Generalized Mixed-Criticality Scheduling based on RUN

Romain GRATIA — IRT SystemX

Thomas ROBERT — Télécom ParisTech

Laurent PAUTET — Télécom ParisTech

Automotive Electronics and Software Project

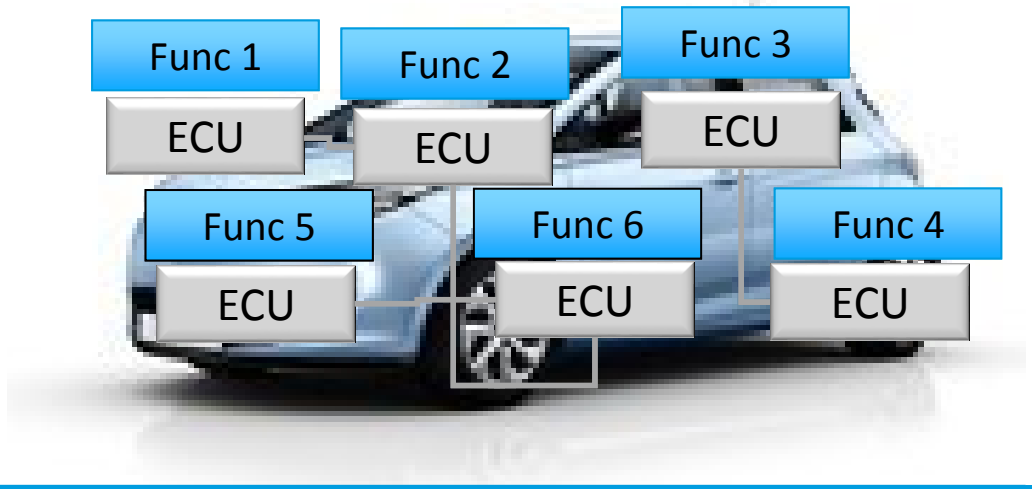
Real-Time Networks and Systems 2015

Context

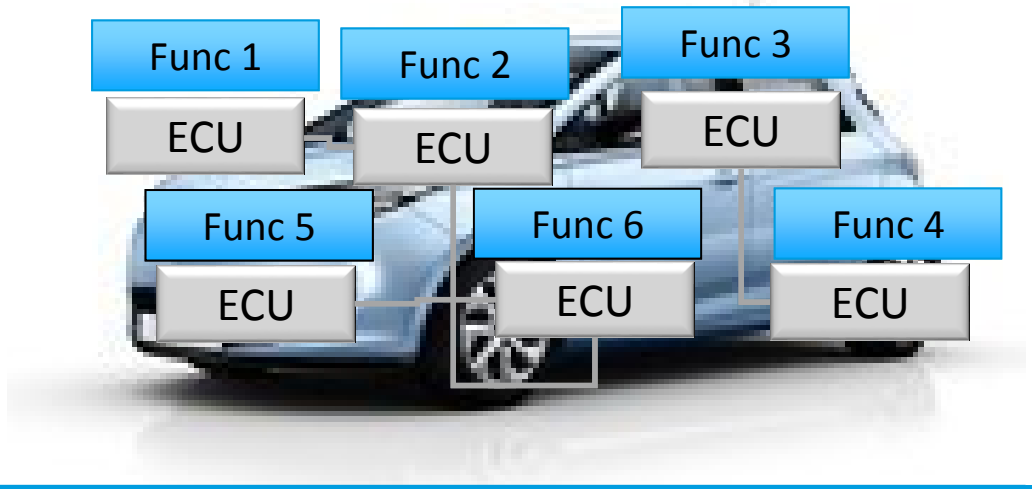
Embedded system design

Context

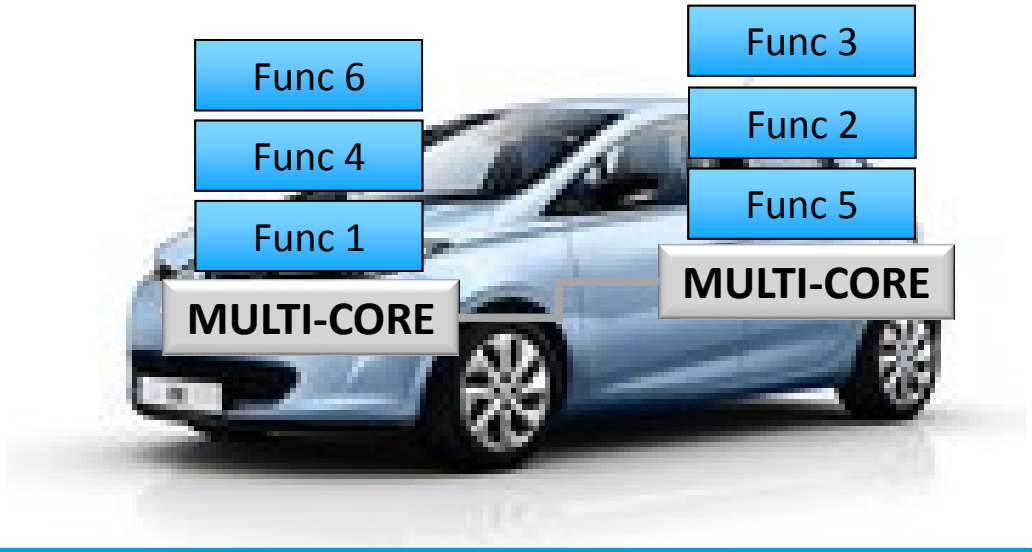
Embedded system design



- **Design revisited as:**
  - Many functions => High conception complexity
  - Uniprocessor replaced by multiprocessor

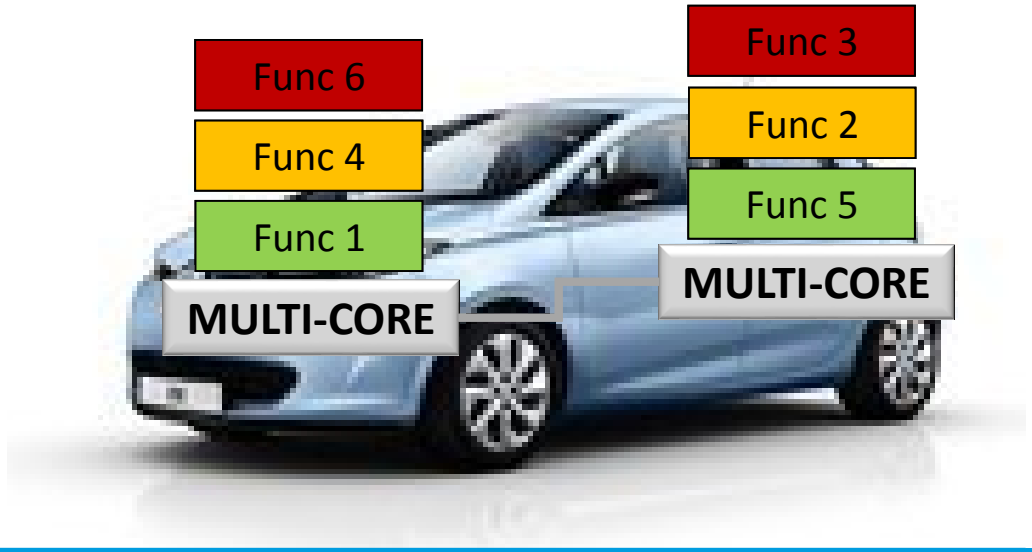


- **Design revisited as:**
  - Many functions => High conception complexity
  - Uniprocessor replaced by multiprocessor

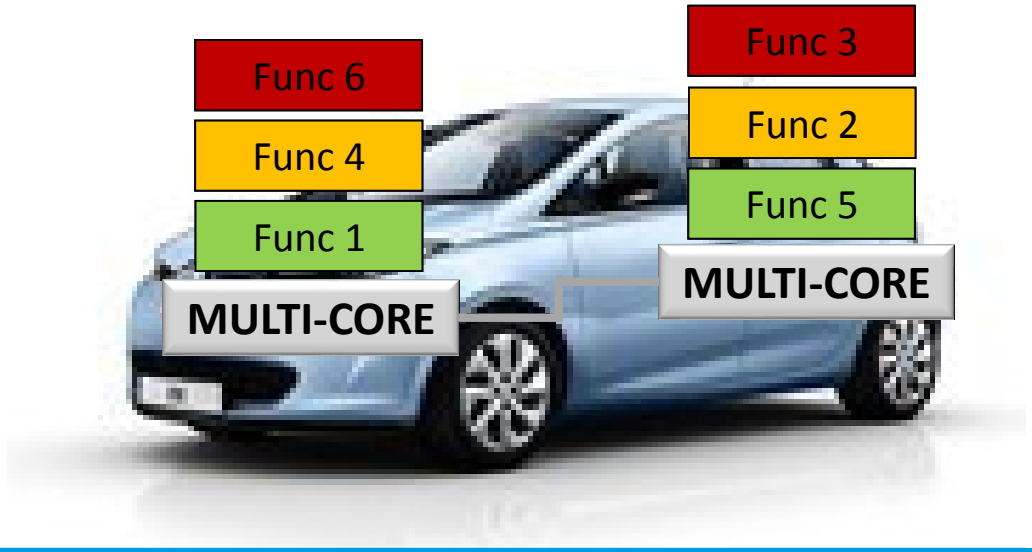


Embedded system design

- **Design revisited as:**
  - Many functions => High conception complexity
  - Uniprocessor replaced by multiprocessor
- **Mixed safety requirements**
  - Classification in several criticality levels (CL)
  - First approach: isolation => partitioning based on CL… but inefficient



- **Design revisited as:**
  - Many functions => High conception complexity
  - Uniprocessor replaced by multiprocessor
- **Mixed safety requirements**
  - Classification in several criticality levels (CL)
  - First approach: isolation => partitioning based on CL… but inefficient
- ↳ **Mixed-Criticality (MC) scheduling**

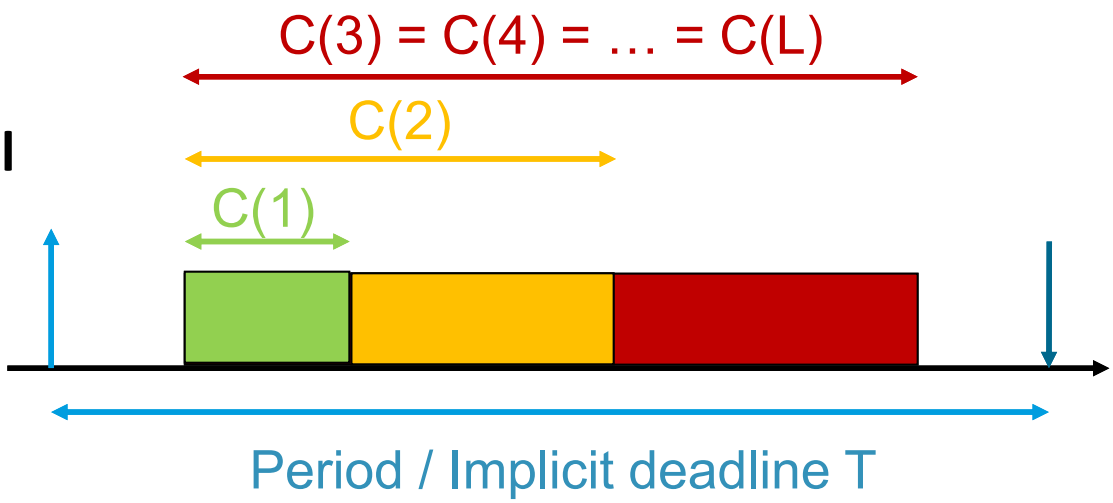


Task timing parameters in mixed-criticality systems

- **MC task model = Vestal periodic task model with implicit deadlines**
- **Execution time budget estimation depends on criticality level:**

Example:

In a system with L criticality level a CL-3 Task has a budget for each criticality level



- **Functionalities modeled by set of MC tasks**
- **Degraded execution mode = shutdown lower criticality tasks**

Execution Modes

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

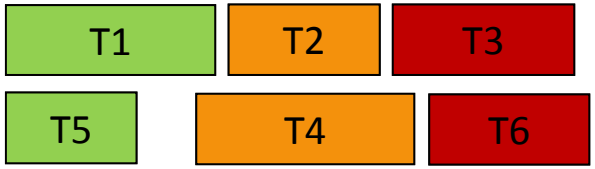
Execution Modes

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

Mode X=1:

Fully-featured mode



Active

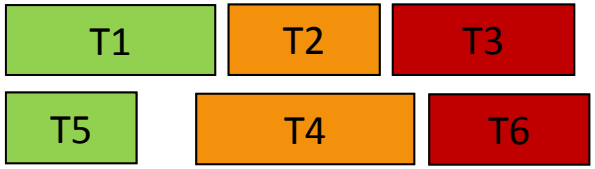
- **System starts in mode 1**

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

Mode X=1:

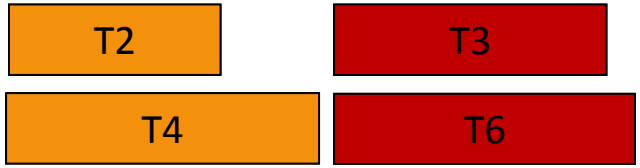
Fully-featured mode



Active

Mode X=2:

Important functionalities



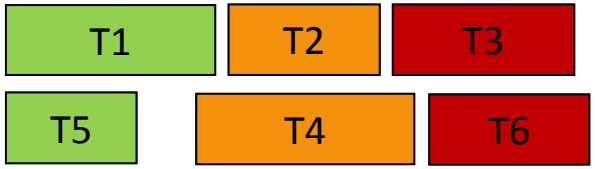
- **System starts in mode 1**

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

Mode X=1:

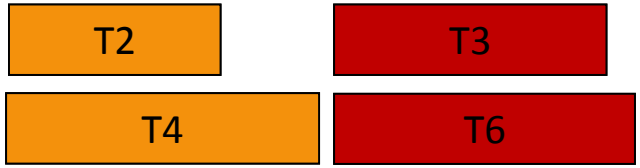
Fully-featured mode



Active

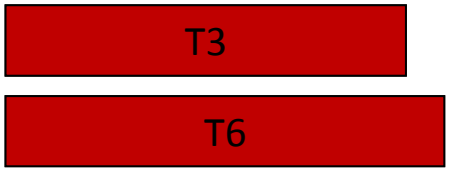
Mode X=2:

Important functionalities



Mode X=3:

Vital functionalities



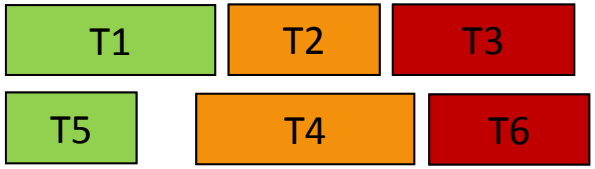
- **System starts in mode 1**

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

Mode X=1:

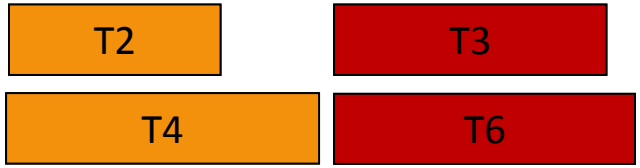
Fully-featured mode



Active

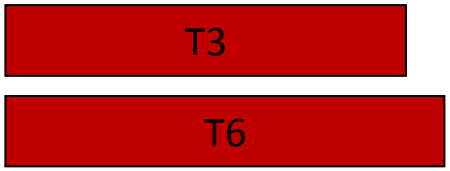
Mode X=2:

Important functionalities



Mode X=3:

Vital functionalities



- **System starts in mode 1**

  Timing Failure Event

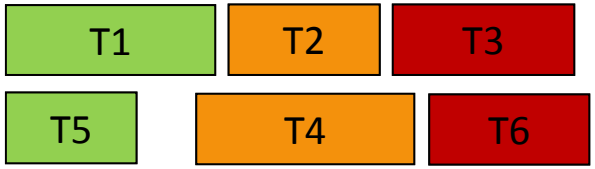
Execution Modes

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

Mode X=1:

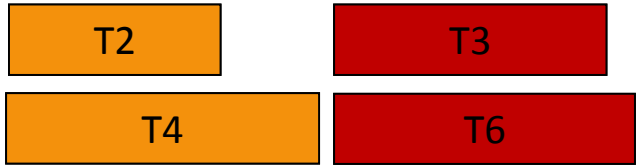
Fully-featured mode



Active

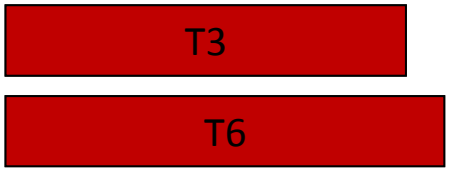
Mode X=2:

Important functionalities



Mode X=3:

Vital functionalities



- **System starts in mode 1**

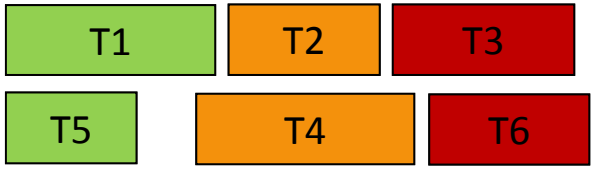
  Timing Failure Event → Mode change

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

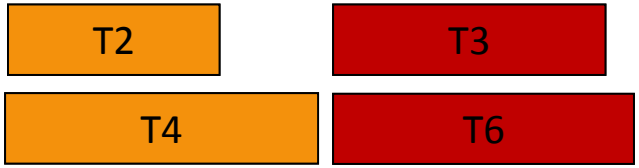
Mode X=1:

Fully-featured mode



Mode X=2:

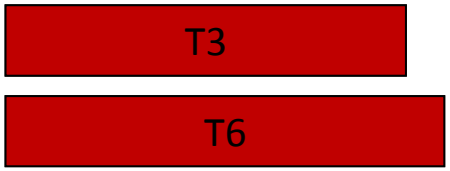
Important functionalities



Active

Mode X=3:

Vital functionalities



- **System starts in mode 1**

  Timing Failure Event → Mode change

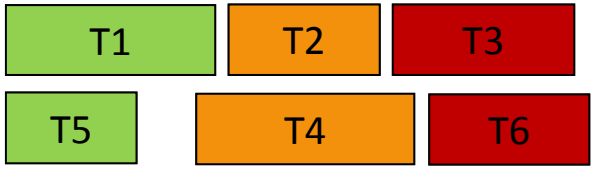
Execution Modes

- **Execution mode X =**

  Execute tasks of criticality level ≥ X ($\Gamma_{\geq X}$) with budgets C(X)

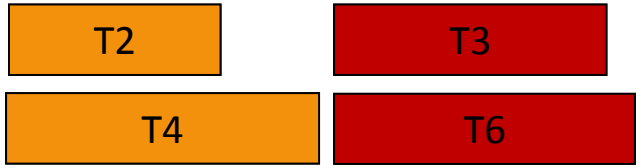
Mode X=1:

Fully-featured mode



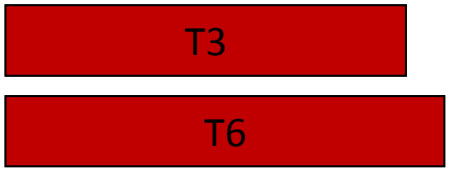
Mode X=2:

Important functionalities



Mode X=3:

Vital functionalities



Active

- **System starts in mode 1**

  Timing Failure Event → Mode change

Related works

- **Several mixed-criticality algorithms for multi-processor exist**
- **Standard real-time scheduling algorithms are adapted to MC model**

- **Problem:**

Algorithms	Schedulability performances	Number of Preemptions	Number of criticality levels managed
fpEDF-VD [Baruah14]	Average	Few	2
MC-Fluid [Lee14]	Best known	Many	2
MC^2[Mollison10]	Restricted	Few	5

Related works

- **Several mixed-criticality algorithms for multi-processor exist**
- **Standard real-time scheduling algorithms are adapted to MC model**

- **Problem:**

Algorithms	Schedulability performances	Number of Preemptions	Number of criticality levels managed
fpEDF-VD [Baruah14]	Average	Few	2
MC-Fluid [Lee14]	Best known	Many	2
MC^2[Mollison10]	Restricted	Few	5

Can we design a scheduling algorithm be good in all areas ?

Related works

- **Several mixed-criticality algorithms for multi-processor exist**
- **Standard real-time scheduling algorithms are adapted to MC model**

- **Problem:**

Algorithms	Schedulability performances	Number of Preemptions	Number of criticality levels managed
fpEDF-VD [Baruah14]	Average	Few	2
MC-Fluid [Lee14]	Best known	Many	2
MC^2[Mollison10]	Restricted	Few	5

Can we design a scheduling algorithm be good in all areas ?

- **We present our MC scheduling algorithm :**

GMC-RUN

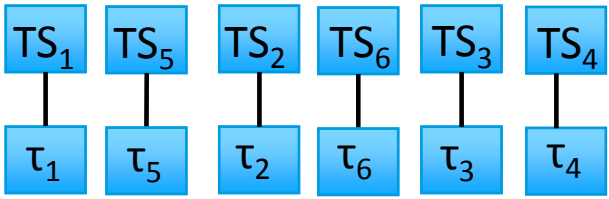
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors

τ_1 τ_5 τ_2 τ_6 τ_3 τ_4

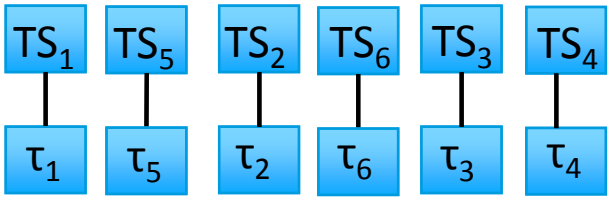
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors



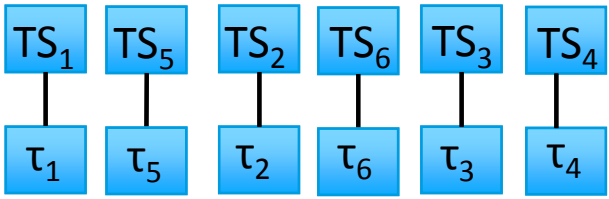
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**



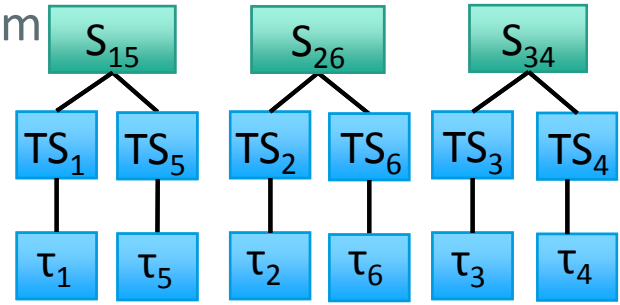
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server S = uniprocessor scheduling problem



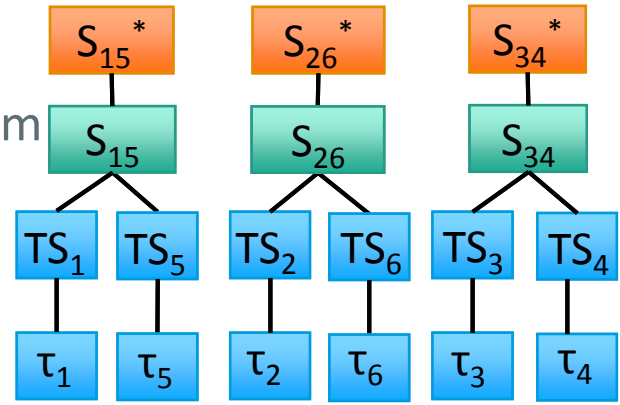
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server S = uniprocessor scheduling problem
     - Leaf primal servers schedule task servers



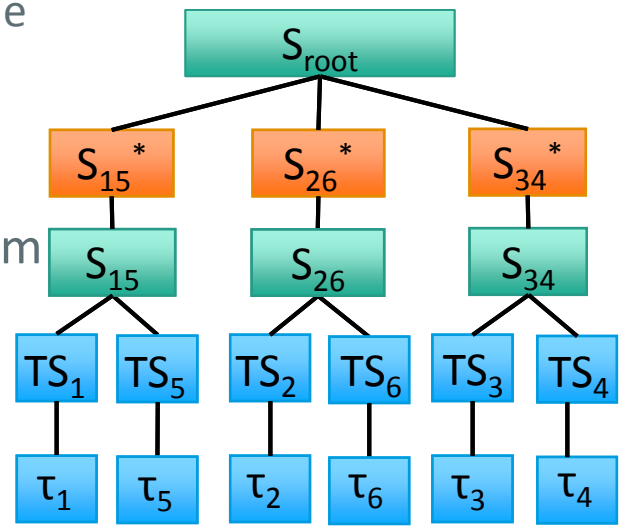
Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server S = uniprocessor scheduling problem
     - Leaf primal servers schedule task servers
  2. Dual server S^* = schedule slack time



Principles

- **RUN schedules periodic tasks with implicit deadlines**
- **Extensively use servers = task used to schedule tasks or other servers**
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server S = uniprocessor scheduling problem
     - Leaf primal servers schedule task servers
  2. Dual server S^* = schedule slack time
- **Hierarchy of servers is formed offline**

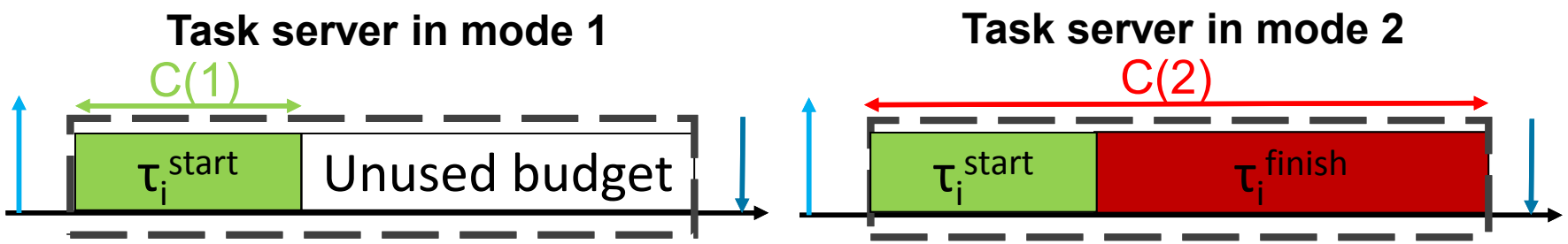


RUN model and MC model

- **Problem:**
  - One execution time budget per Task Server used in whole RUN structure
  - MC model several execution time budgets per task
- **Objective: leave task servers unchanged to reuse RUN**
  - Preserve RUN good schedulability performances
  - Preserve RUN properties concerning the number of pre-emptions
- **First consider systems with only two criticality levels**

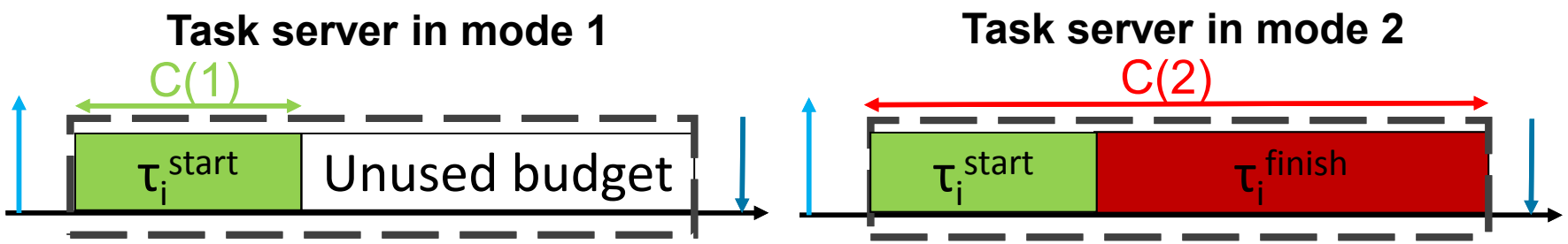
RUN model and MC model

- **Problem:**
  - One execution time budget per Task Server used in whole RUN structure
  - MC model several execution time budgets per task
- **Objective: leave task servers unchanged to reuse RUN**
  - Preserve RUN good schedulability performances
  - Preserve RUN properties concerning the number of pre-emptions
- **First consider systems with only two criticality levels**
- ↳ **Schedule CL-2 tasks using budgets of mode 2**



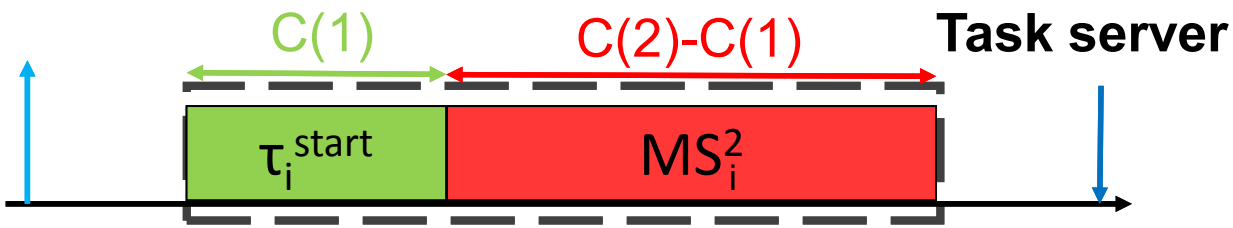
- **Problem:**
  - One execution time budget per Task Server used in whole RUN structure
  - MC model several execution time budgets per task
- **Objective: leave task servers unchanged to reuse RUN**
  - Preserve RUN good schedulability performances
  - Preserve RUN properties concerning the number of pre-emptions
- **First consider systems with only two criticality levels**

↳ **Schedule CL-2 tasks using budgets of mode 2**

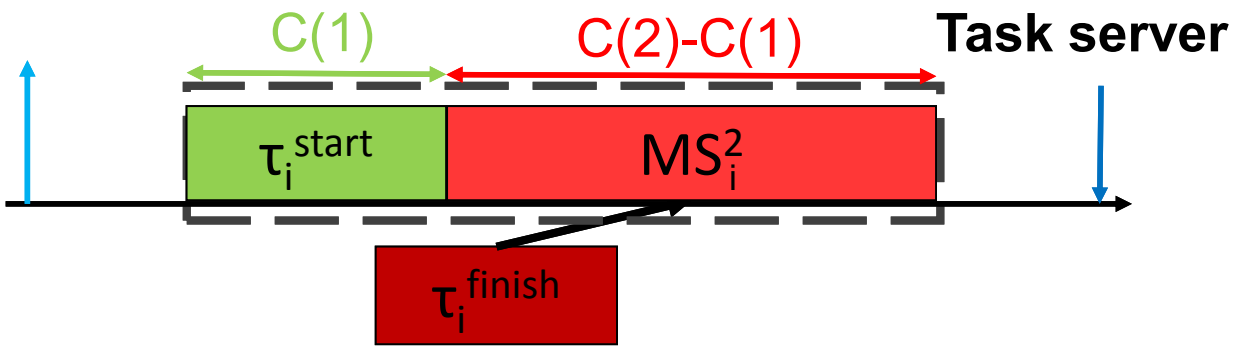


↳ **Introduce modal servers for CL-2 tasks**

- **Modal server MS_i for τ_i a CL-2 task :**



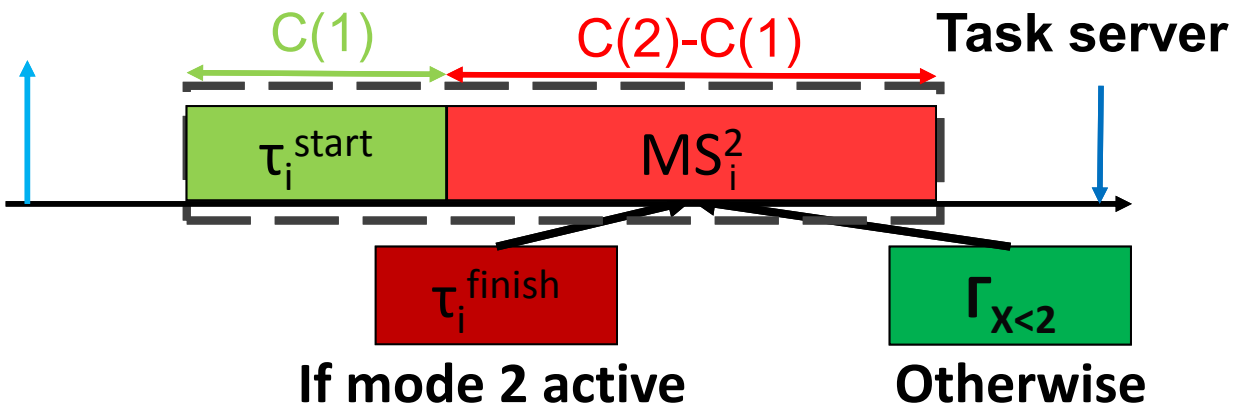
- **Modal server MS_i for τ_i a CL-2 task :**



If mode 2 active

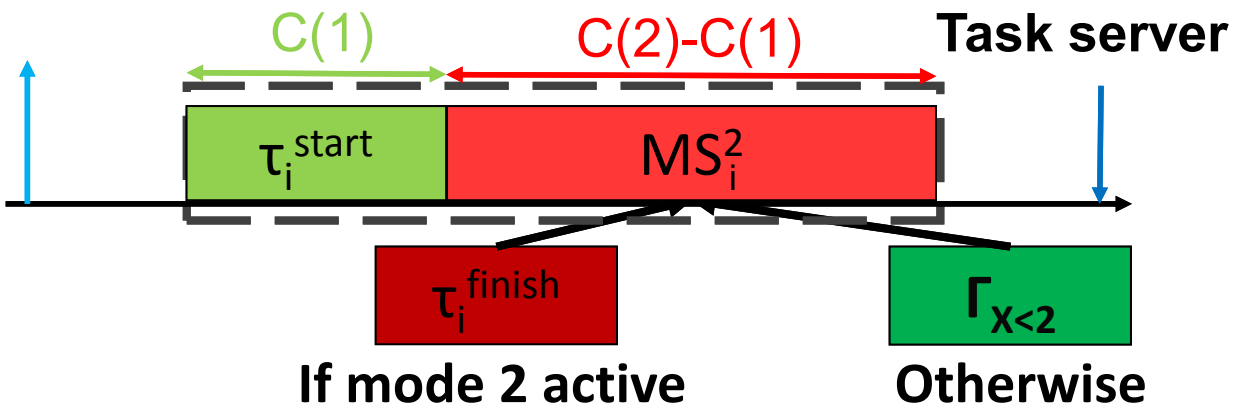
- **In mode 2: τ_i^{start} and τ_i^{finish} scheduled**

- **Modal server MS_i for τ_i a CL-2 task :**



- **In mode 2: τ_i^{start} and τ_i^{finish} scheduled**
- **In mode 1: τ_i^{start} and $\Gamma_{X<2}$ scheduled**

- **Modal server MS_i for τ_i a CL-2 task :**



- **In mode 2: τ_i^{start} and τ_i^{finish} scheduled**
- **In mode 1: τ_i^{start} and $\Gamma_{X<2}$ scheduled**

How to define $\Gamma_{X<2}$?

Allocation problem and schedulability conditions

- **Allocation problem: for each modal server, find the lower criticality task set that it can schedule correctly**
- **MS schedules correctly $\Gamma_{X<2}$ if one schedulability condition fulfilled**
  - SC 1 : if and only if MS period divides each $\Gamma_{X<2}$ task period,
    $U(MS) \geq U(\Gamma_{X<2})$
  - SC 2 (always applicable) :
    compare Demand Bound Function and Supply Bound Function
    on a hyperperiod (not trivial)
- **If no conditions fulfilled promote CL-1 task to a CL-2 one**
  - Mode 1: execution of the CL-1 task
  - Mode 2: nothing executed

Aggregated modal server

Gather several unused budgets

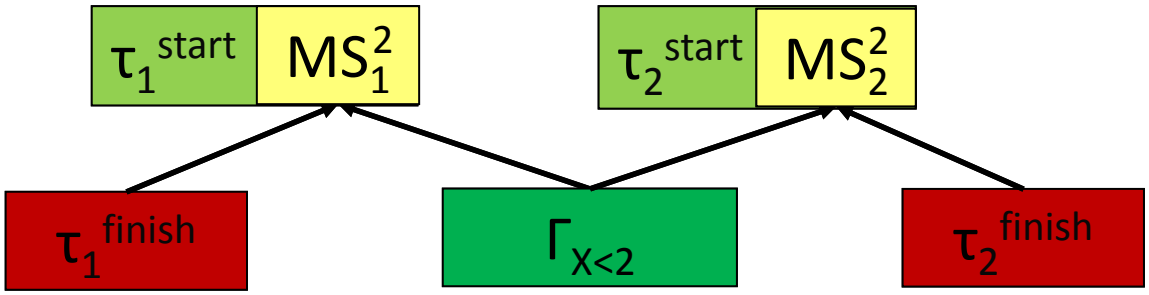
- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks

Aggregated modal server

Gather several unused budgets

- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks

➪ Gather modal server budgets to schedule same set of tasks



If Timing Failure Event **Otherwise** **If Timing Failure Event**

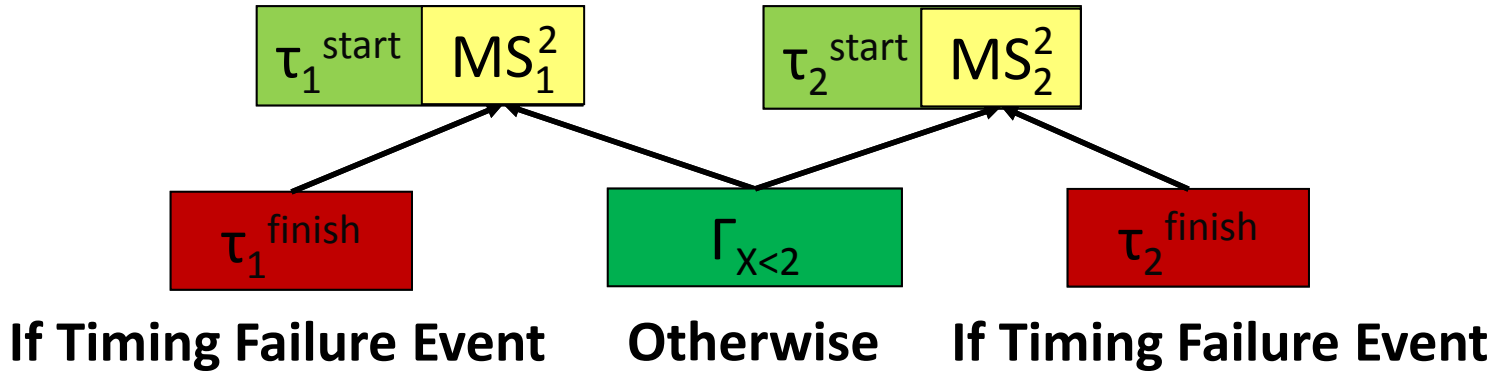
Aggregated modal server

Gather several unused budgets

- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks

⇨ Gather modal server budgets to schedule same set of tasks

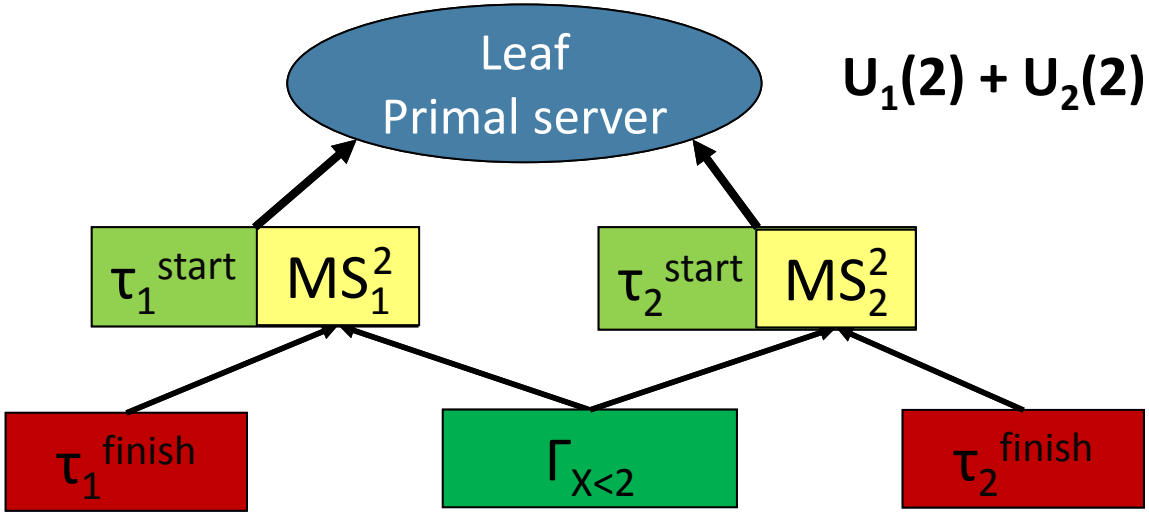
- Aggregation condition: execute sequentially modal servers



Aggregated modal server

Gather several unused budgets

- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks
- ↳ Gather modal server budgets to schedule same set of tasks
- Aggregation condition: execute sequentially modal servers
- ↳ Schedule corresponding task servers in a same leaf primal server



$U_1(2) + U_2(2) \leq 1$

If Timing Failure Event | Otherwise | If Timing Failure Event

Solving the allocation problem

- **Best allocation: allocation requiring the fewest number of processors**
- **Finding the best allocation is a hard optimization problem**

1. **Many allocations possible**
   - Many possible aggregated modal servers
   - Many possible task allocation in modal servers
2. **Assessing schedulability conditions not trivial**
   - Schedulability conditions are non-linear (SBF/DBF) and two possible tests
   - Aggregated modal servers prevent handling task allocation in isolation

- **Our choice: evolutionary algorithm for its flexibility**

1. **Find a representation to describe all possible allocations:**
   - Implemented as matrix
   - Presented as tree manipulation

Evolutionary algorithm

1. **Find a representation to describe all possible allocations:**
   - Implemented as matrix
   - Presented as tree manipulation
2. **Define operators to explore the set of solutions**
   - Mutation: (de-)allocation of a task to a modal server
   - Crossover: form two new allocations by mixing two existing ones

Evolutionary algorithm

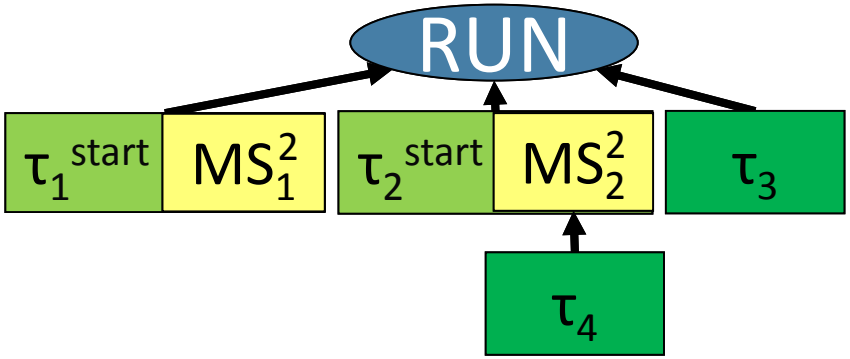
1. **Find a representation to describe all possible allocations:**
   - Implemented as matrix
   - Presented as tree manipulation
2. **Define operators to explore the set of solutions**
   - Mutation: (de-)allocation of a task to a modal server
   - Crossover: form two new allocations by mixing two existing ones
3. **Evaluate allocations and conserve only best ones**

Mutation operator

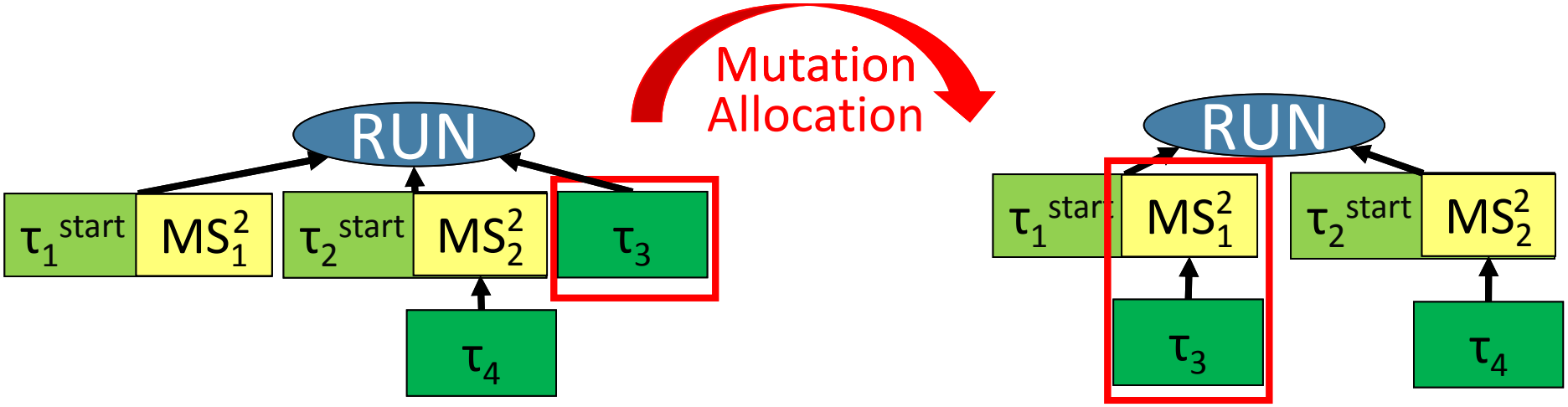
- **Mutation: (de-)allocation of a task to a modal server**

Mutation operator

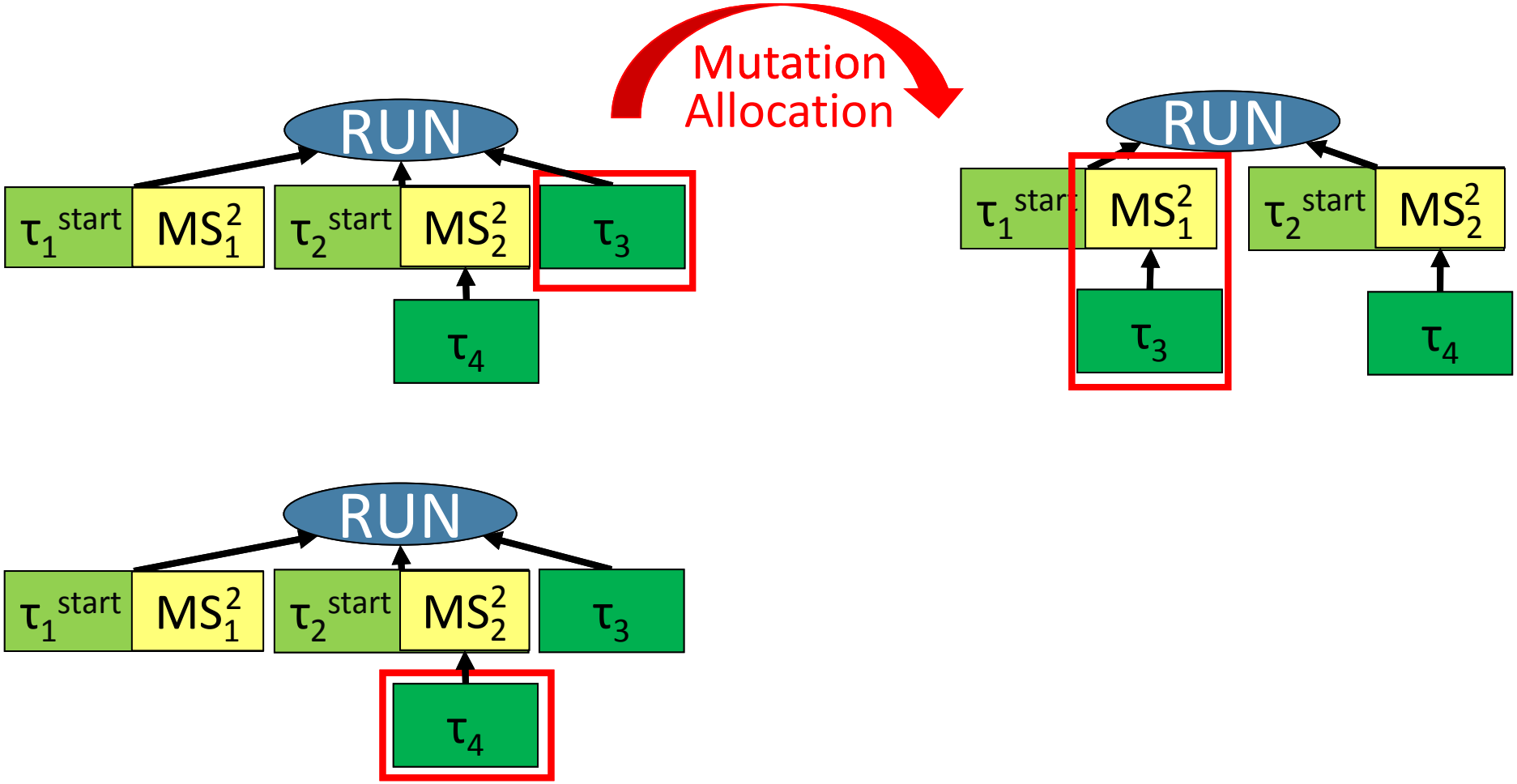
- **Mutation: (de-)allocation of a task to a modal server**



- **Mutation: (de-)allocation of a task to a modal server**

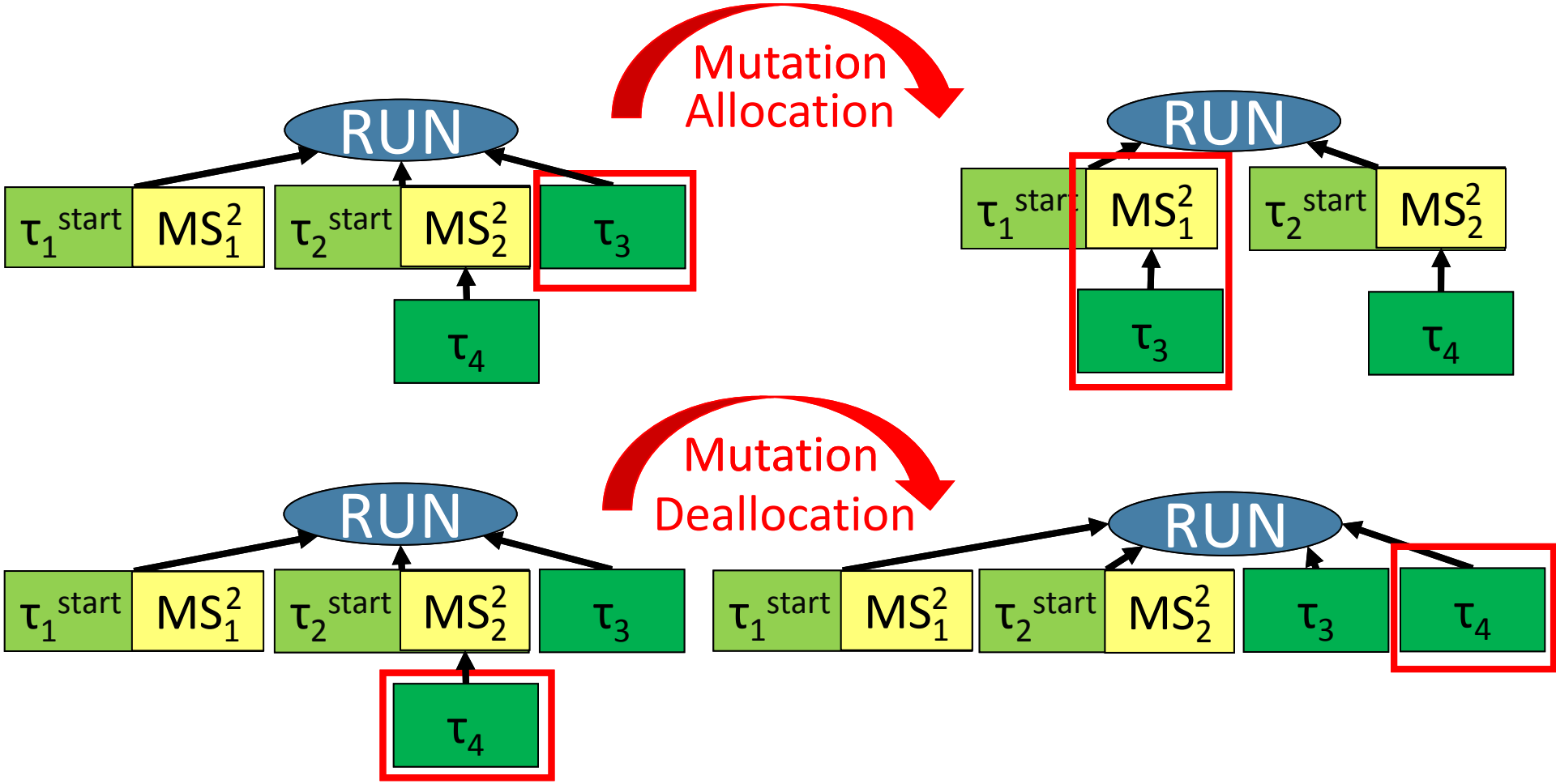


- **Mutation: (de-)allocation of a task to a modal server**

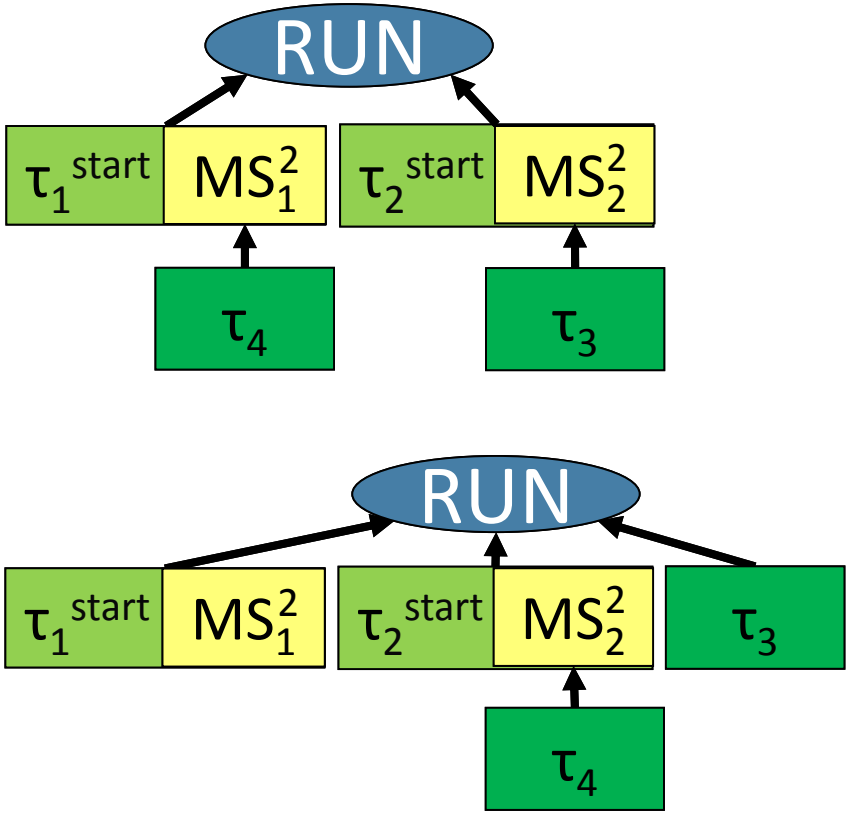


Mutation operator

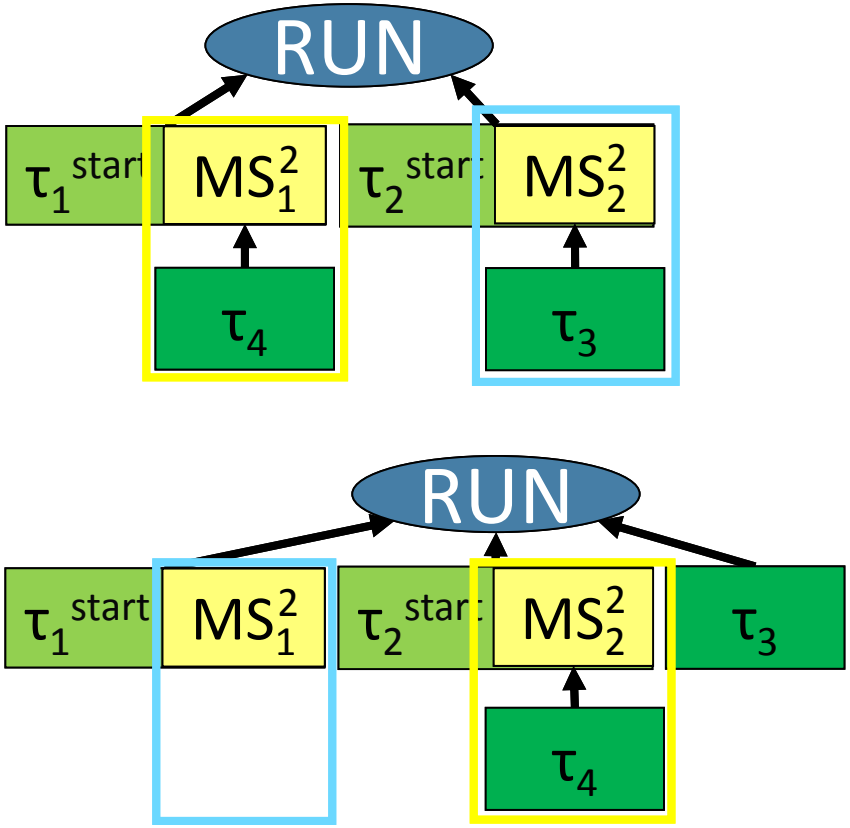
- **Mutation: (de-)allocation of a task to a modal server**



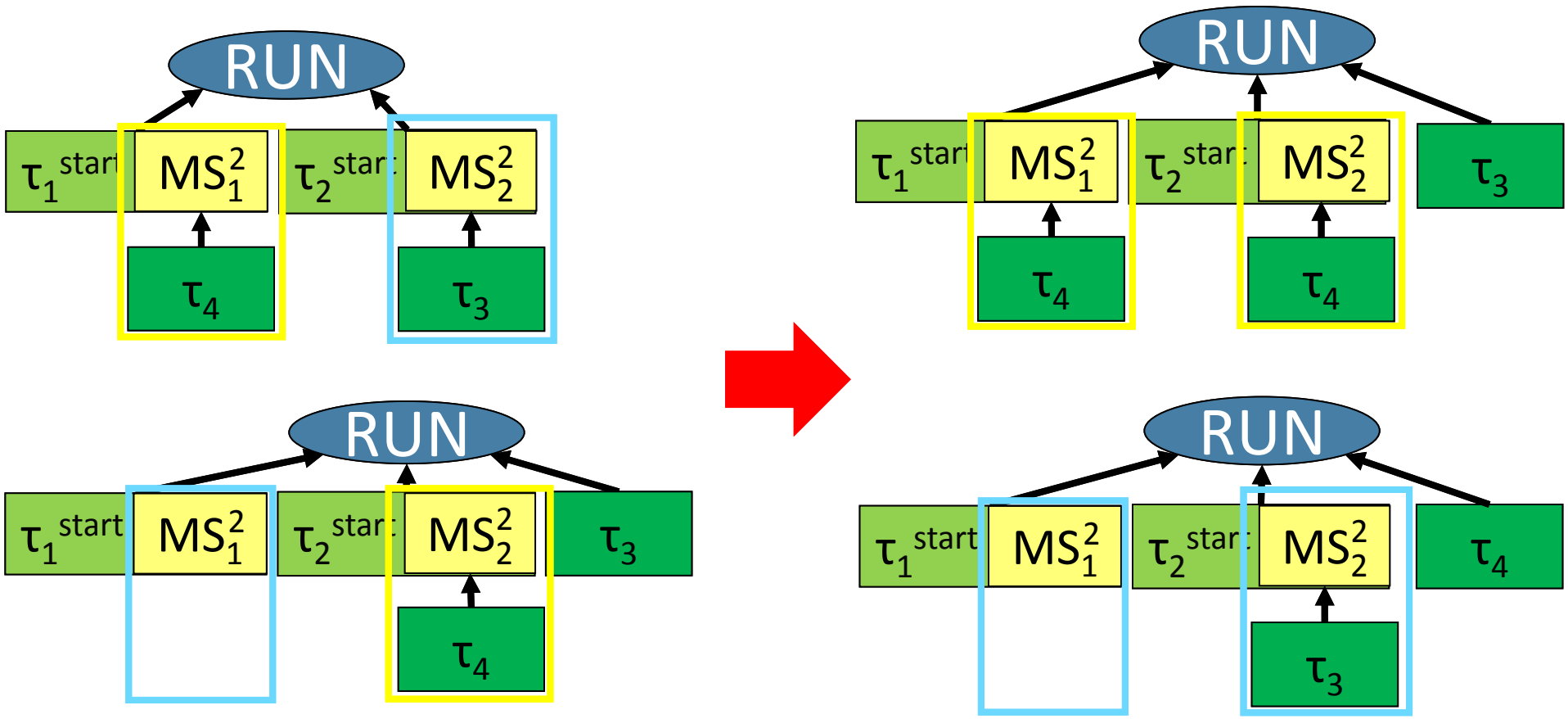
- **Crossover: form two new allocations by mixing two existing ones**



- **Crossover: form two new allocations by mixing two existing ones**



- **Crossover: form two new allocations by mixing two existing ones**

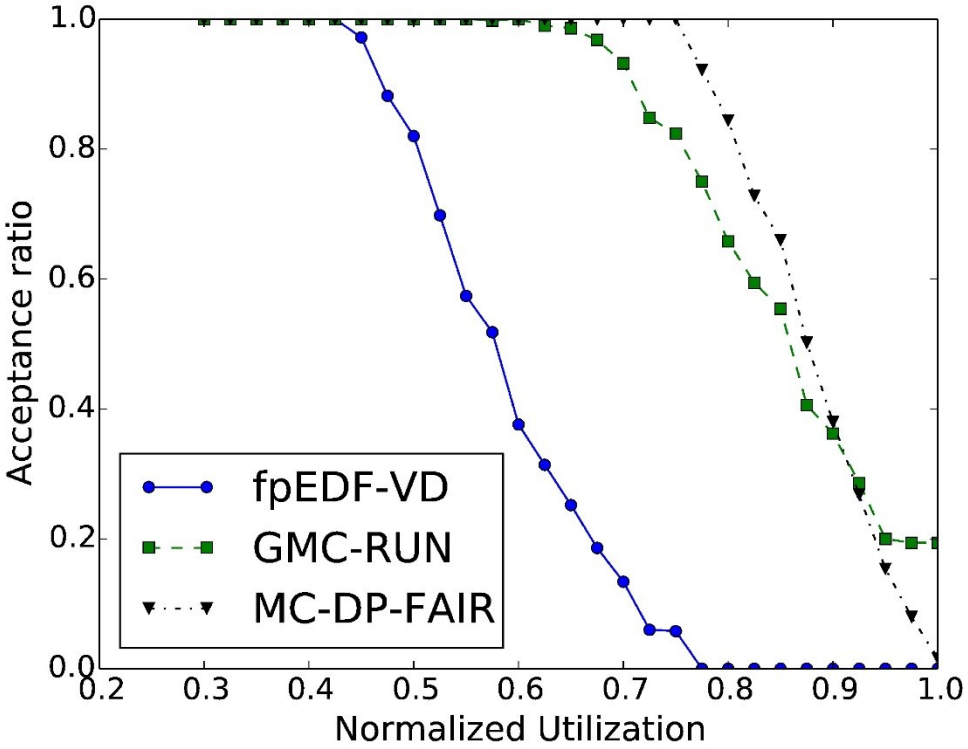


Two criticality levels system : schedulability performances

- X-axis: task set utilization normalized by the number of CPU (here 2)
- Y-axis : acceptance ratio for 500 task sets

➪ GMC-RUN close to MC-DP-Fair

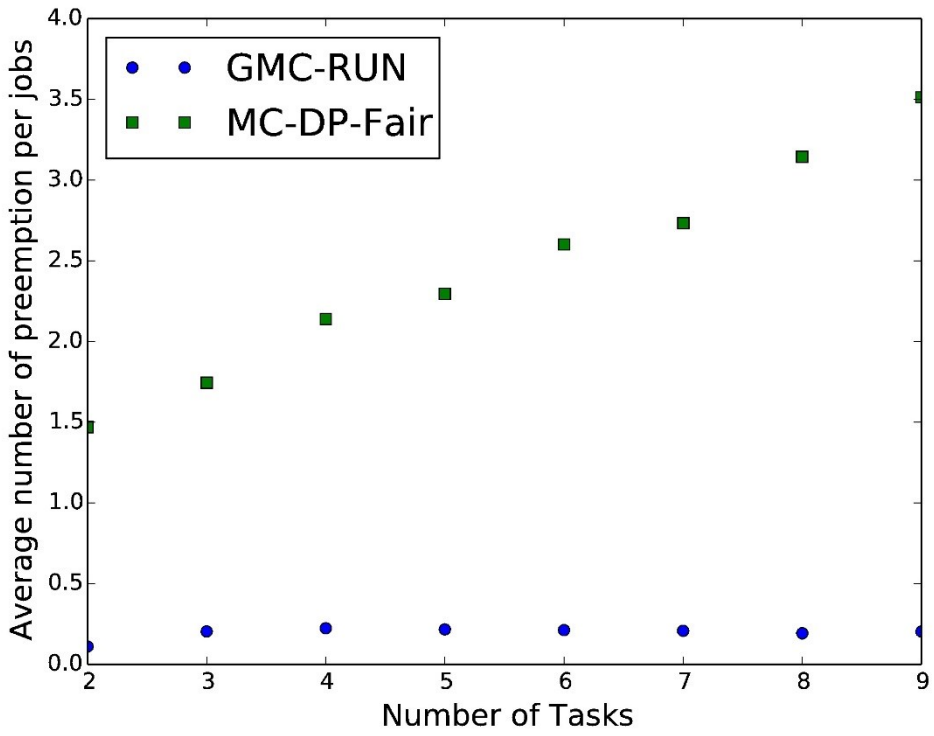
➪ fpEDF-VD outperformed



Experimental assessment

Two criticality levels system : number of pre-emptions

- **Consider only task sets schedulable by GMC-RUN and MC-DP-Fair**
- ↳ **RUN pre-emption performances preserved**
- ↳ **GMC-RUN outperforms MC-DP-Fair**



Overview of the generalization

- **Actual systems are generally composed of at least 4 criticality levels**

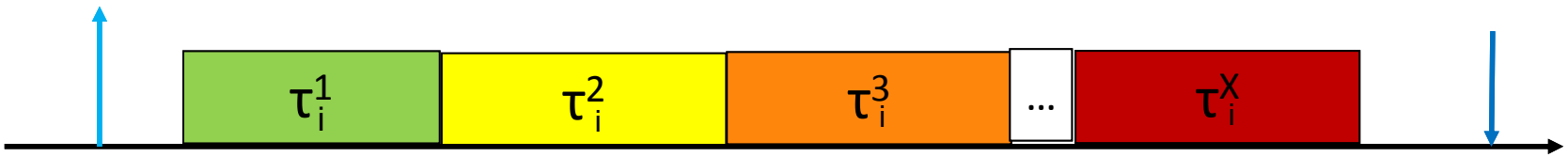
- **Approach to schedule systems with <u>2 criticality levels</u> :**
  - **Schedule CL-1 task into CL-2 task unused budgets**
  - **Or promote CL-1 task to a CL-2 task one**

  - **Remove the lowest criticality level: criticality level reduction**

- **Approach to deal with <u>n criticality levels</u> :**
  - **Recursively apply criticality level reduction from lowest to highest criticality levels**

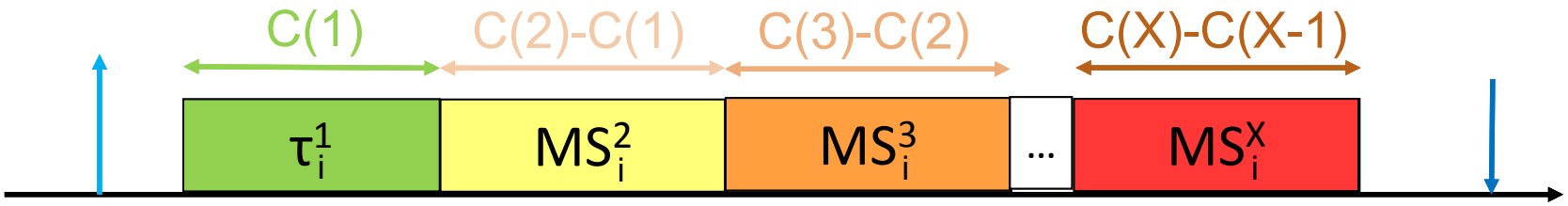
Generalization rules: modal servers

- **Tasks τ_i of criticality X > 2 has X subtasks:**



Generalization rules: modal servers

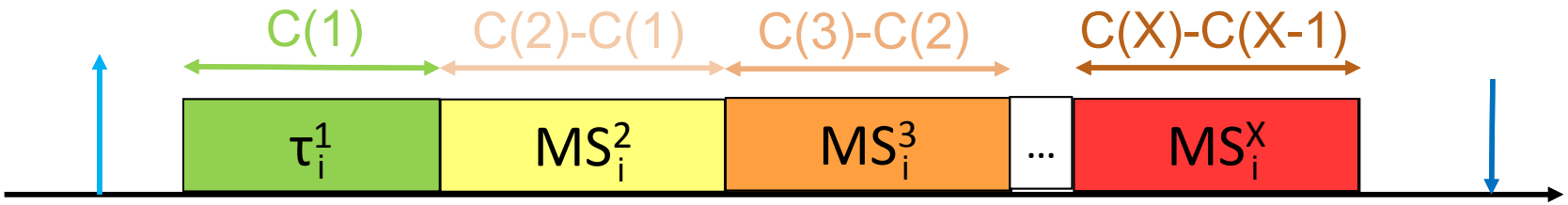
- **Tasks τ_i of criticality X > 2 has X subtasks:**



- **Each subtask τ_i^L can be scheduled by a modal server MS^L**
  - Completes τ_i execution when mode L or higher is active
  - Schedules tasks of criticality < L, otherwise

Generalization rules: modal servers

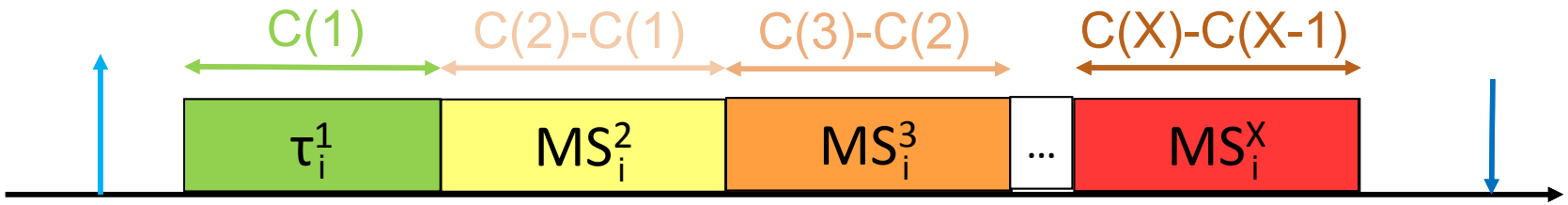
- **Tasks τ_i of criticality X > 2 has X subtasks:**



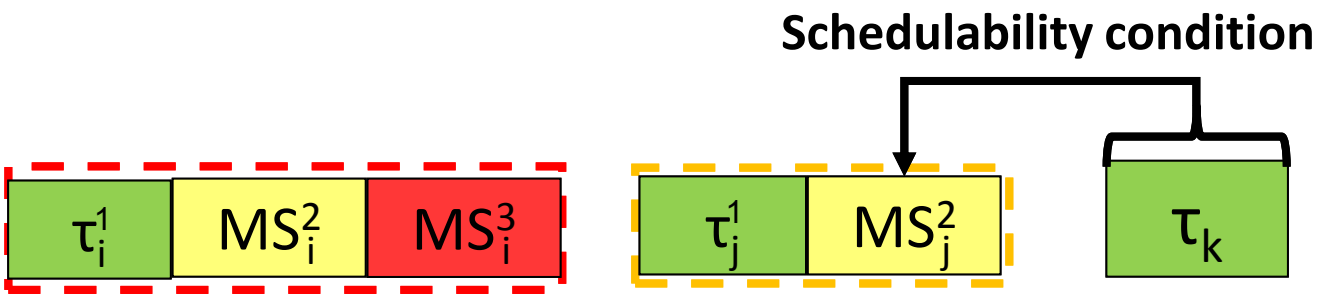
- **Each subtask τ_i^L can be scheduled by a modal server MS^L**
  - Completes τ_i execution when mode L or higher is active
  - Schedules tasks of criticality < L, otherwise

Generalization rules: modal servers

- **Tasks τ_i of criticality X > 2 has X subtasks:**

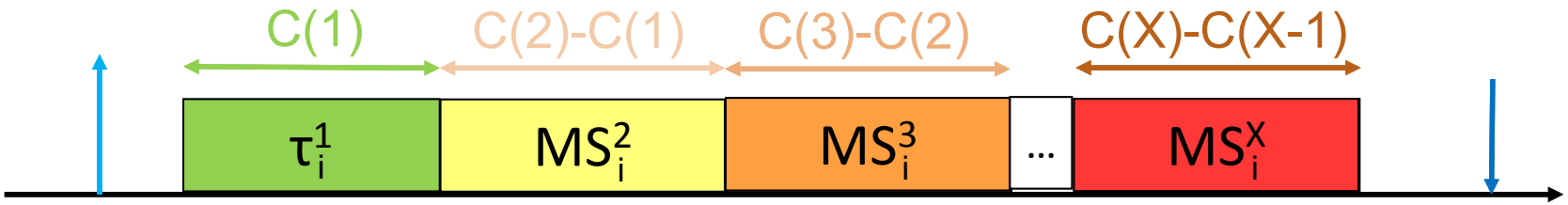


- **Each subtask τ_i^L can be scheduled by a modal server MS^L**
  - Completes τ_i execution when mode L or higher is active
  - Schedules tasks of criticality < L, otherwise

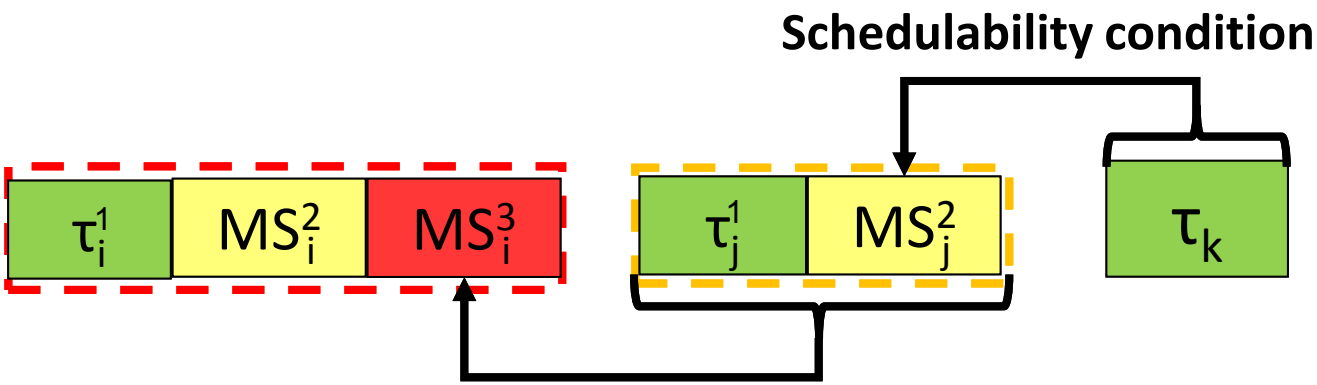


Generalization rules: modal servers

- **Tasks τ_i of criticality X > 2 has X subtasks:**

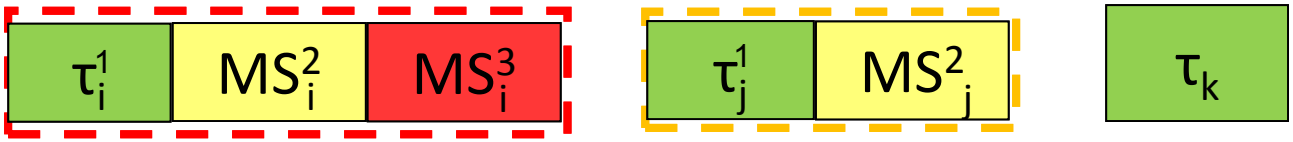


- **Each subtask τ_i^L can be scheduled by a modal server MS^L**
  - Completes τ_i execution when mode L or higher is active
  - Schedules tasks of criticality < L, otherwise



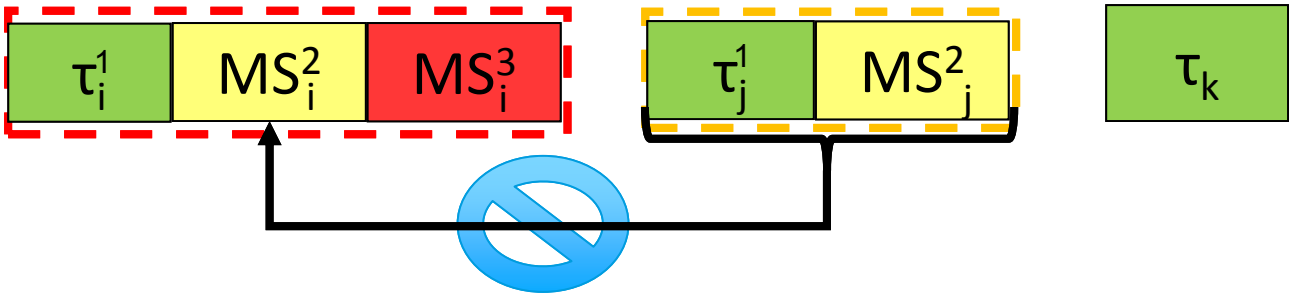
Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise



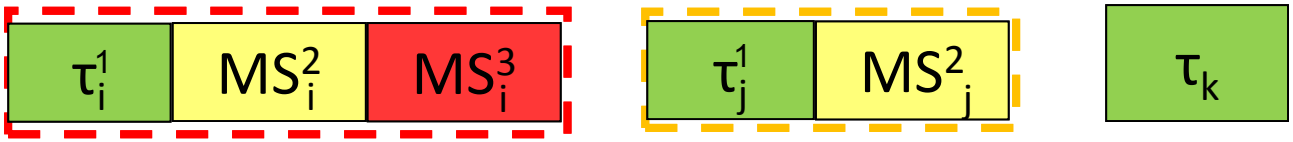
Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise



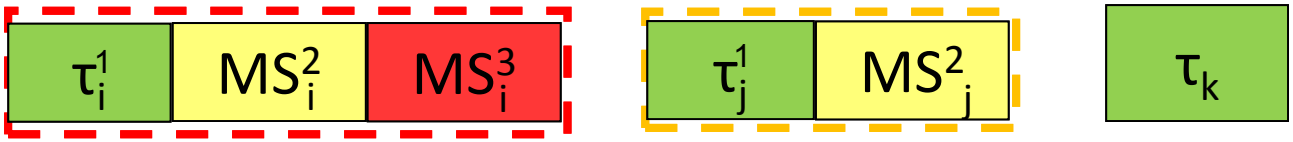
Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise



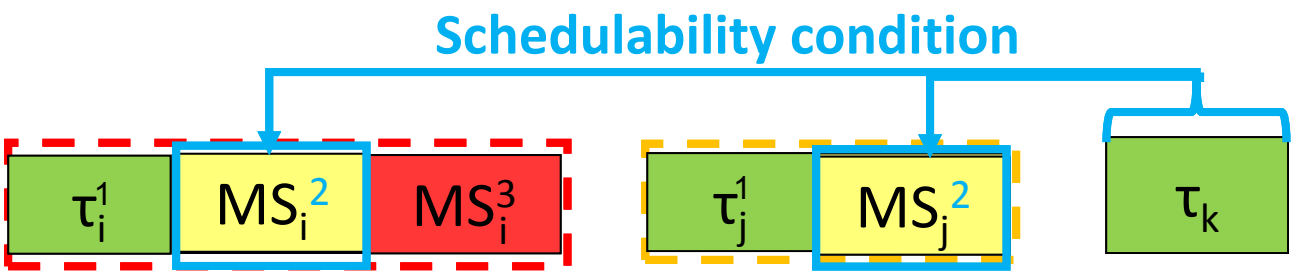
Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise
- **Aggregate MS when they fit in a primal leaf server**



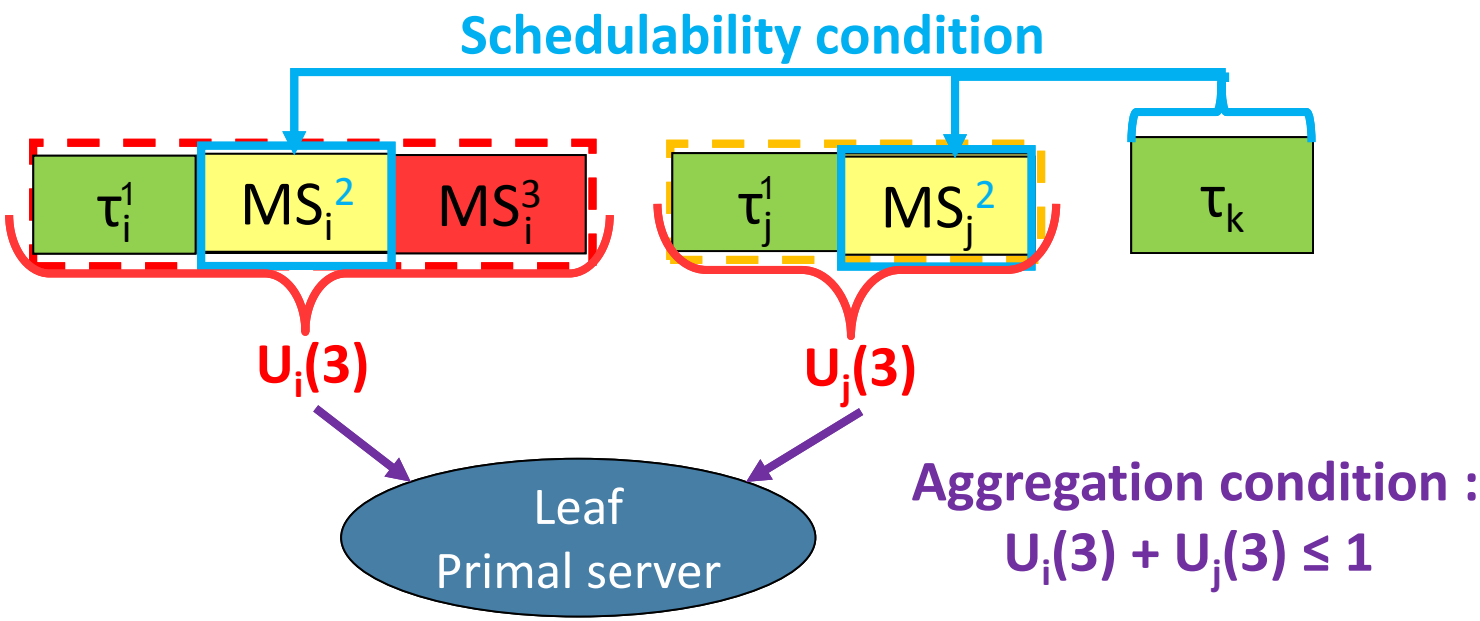
Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise
- **Aggregate MS when they fit in a primal leaf server**
- **Schedulability condition: use timing parameters of MS and task at its CL**



Generalization rules: modal servers

- **Each modal server MS^L**
  - either completes τ_i execution when mode L or higher is active
  - or schedules tasks of criticality < L, otherwise
- **Aggregate MS when they fit in a primal leaf server**
- **Schedulability condition: use timing parameters of MS and task at its CL**
- **Aggregation condition: sum TS utilizations at max CL (here CL-3)**



Aggregation condition :
$U_i(3) + U_j(3) \leq 1$

Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

Worst case allocation : promoting task criticality only

- **Worst case allocation: no allocation possible**

Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

- **Worst case allocation: no allocation possible**
- ↳ **Promote task criticality level**

Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- ↳ **Promote task criticality level**

Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- ↳ **Promote task criticality level**

Number of criticality levels: 3

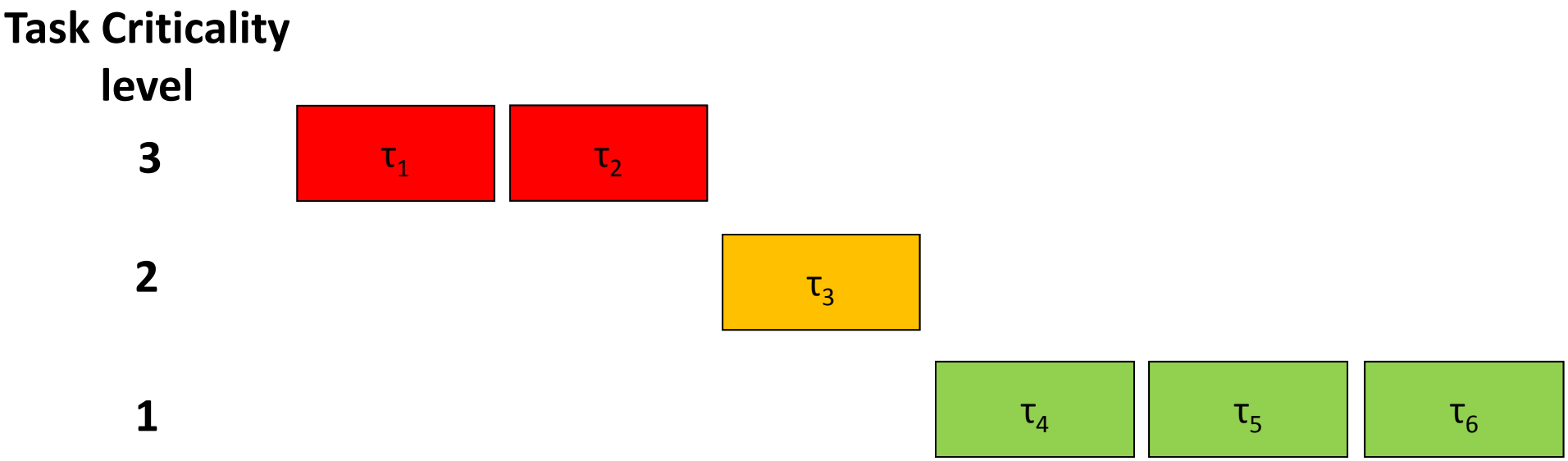
Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- **Promote task criticality level**

Number of criticality levels: 3



Scheduling more than two criticality levels

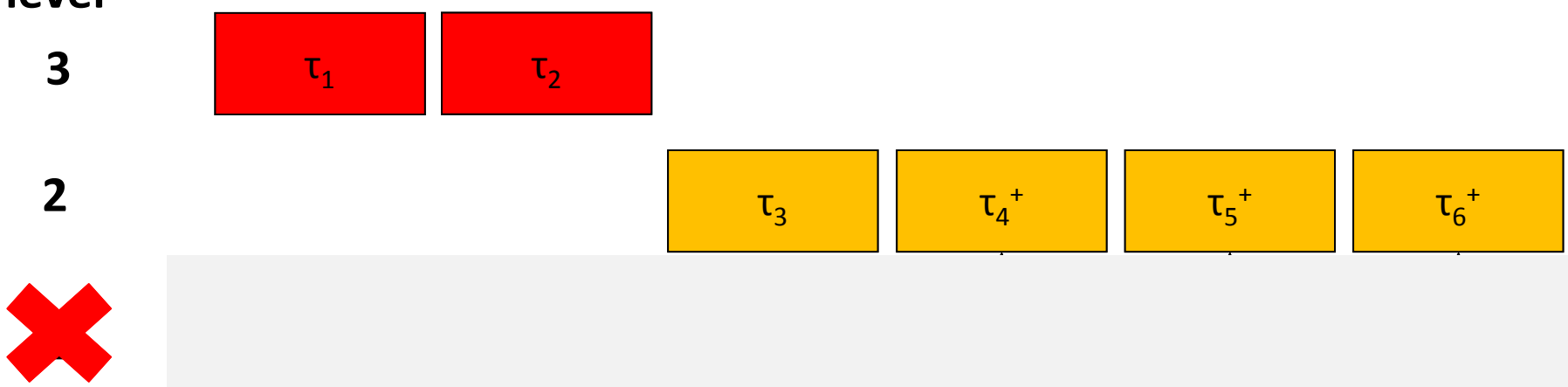
Worst case allocation : promoting task criticality only

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- **Promote task criticality level**

Number of criticality levels: 2

Task Criticality level



Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

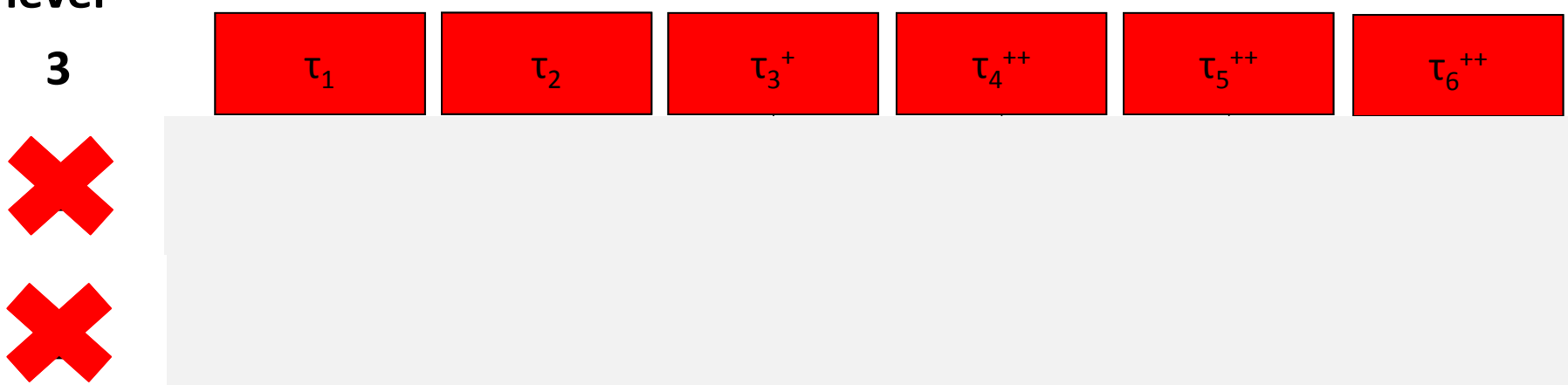
Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- **Promote task criticality level**

Number of criticality levels: 1

Task Criticality level

3



Scheduling more than two criticality levels

Worst case allocation : promoting task criticality only

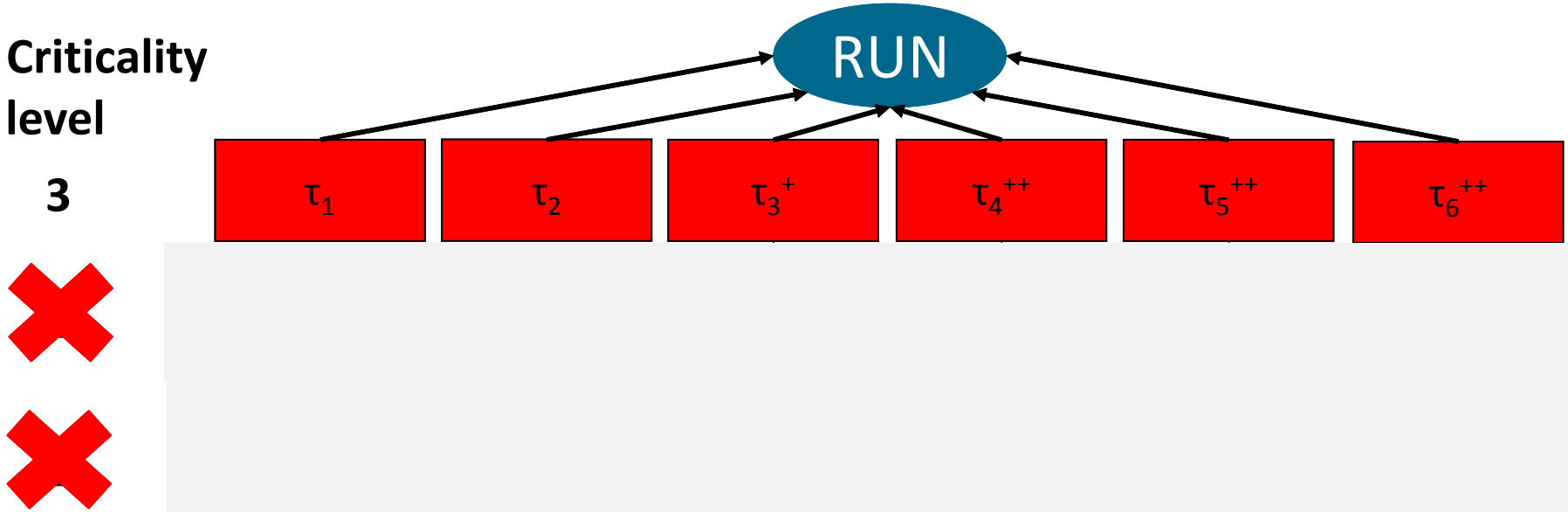
Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
- **Promote task criticality level**

Number of criticality levels: 1

Task Criticality level

3



Scheduling more than two criticality levels

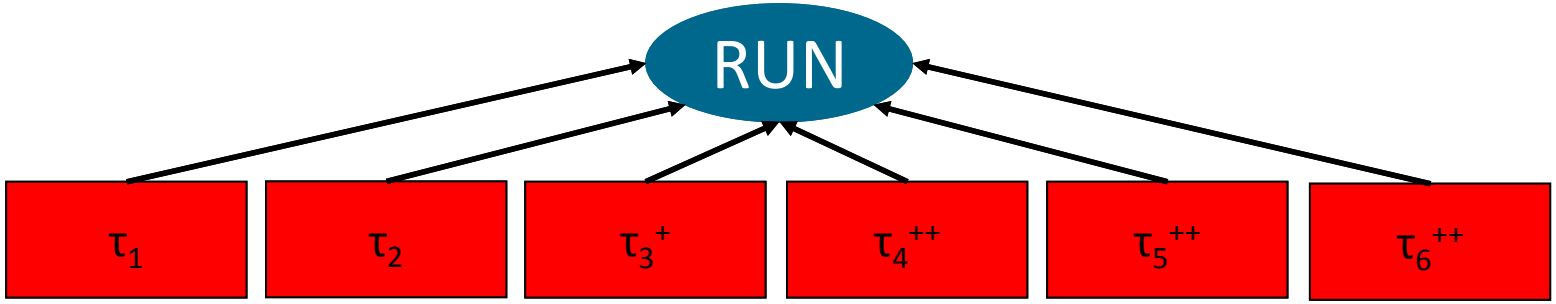
Worst case allocation : promoting task criticality only

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **Worst case allocation: no allocation possible**
  - **Promote task criticality level**
  - **3 processors required**
  - **Only promoting ensures a valid but inefficient scheduling**

Task Criticality level

3



Feasible allocation with better performances

Reduction of criticality level 1

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

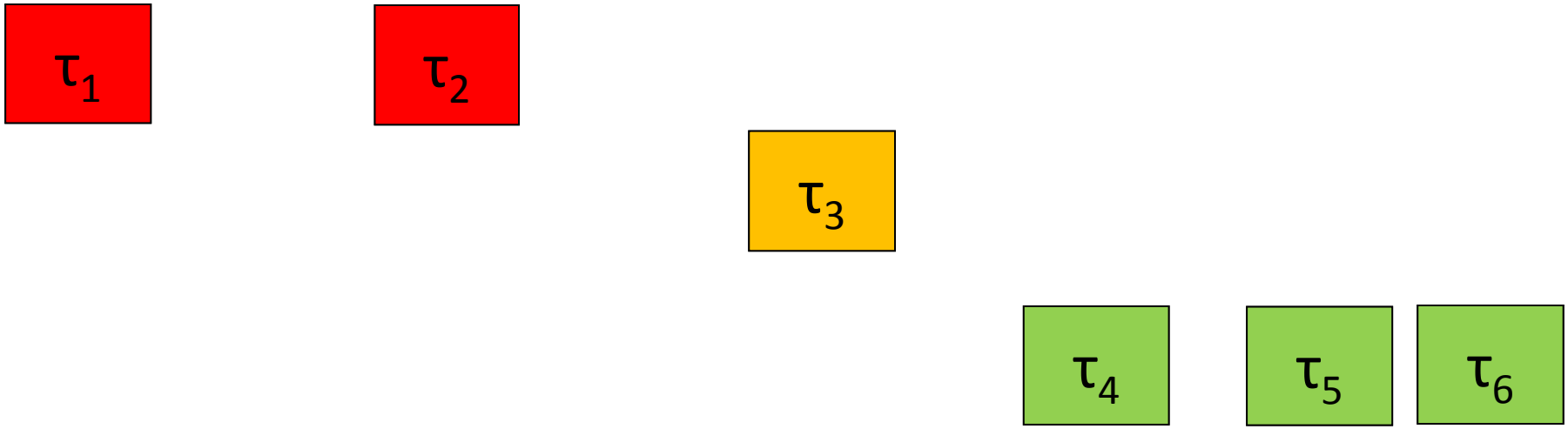
- **More realistic case: some allocations are possible**

Feasible allocation with better performances

Reduction of criticality level 1

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

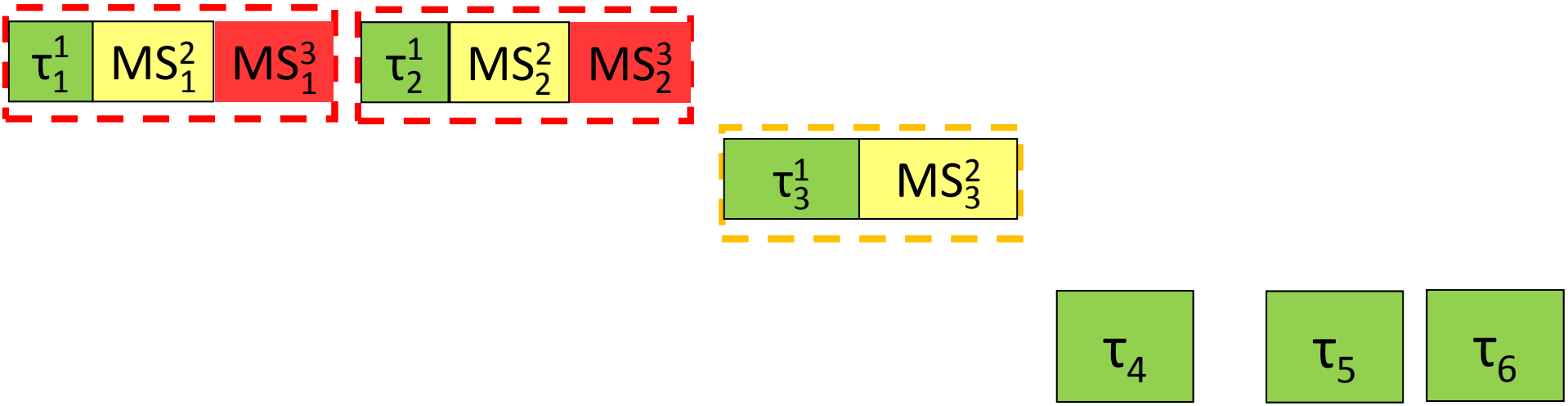


Feasible allocation with better performances

Reduction of criticality level 1

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

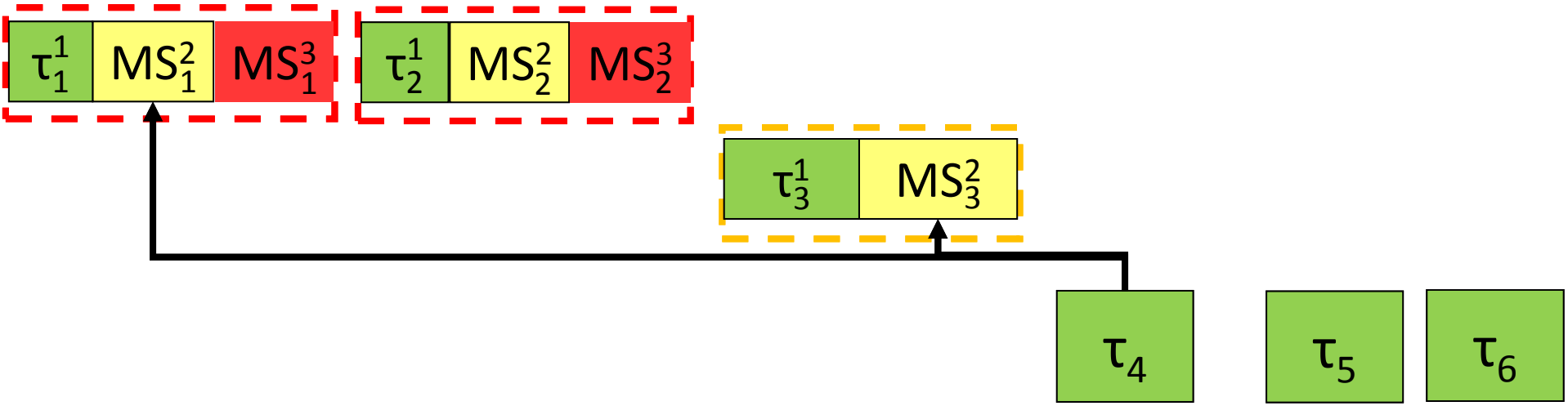


Feasible allocation with better performances

Reduction of criticality level 1

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

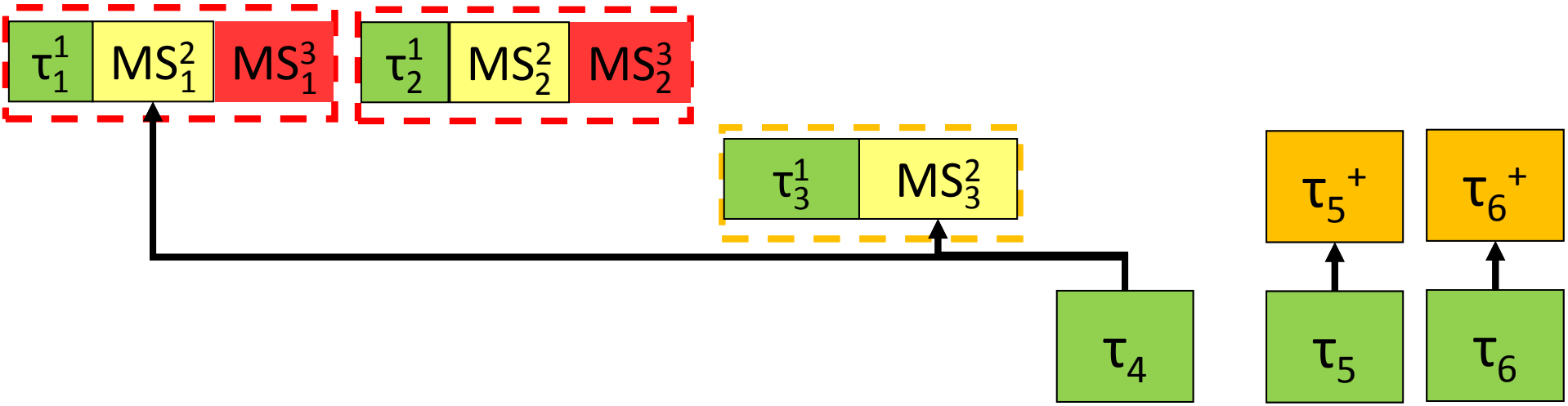


Feasible allocation with better performances

Reduction of criticality level 1

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_4	10	1	0.3	0.3	0.3
τ_5	20	1	0.3	0.3	0.3
τ_6	40	1	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

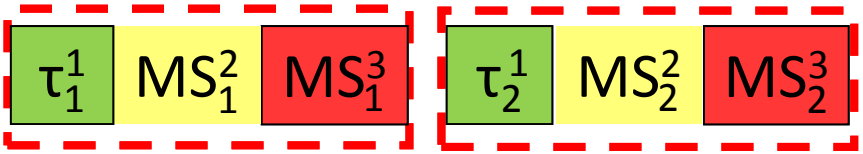


Feasible allocation with better performances

Reduction of criticality level 2

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_5^+	20	2	0.3	0.3	0.3
τ_6^+	40	2	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

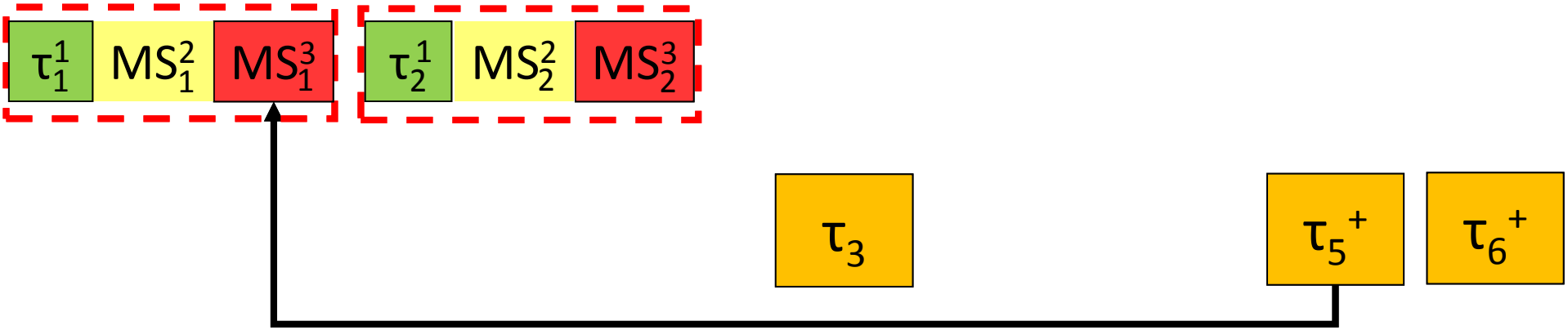


Feasible allocation with better performances

Reduction of criticality level 2

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_5^+	20	2	0.3	0.3	0.3
τ_6^+	40	2	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

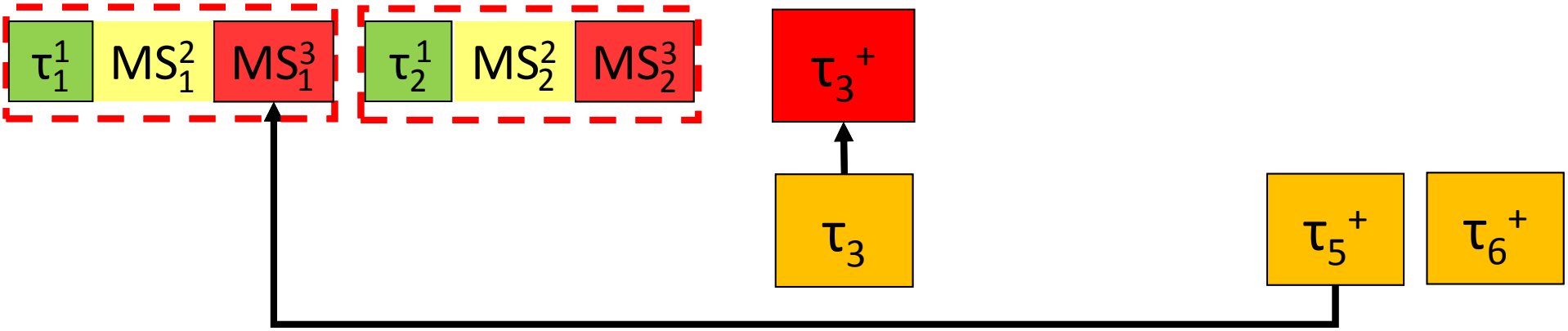


Feasible allocation with better performances

Reduction of criticality level 2

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_5^+	20	2	0.3	0.3	0.3
τ_6^+	40	2	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

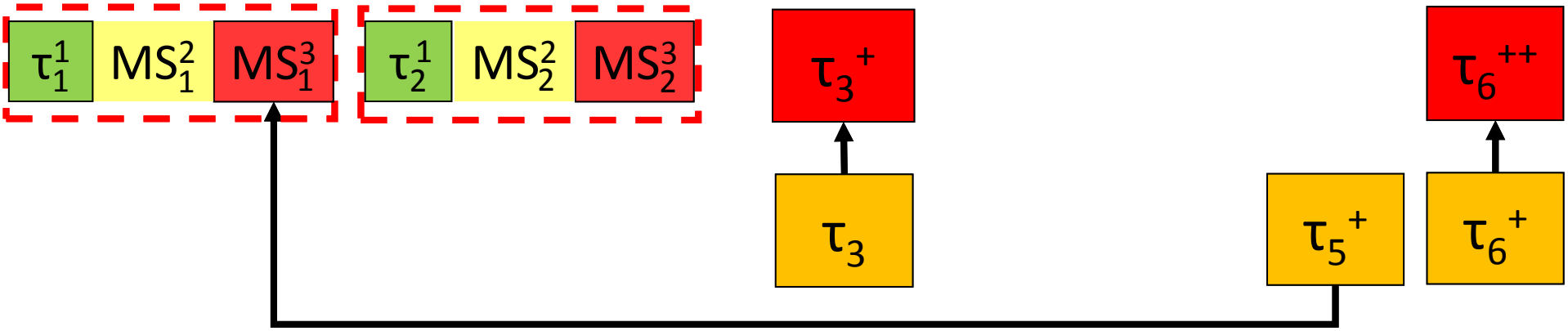


Feasible allocation with better performances

Reduction of criticality level 2

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3	5	2	0.1	0.2	0.2
τ_5^+	20	2	0.3	0.3	0.3
τ_6^+	40	2	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

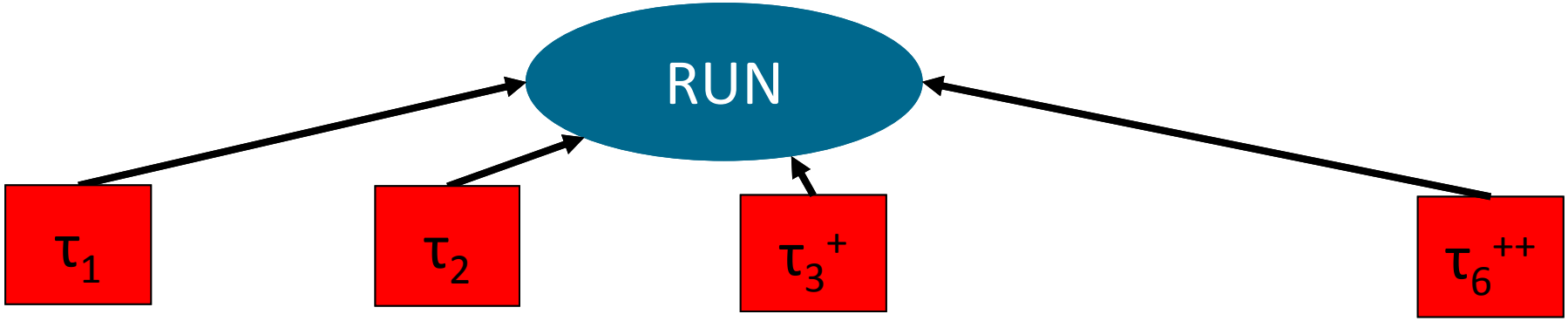


Feasible allocation with better performances

Final result

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3^+	5	3	0.1	0.2	0.2
τ_6^{++}	40	3	0.4	0.4	0.4

- **More realistic case: some allocations are possible**

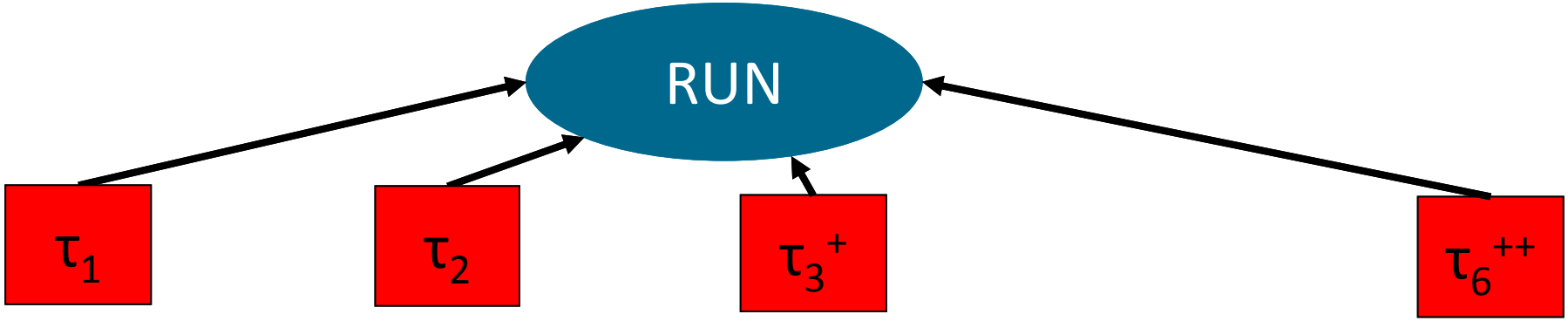


Feasible allocation with better performances

Final result

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3^+	5	3	0.1	0.2	0.2
τ_6^{++}	40	3	0.4	0.4	0.4

- More realistic case: some allocations are possible
- Task set schedulable on:
  - 2 processors with GMC-RUN
  - One processor saved

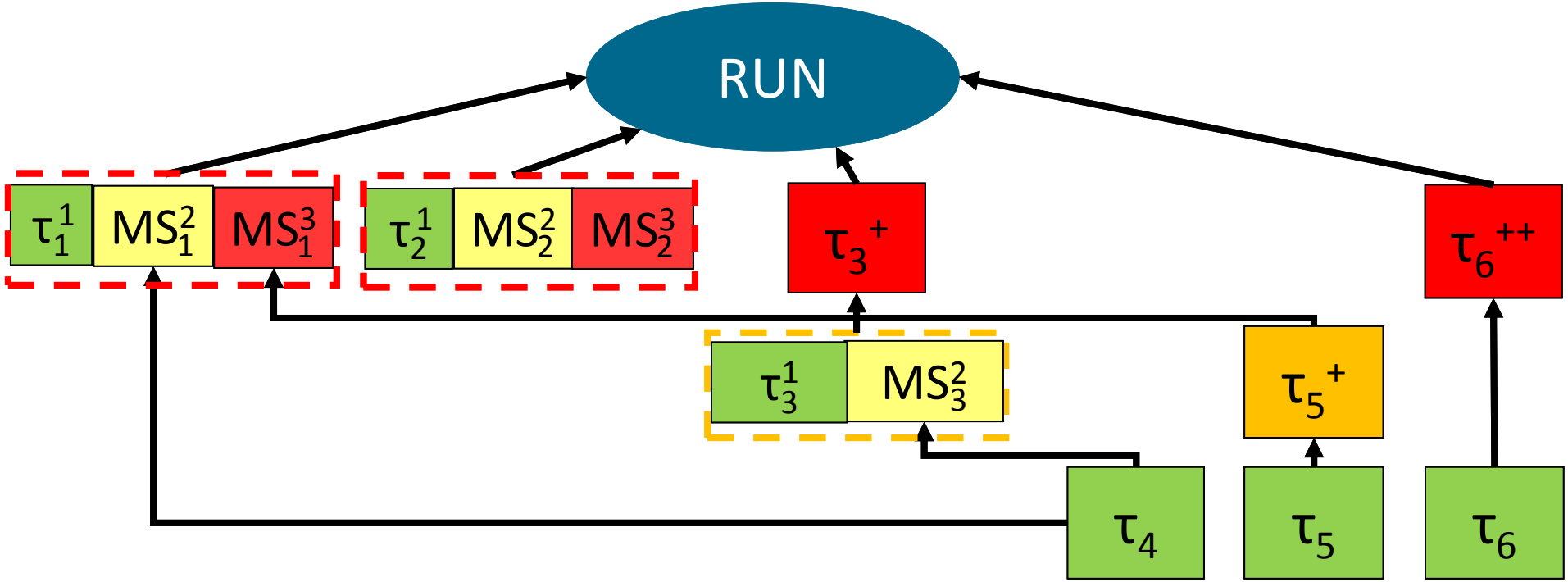


Feasible allocation with better performances

Final result

Task	Period	Criticality	U(1)	U(2)	U(3)
τ_1	5	3	0.2	0.4	0.7
τ_2	5	3	0.2	0.3	0.5
τ_3^+	5	3	0.1	0.2	0.2
τ_6^{++}	40	3	0.4	0.4	0.4

- More realistic case: some allocations are possible
- Task set schedulable on:
  - 2 processors with GMC-RUN
  - One processor saved



- **GMC-RUN:**
  - Handles more than two criticality levels
  - RUN working preserved: pre-emption performance preserved
  - Performs better than fpEDF-VD and is close to MC-Fluid

- **Future work:**
  - Experiments on systems with more than two criticality levels
  - Implementation in Litmus^{RT}

Thank You !

References

- [Baruah14] S. Baruah, B. Chattopadhyay, H. Li, and I. Shin, “Mixed-criticality scheduling on multiprocessors,” *Real-Time Systems*, vol. 50, no. 1, pp.142–177, 2014.
- [Burns] A. Burns and R. Davis, “Mixed criticality systems: a review.” http://www-users.cs.york.ac.uk/burns/review.pdf
- [Lee14] J. Lee, K.-M. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, and I. Lee, “Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors,” in *RTSS’2014*, Dec 2014, pp. 41–52.
- [Mollison10] M. Mollison, J. Erickson, J. Anderson, S. Baruah, and J. Scoredos. Mixed-criticality real-time scheduling for multicore systems, in *ICESS’2010*, June 2010, pp. 1864-1871.
- [Regnier11] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, “Run: Optimal multiprocessor real-time scheduling via reduction to uniprocessor,” in *RTSS’2011*, Nov 2011, pp. 104–115.