worst Case Execution Time (WCET) C_i
  - Utilization $U_i = \frac{C_i}{P_i}$

- **Refined problem : deploying these applications on a multicore platform = problem of preemptively scheduling real-time tasks on a multicore platform**

- **Task set has to pass a schedulability test based on utilization bound :**

$$U_{set} = \sum U_i \leq L(M, A)$$

- **Complexity of multi-core architecture has changed the way WCET are assessed :**



- **Implicit hypothesis: all tasks are paramount to system dependability**
  - Use of WCETs unavoidable
  - Poor effective utilization of the platform

- Concepts and definitions behind Mixed-criticality
- Presentation of RUN principles
- Adaptation of RUN to Mixed-Criticality

Solution: several operating modes

Consider that some tasks can be sacrificed (stopped):

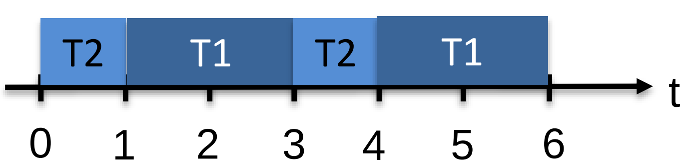
- **Classify tasks into two "criticality levels"**
  Lo (not paramount) and Hi (paramount)
- **The scheduler allocates a CPU time budget B_i to each task T_i**
- **Monitor execution times to enforce two execution modes :**
  - Full mode (F-mode) :
    - Execute all tasks
      Remark: In most cases, tasks completed consuming less than B_i
    - Set B_i to an optimistic time bound (e.g. $t_{80\%}$)
  - Safe mode (S-mode):
    - Execute only Hi-criticality tasks
    - $B_i = C_i$ required
- **Task scheduling starts in F-mode and switch to S-mode**
  as soon as required…

The mode change problem

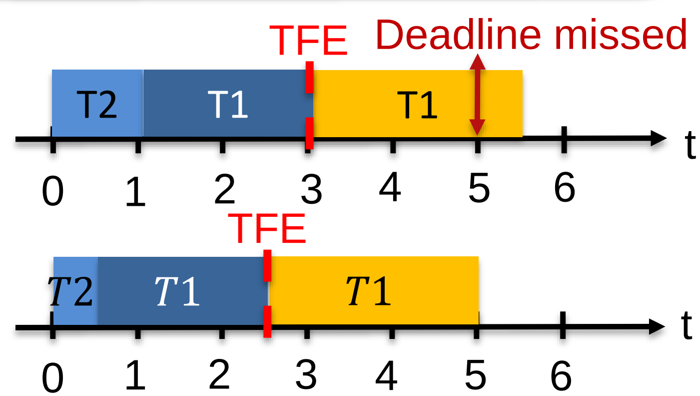
- **Timing Failure Event (TFE): a task depleted its budget without completing**
- **TFE triggers a Mode Change :**
  - Lo-criticality tasks are discarded
  - Use of the S-mode timing parameters

Task	Period	Criticality	B_{F-mode}	B_{S-mode}
T1	5	Hi	2 (U_1=0,4)	4,5 (U_1=0,9)
T2	3	Lo	1 (U_2=0,33)	1 (U_2=0,33)

Execution without Mode Change



Execution with Mode Change

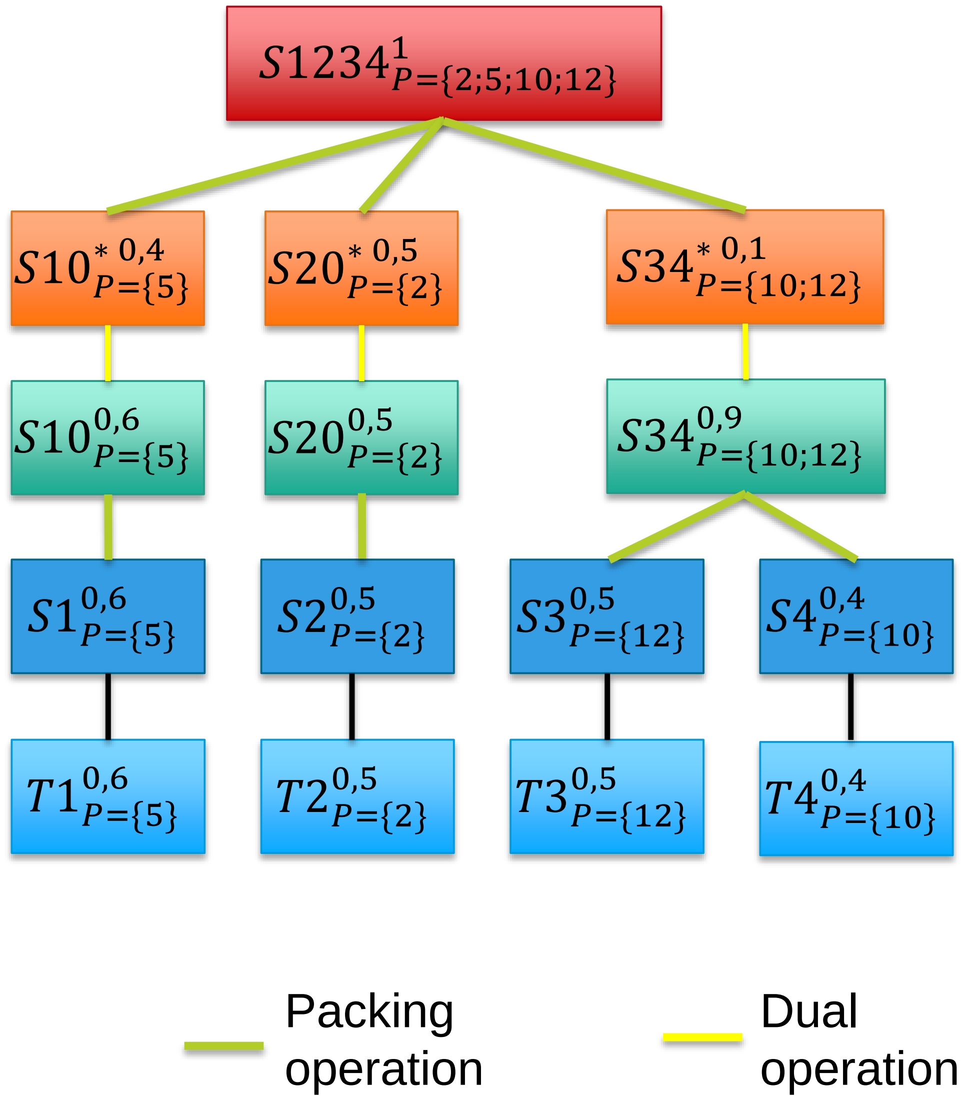


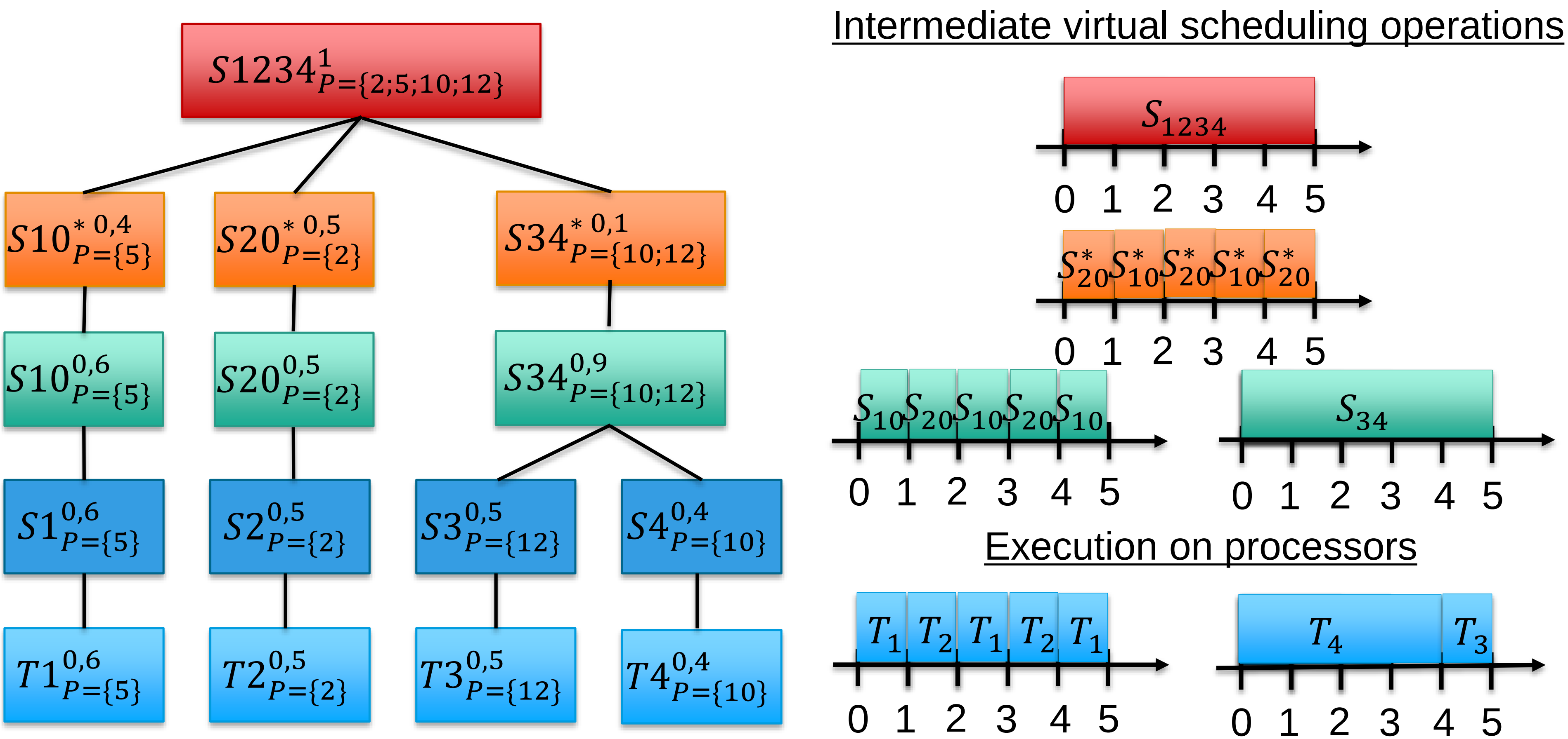
- **RUN aims at transforming a multiprocessor scheduling problem into multiple uniprocessor ones:**
  - Scheduling based on a hierarchy of servers
  - Two kinds of servers:
    - Primal Server: schedule other servers with EDF scheduling policy
    - Dual Server: corresponds to idle time of primal server
  - The hierarchy is generated **offline**…
  - … and is used **online** to take the scheduling decisions
- **For each task set RUN will use the least number of processors necessary for its proper scheduling**

Generation of the hierarchy of servers

Task	Period	Utilization
T1	5	0,6
T2	2	0,5
T3	12	0,5
T4	10	0,4

Number of processors required: 2





$S1234^{1}_{P=\{2;5;10;12\}}$
$S10^{*\,0,4}_{P=\{5\}}$
$S20^{*\,0,5}_{P=\{2\}}$
$S34^{*\,0,1}_{P=\{10;12\}}$
$S10^{0,6}_{P=\{5\}}$
$S20^{0,5}_{P=\{2\}}$
$S34^{0,9}_{P=\{10;12\}}$
$S1^{0,6}_{P=\{5\}}$
$S2^{0,5}_{P=\{2\}}$
$S3^{0,5}_{P=\{12\}}$
$S4^{0,4}_{P=\{10\}}$
$T1^{0,6}_{P=\{5\}}$
$T2^{0,5}_{P=\{2\}}$
$T3^{0,5}_{P=\{12\}}$
$T4^{0,4}_{P=\{10\}}$
Intermediate virtual scheduling operations
S_{1234}
0 1 2 3 4 5
S^*_{20} S^*_{10} S^*_{20} S^*_{10} S^*_{20}
0 1 2 3 4 5
S_{10} S_{20} S_{10} S_{20} S_{10}
0 1 2 3 4 5
S_{34}
0 1 2 3 4 5
Execution on processors
T_1 T_2 T_1 T_2 T_1
0 1 2 3 4 5
T_4 T_3
0 1 2 3 4 5

Objective:

Reduce the required number of processors to schedule a task set compared to non-modified RUN

- **First ensure the proper scheduling of the S-mode**
- **Enable Lo tasks when Hi tasks complete early:**
  - Split Hi-criticality task budgets into two => *start* and *finish* tasks
    start budget B_i^{F-mode}
    finish budget $B_i^{S-mode} - B_i^{F-mode}$
  - Re-allocate finish budget to schedule Lo task in F-mode
- **Introduce Modal Servers into RUN hierarchy of servers**
- **Create another RUN schedule for remaining Lo tasks**

- **Modal server = execution server with a periodically-replenished budget**
  - Schedule a set of Lo tasks in F-mode
  - Schedule one Hi-criticality task (*finish* task)
- **Two schedulability tests for Lo task scheduling**
  - When all task periods are a multiple of Modal Server period Modal Server utilization = schedulability bound
  - Otherwise, MS is less effective and provides a fewer amount of utilization to Lo tasks
- **How to assign Lo tasks to Modal Servers based on its utilization ?**
- **Allocation problem has been transformed into an accessibility problem**

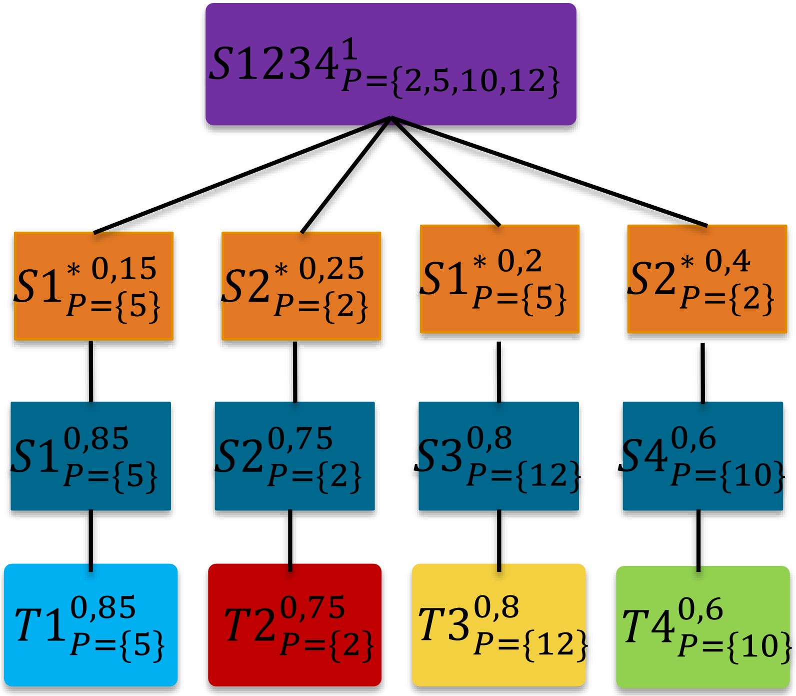
Example

Task	Period	Criticality	Utilization (F-mode)	Utilization (S-mode)
T1	5	Hi	0,6	0,85
T2	2	Hi	0,5	0,75
T3	12	Hi	0,5	0,8
T4	10	Hi	0,4	0,6
T5	8	Lo	0,25	0
T6	15	Lo	0,25	0
T7	3	Lo	0,5	0
T8	20	Lo	0,125	0

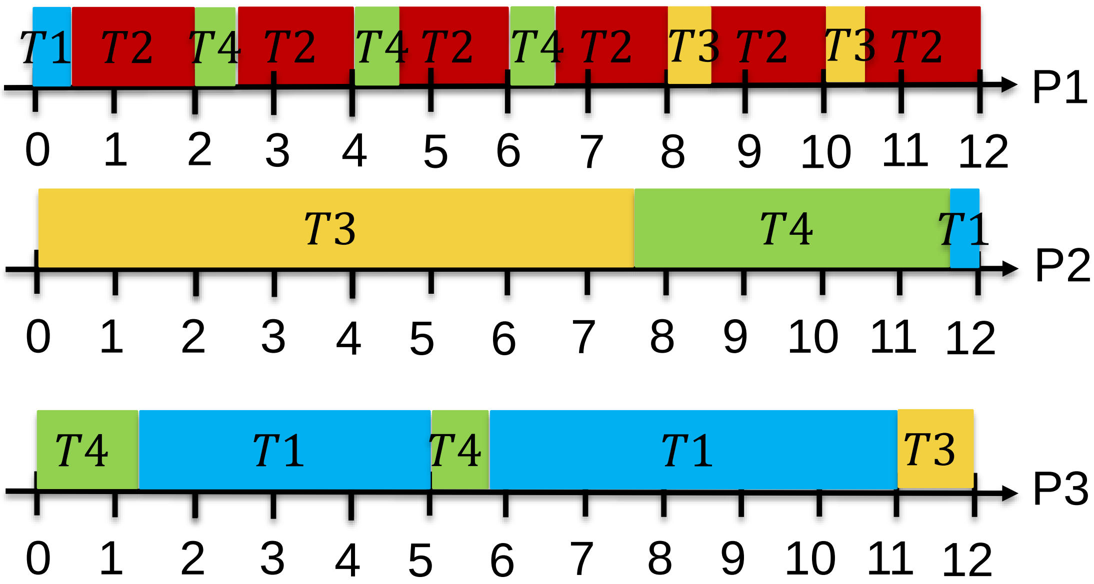
- **Scheduling Lo/F-mode+Hi/S-mode tasks RUN needs 5 processors**
- **Scheduling F-mode needs 4 processors**
- **Scheduling S-mode needs 3 processors**
- **Objective: schedule this system with fewer than 5 processors**

- **Generate the hierarchy of servers for Hi tasks only**

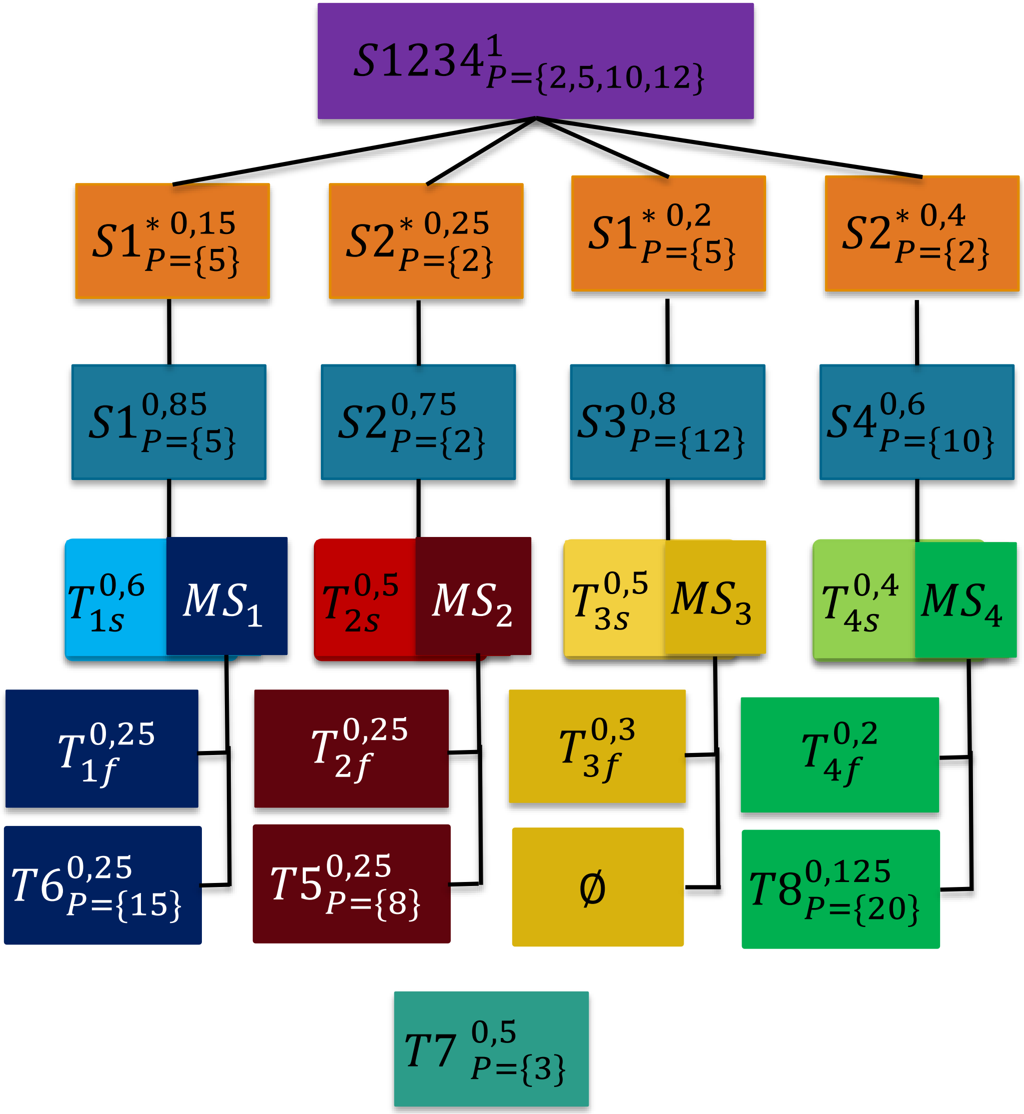
Offline



Online



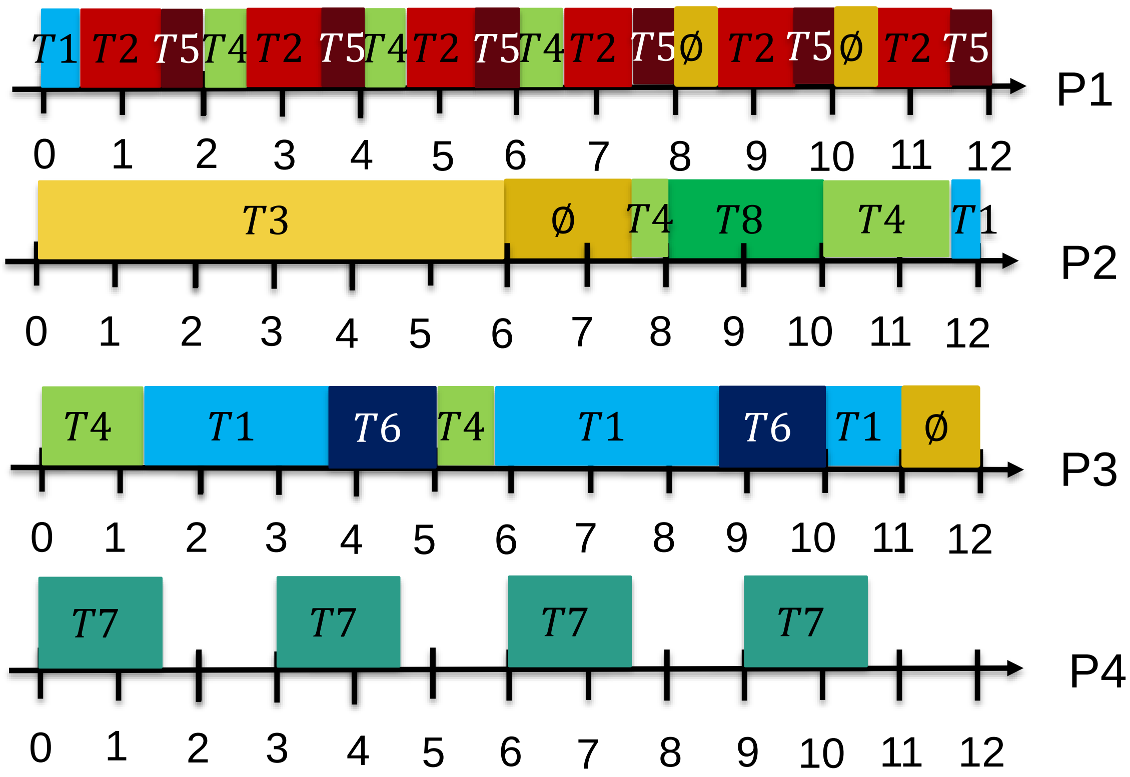
Offline scheduling of the F-mode



- **Start from S-mode hierarchy of servers**
- **Separate Hi tasks into 2 parts**
- **Allocate Lo tasks to Modal Servers**
- **Schedule independently remaining Lo tasks**

Processors scheduled by the modified S-mode hierarchy of servers

Processor scheduled independently for other Lo-criticality tasks



Results

Set of Tasks / Mode	Ceiling utilization
All/S-mode	5
Lo/F-mode	2
Hi/S-mode	3
Adapted RUN	4

- **Our adaptation requires fewer processors to schedule this task set**
- **Lo task allocation done with Uppaal**
- **Better result than fpEDF-VD requires : 8 processors**

- 1^{st} adaptation of RUN to Mixed-Criticality systems
- Optimality is lost
- Mode changes correctly handled
- Fewer processors requires for scheduling task set
- Use of Uppaal for Lo task allocation
- Future works:
  - Extensive experiments to be conducted

Any questions ?