A Compositional Analysis Framework for Hierarchical and Partitioned Real -Time Systems

Insup Lee
PRECISE Center
Department of Computer and Information Science
University of Pennsylvania

June 2, 2009

AFOSR FA9550-07-1-0216 (PM: Dr. David Luginbuhl)

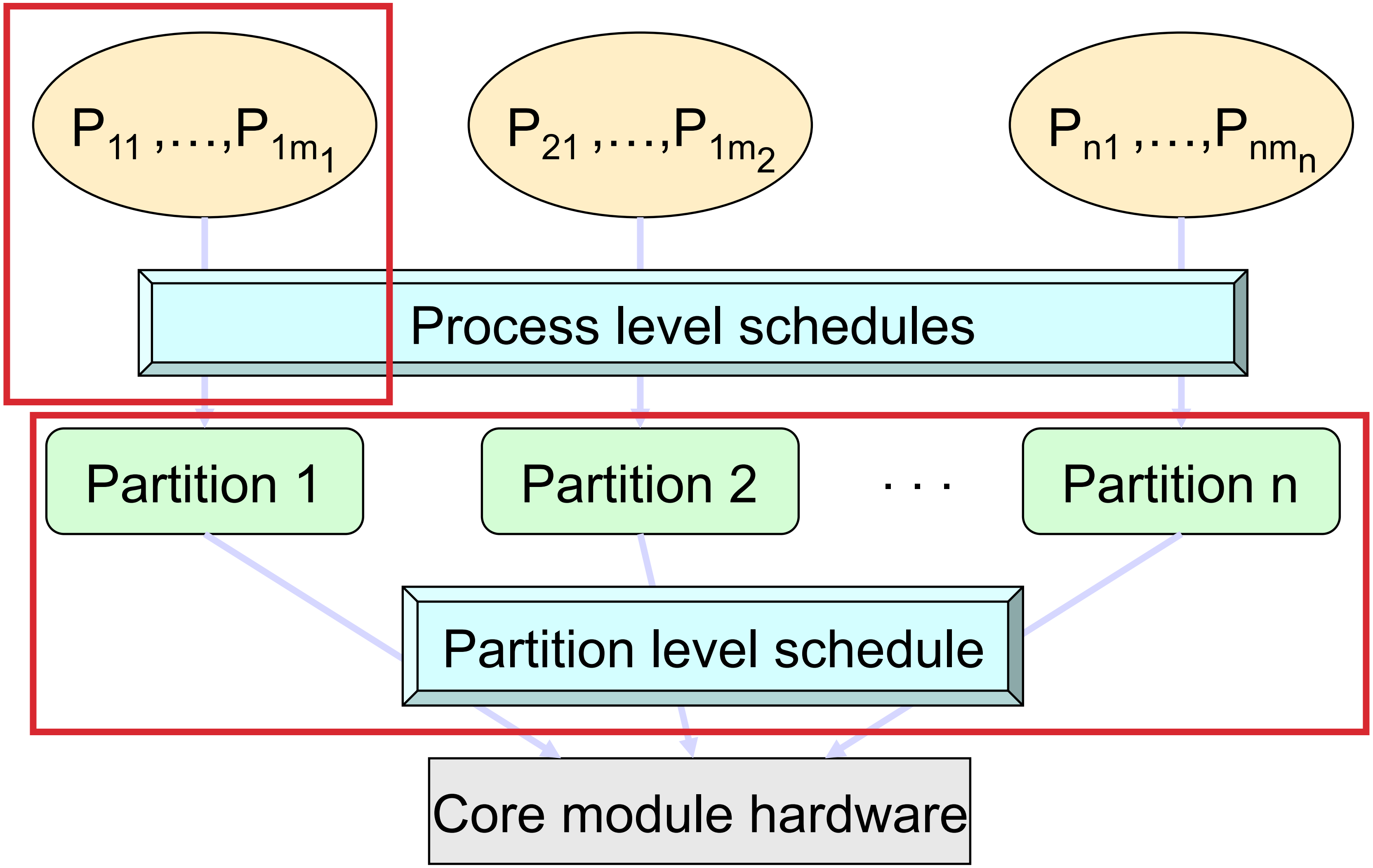
Motivation and Goal

- Embedded systems are becoming complex, networked, and large -scale.
- Embedded systems have many para-functional aspects:
  - physically coupled, real-time, location-aware, resource -constrained, heterogeneous, and etc.
  - real-time: required to react to events or complete tasks in specific time
  - resource-constrained: subject to operating with scarce resources, such as processor power, memory, power, bandwidth
- Component-based approach for the design of large complex systems
  - Interoperability, predictability, scalability,…
- Goal: resource-sensitive component framework
  - Hierarchical, compositional, incremental

Component technologies

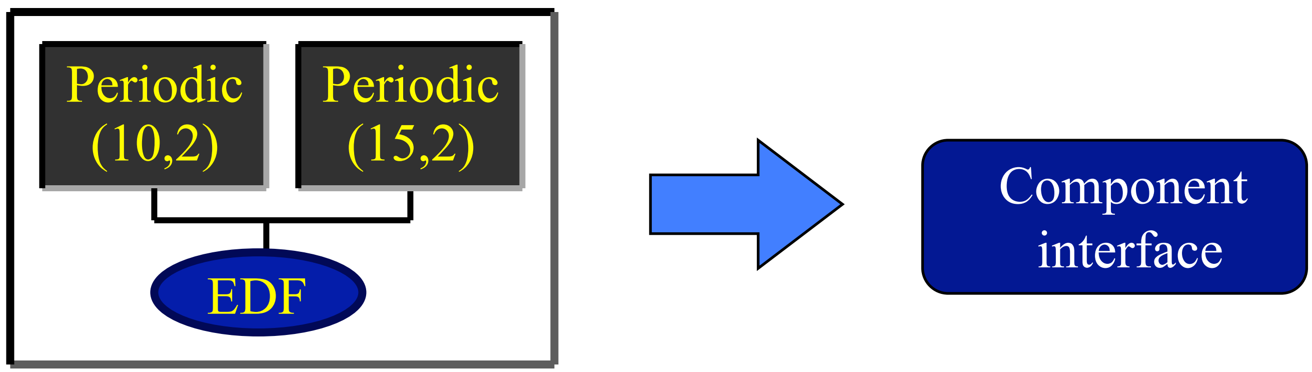
- Enable component-based development
  - abstract components through interfaces
    - Interfaces preserve intellectual property
  - compose components preserving compositionality
  - facilitate modularity, portability, and reusability
- Traditional focus: functional, behavioral aspects
  - need: non-functional aspects, such as timeliness, reliability, safety, and resource use

ARINC 653: Schedulability



Abstraction and Composition

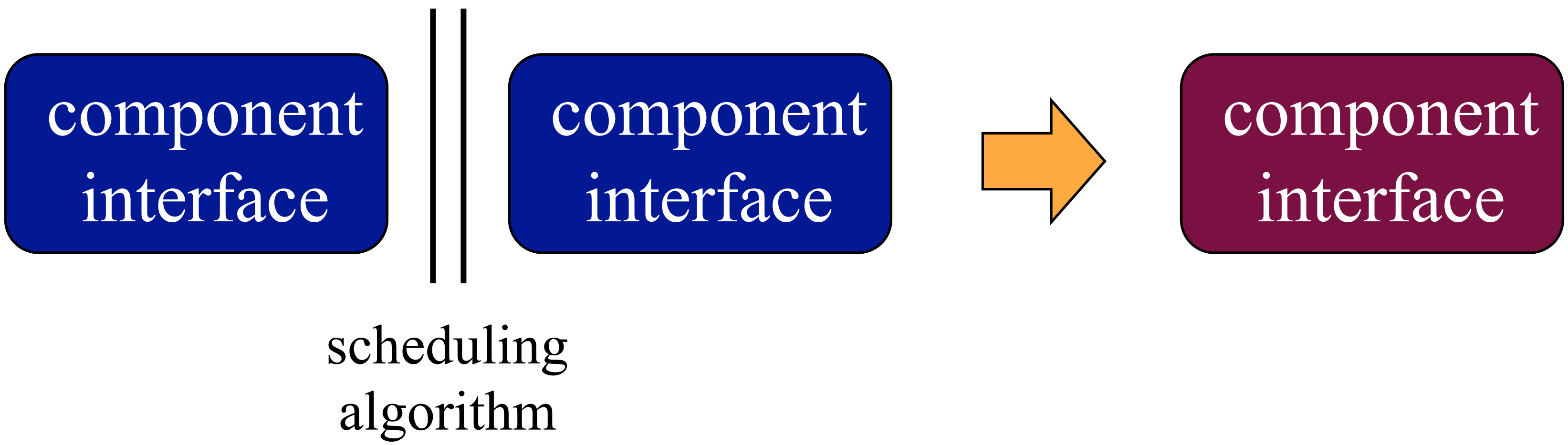
- Abstraction Problem: abstract the real-time application as a component with an interface



- Compute the minimum real-time requirements necessary for guaranteeing the schedulability of a component

Abstraction and Composition

- Composition Problem: compose component-level properties into system-level (or next-level component) properties



Compositionality

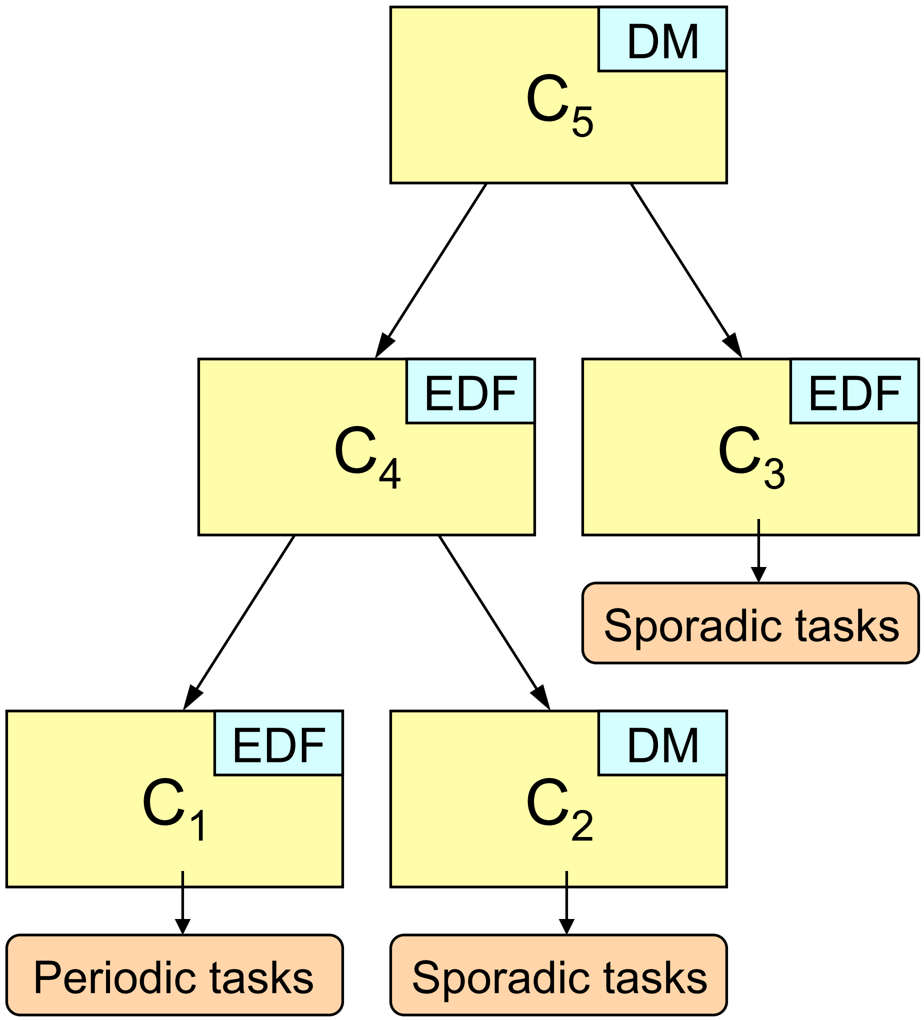
- Compositionality:
  – system-level properties can be established by composing independently analyzed component-level properties
- Compositional reasoning based on assume/guarantee paradigm
  – components are combined to form a system such that properties established at the component-level still hold at the system level.
- Compositional schedulability analysis using the demand /supply bounds
  – Establish the system-level timing properties by combining component-level timing properties through interfaces

Resource Satisfiability Analysis

- Given a task set and a resource model, resource satisfiability analysis is to determine if, for every time,

resource demand, which a task set needs under a scheduling algorithm	≤	(minimum possible) resource supply

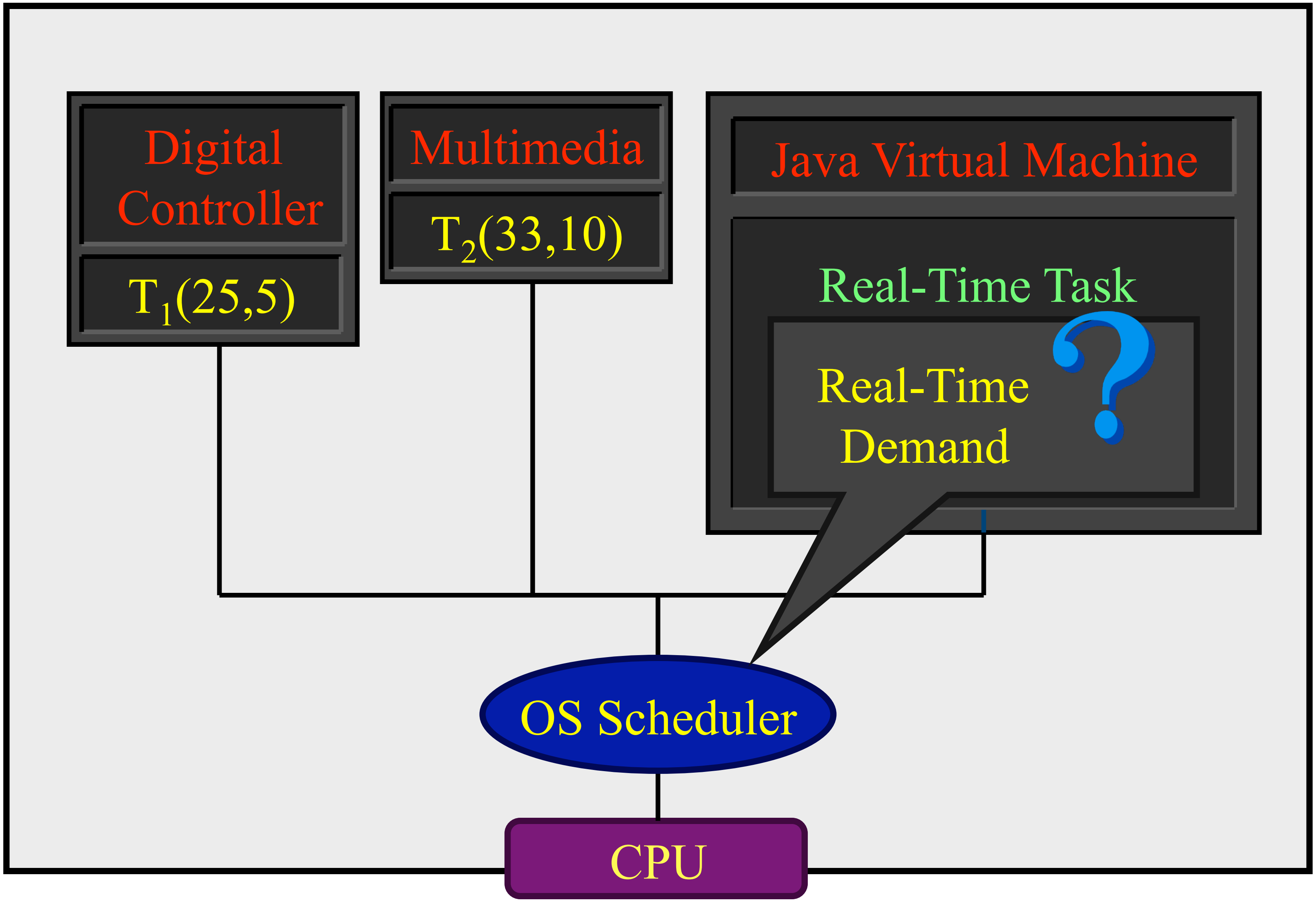
Hierarchical Scheduling Framework



- Resource allocation from parent to child
- Notations
  - Leaf → C_1, C_2, C_3
  - Non-leaf → C_4, C_5
  - Root → C_5

ARINC 653 → Two-level hierarchical framework

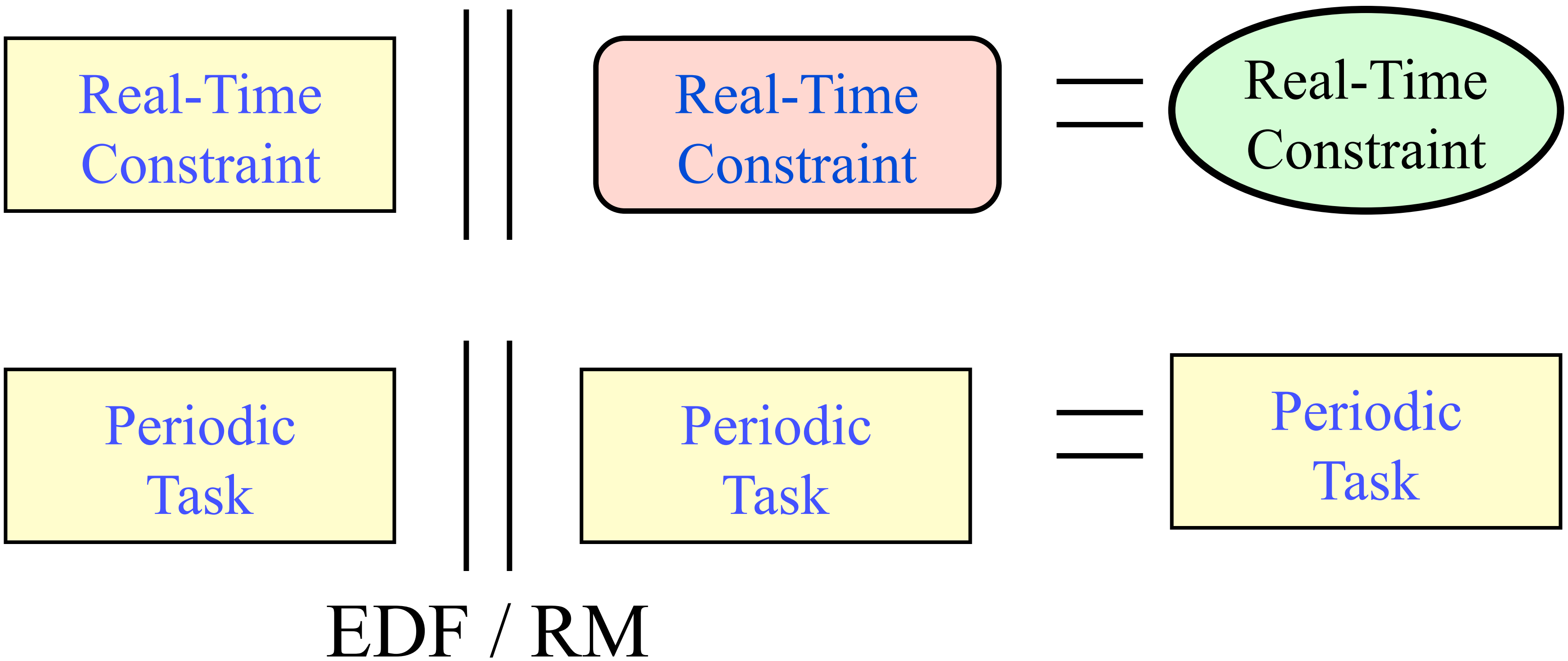
OS Scheduler's Viewpoint



Resource Demand Models

Real-time demand composition

- Combine real-time requirements of multiple tasks into real-time requirement of a single task



Non-composable periodic models?

- What are right abstraction levels for real-time components?

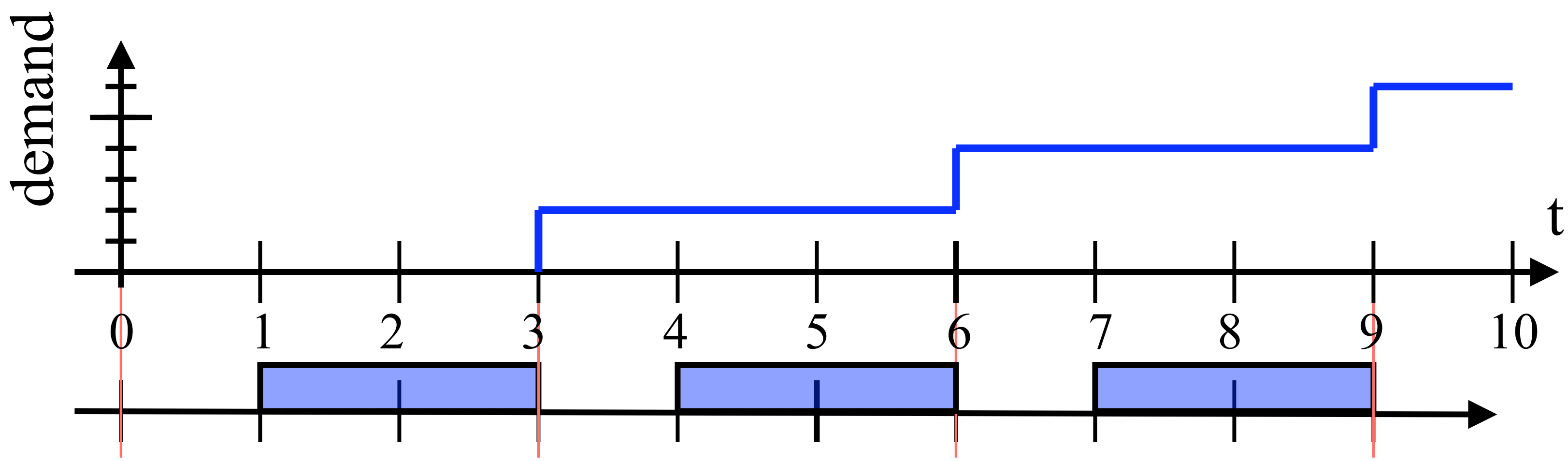
  (period, execution time)

- P1 = (p1,e1); e.g., (3,1)
- P2 = (p2,e2); e.g., (7,1)
- What is P1 || P2?
  - (LCM(p1,p2), e1*n1 + e2*n2); e.g., (21,10) where n1*p1 = n2*p2 = LCM(p1,p2)

- What is the problem?
  - beh(P1) || beh(P2) = beh(P1||P2)?
- Compositionality
  - (P1 || P2) || P3 = P || P3, where P = P1 || P2

Resource Demand Bound

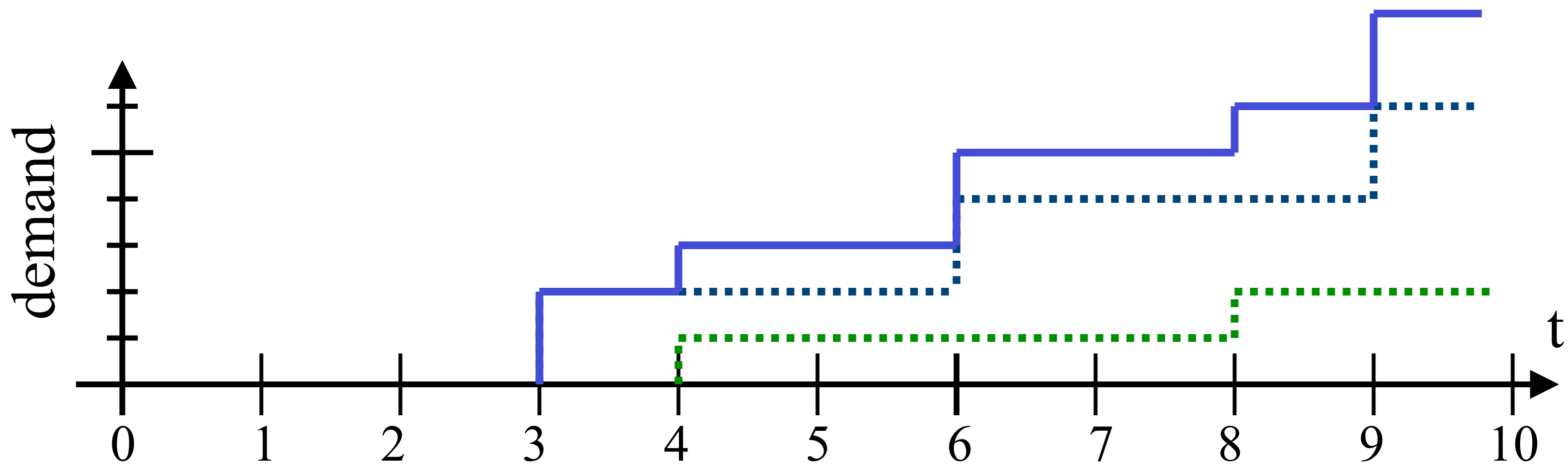
- Resource demand bound during an interval of length t
  - dbf(W,A,t) computes the maximum possible resource demand that W requires under algorithm A during a time interval of length t
- Periodic task model T(p,e) [Liu & Layland, '73]
  - characterizes the periodic behavior of resource demand with period p and execution time e
  - Ex: T(3,2)



Demand Bound - EDF

- For a periodic workload set *W* = {Ti(pi,ei)},
  - dbf (*W*,*A*,*t*) for EDF algorithm [Baruah et al.,'90]

$$\text{dbf}\,(\text{W,EDF,t}) = \sum_{T_i \in W} \left\lfloor \frac{t}{p_i} \right\rfloor \cdot e_i$$



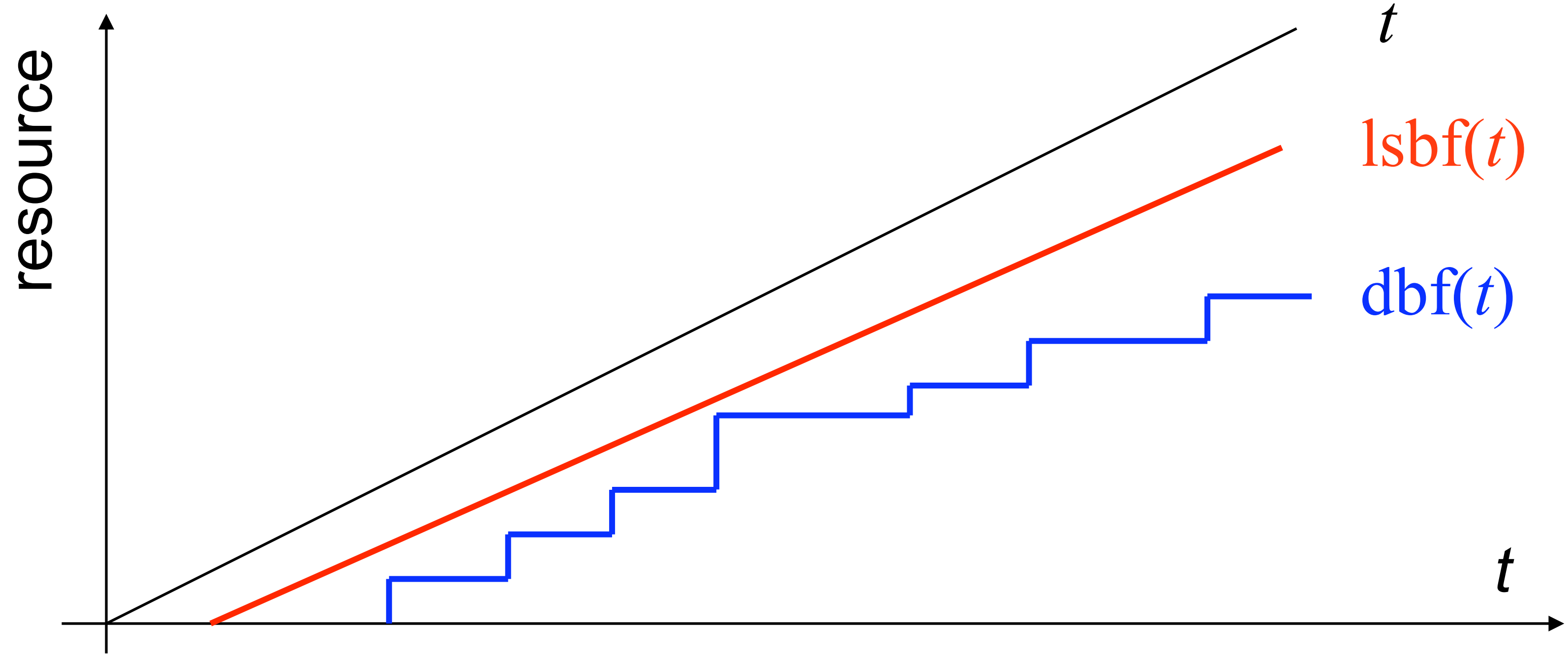
Demand-based Schedulability Analysis

- A periodic task set is schedulable under EDF

  over the periodic resource model $\Gamma(P,Q)$

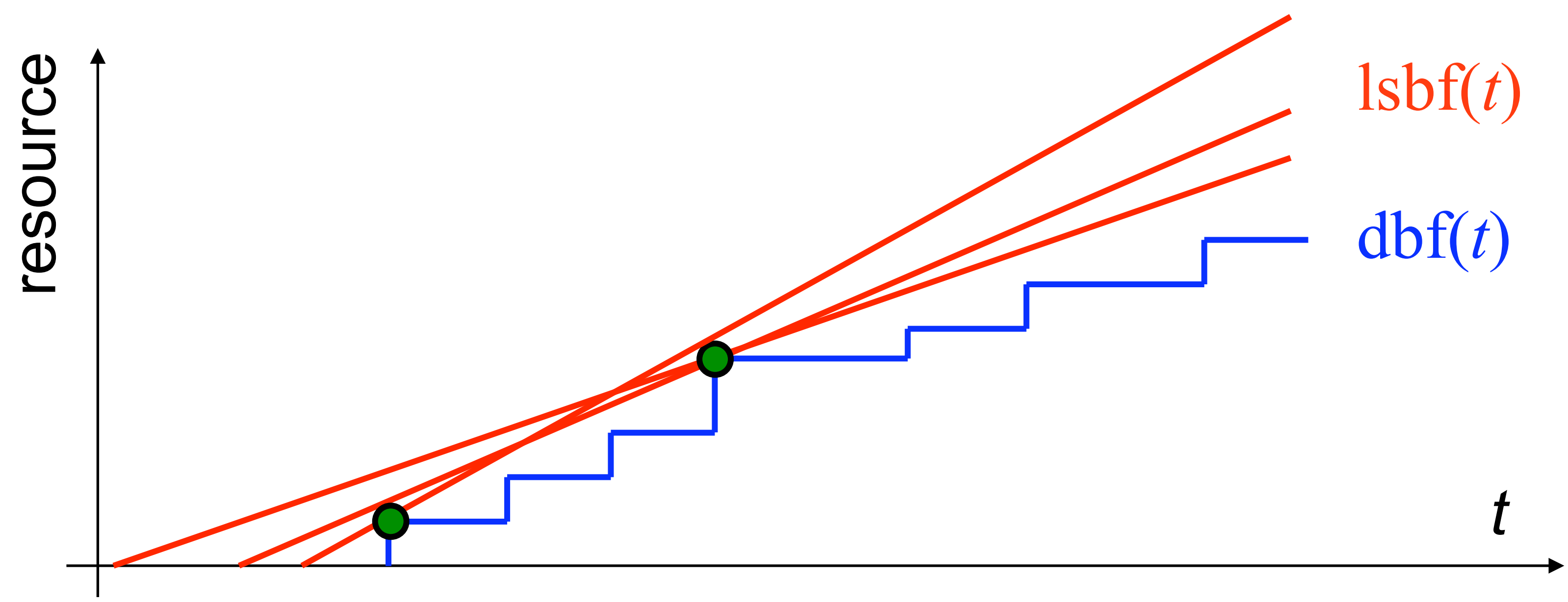
  if and only if $\forall t > 0 \ \mathrm{dbf}(t) \le t \le \mathrm{lsbf}(t)$

  [Shin and Lee, 2003]

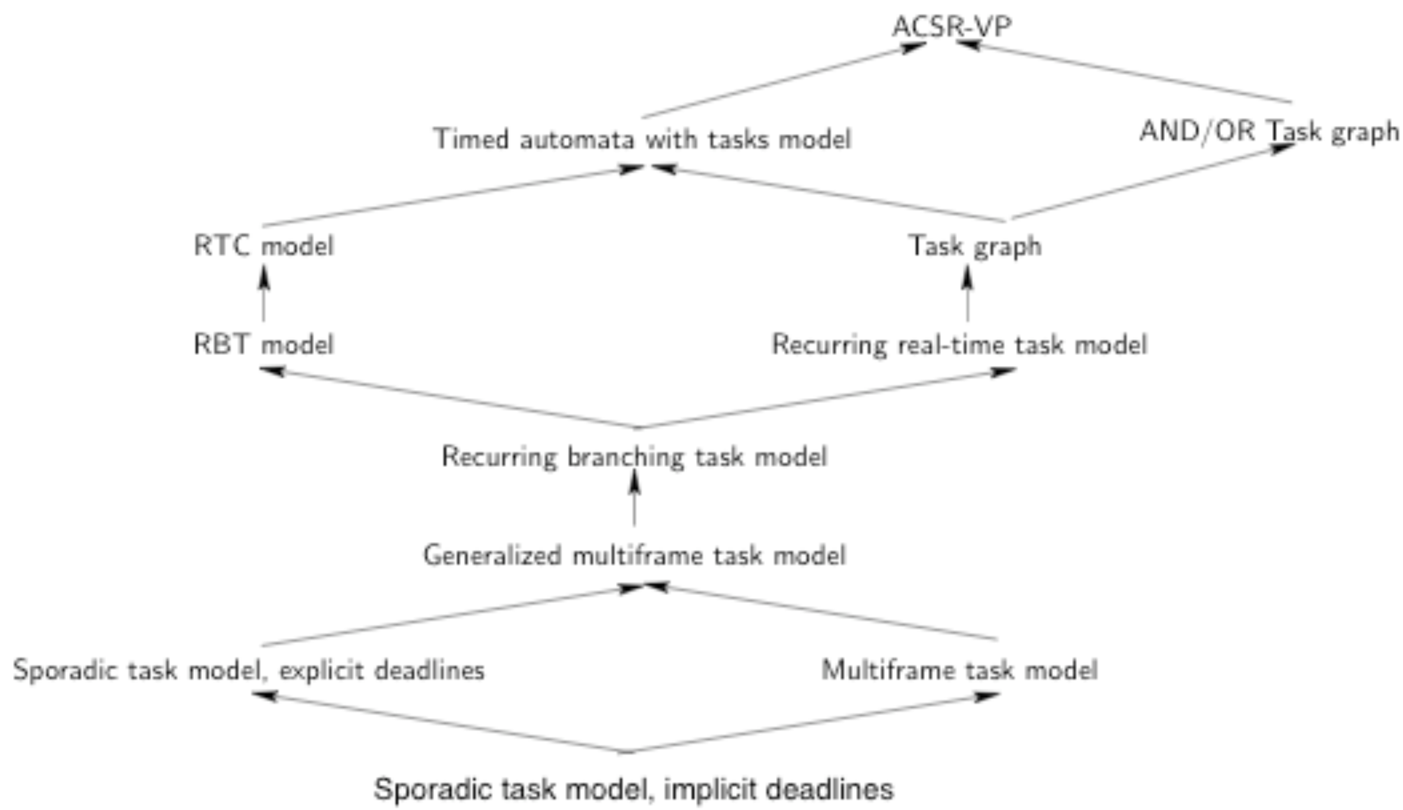


Demand bound revisited

- More than one resource model may be used
  - Consider only LSBF that intersect DBF
- An "optimal" choice from the component perspective may be globally unsuitable



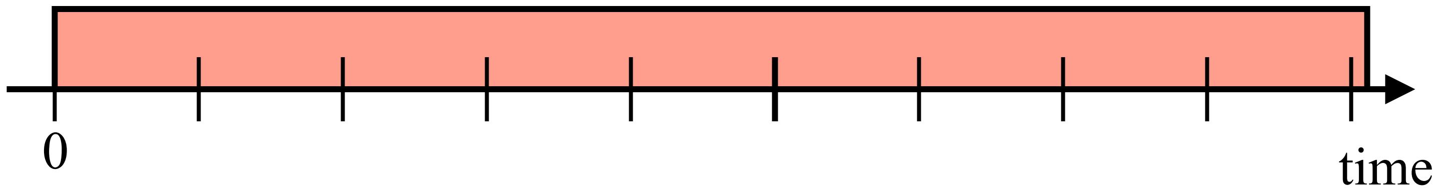
Task (resource demand) representations



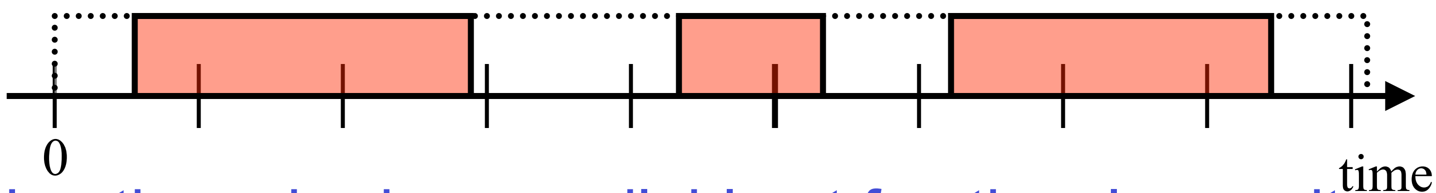
Resource Supply Models

Resource Modeling

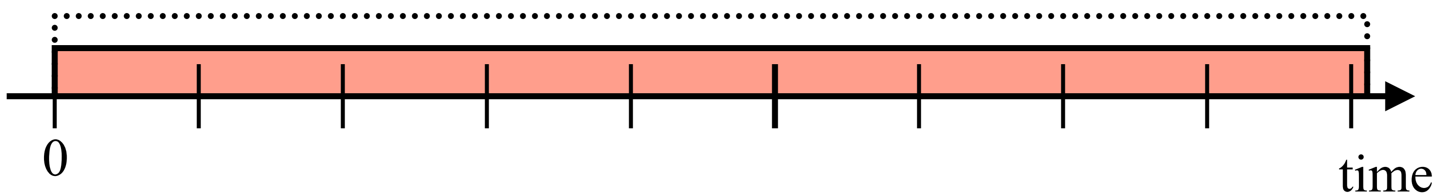
- Dedicated resource : always available at full capacity



- Shared resource : not a dedicated resource
  - Time-sharing : available at some times

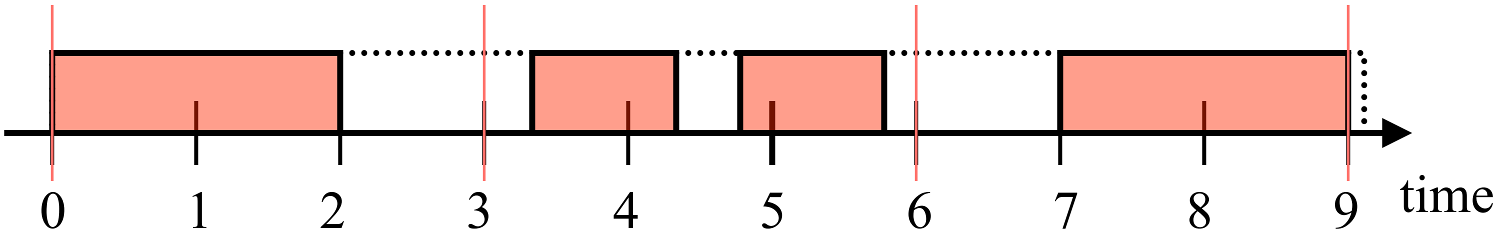


  - Non-time-sharing : available at fractional capacity



Resource Modeling

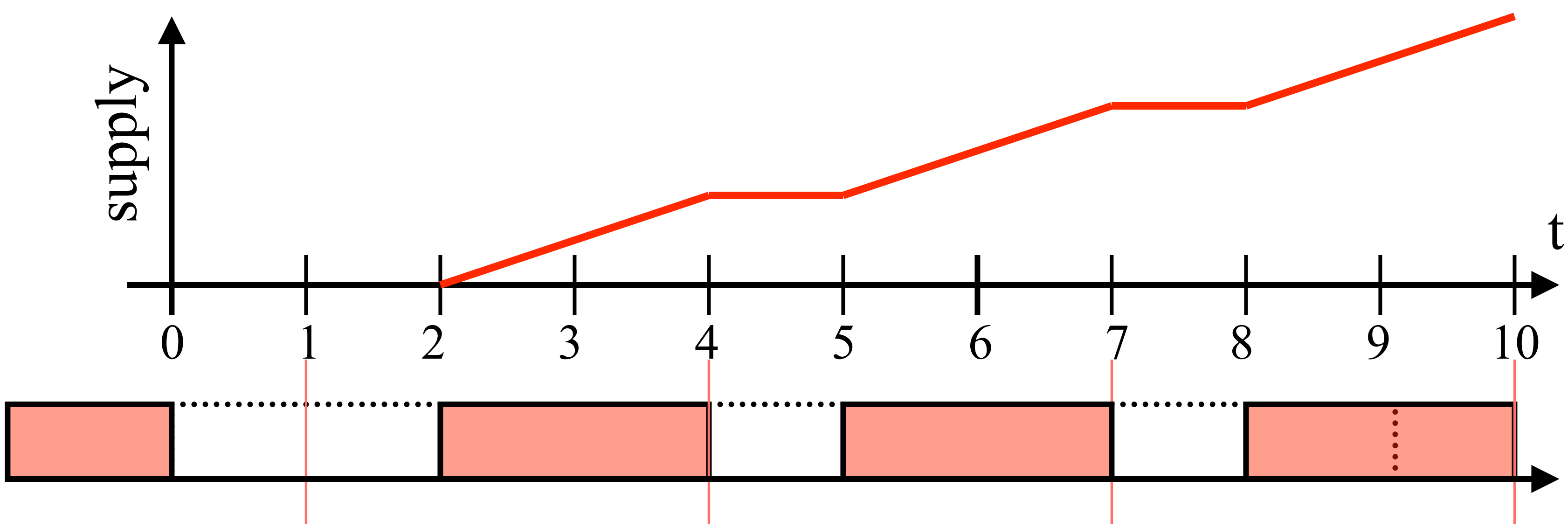
- Time-sharing resources
  - Bounded-delay resource model [Mok et al., '01] characterizes a time-sharing resource w.r.t. a non-time-sharing resource
  - Periodic resource model Γ(Π,Θ) [Shin & Lee, RTSS '03] characterizes periodic resource allocations
  - EDP model [Easwaran et all, RTSS 07]



Resource Supply Bound

- Resource supply during an interval of length t
  – $sbf_R(t)$: the minimum possible resource supply by resource R over all intervals of length t

- For a single periodic resource model, i.e., Γ(3,2)
  – we can identify the worst-case resource allocation

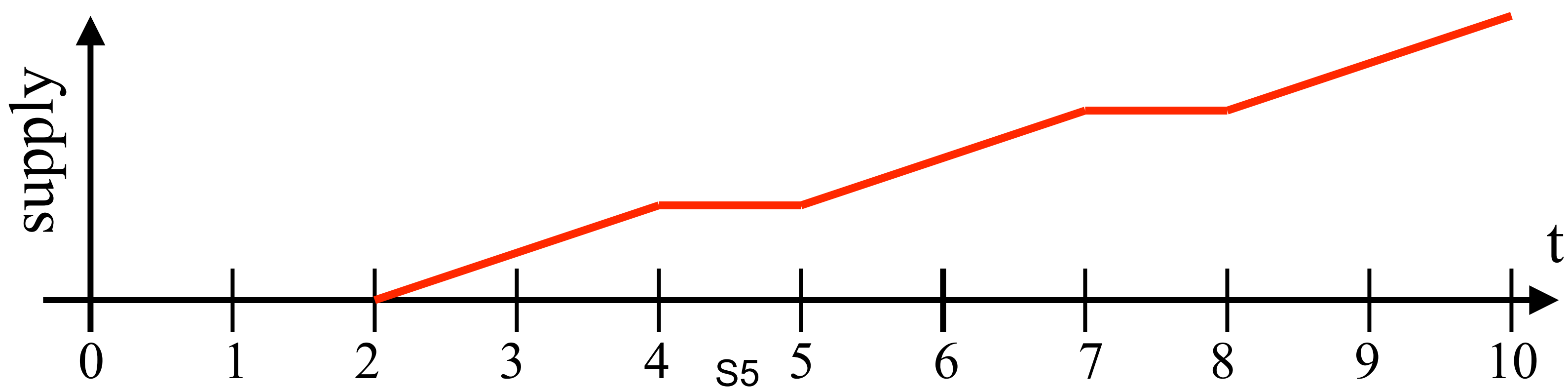


Resource Supply Bound

- Resource supply during an interval of length t
  - $sbf_{\Gamma}(t)$: the minimum possible resource supply by resource R over all intervals of length t

- For a single periodic resource model $\Gamma(\Pi,\Theta)$

$$\mathrm{sbf}_{\Gamma}(t) = \begin{cases} t-(k+1)(\Pi-\Theta) & \text{if } t \in [(k+1)\Pi-2\Theta, (k+1)\Pi-\Theta] \\ (k-1)\Theta & \text{otherwise} \end{cases}$$



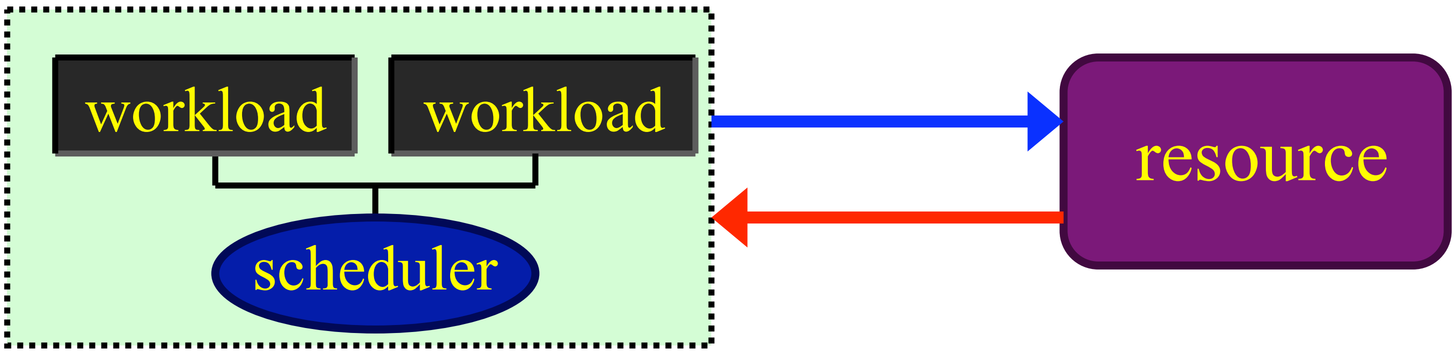
Resource Schedulability Analysis

- Schedulability analysis determines whether

resource demand, which a workload set requires under a scheduling algorithm

≤

resource supply, which available resources provide



Schedulability conditions

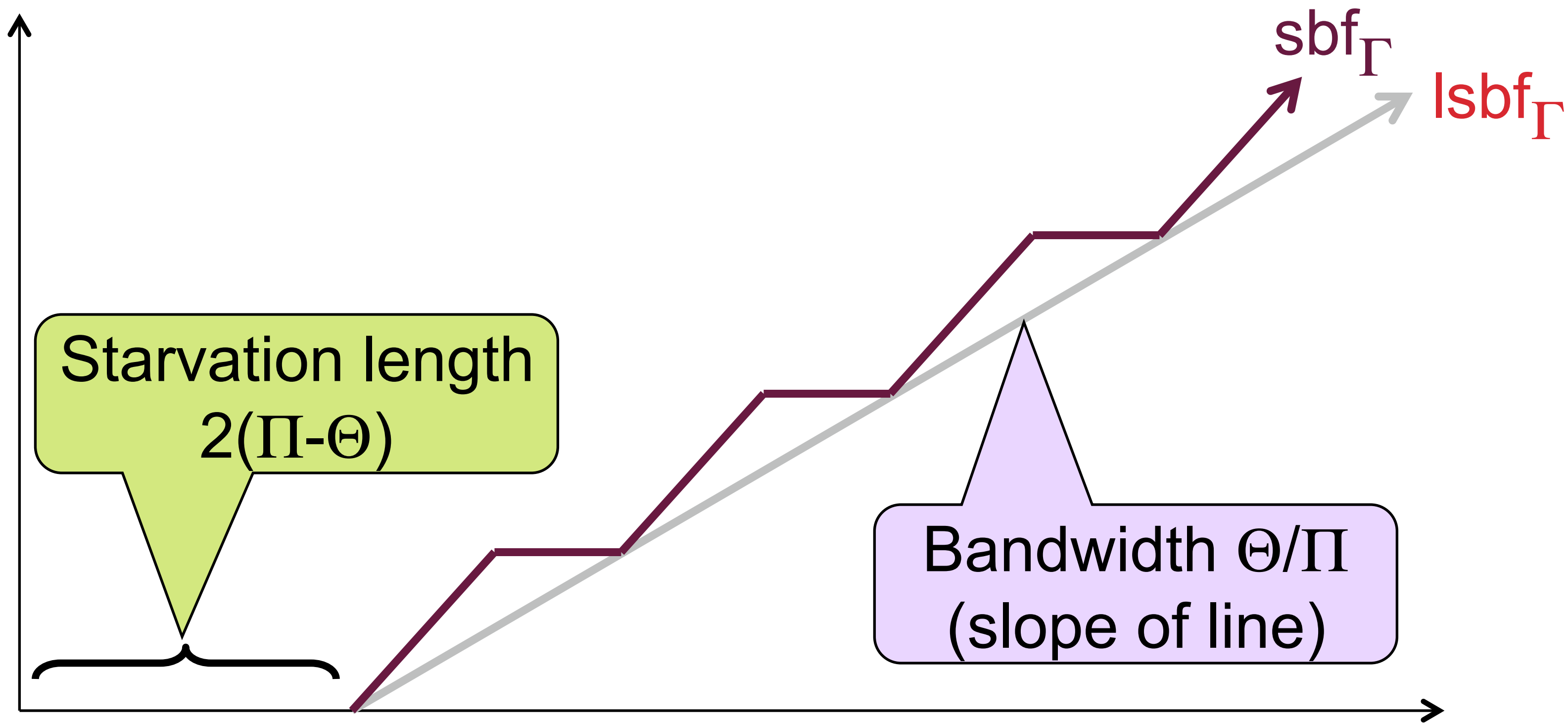
- $sbf_{\Gamma}(t)$: Supply bound function: Minimum resource supply of model Γ in any time interval of length t
- $lsbf_{\Gamma}(t)$: Linear lower bound of $sbf_{\Gamma}(t)$

$$lsbf_{\Gamma}(\mathrm{t}) = \frac{\Theta}{\Pi}(t - \underbrace{2(\Pi - \Theta)})$$

Bandwidth

Starvation length

Schedulability conditions



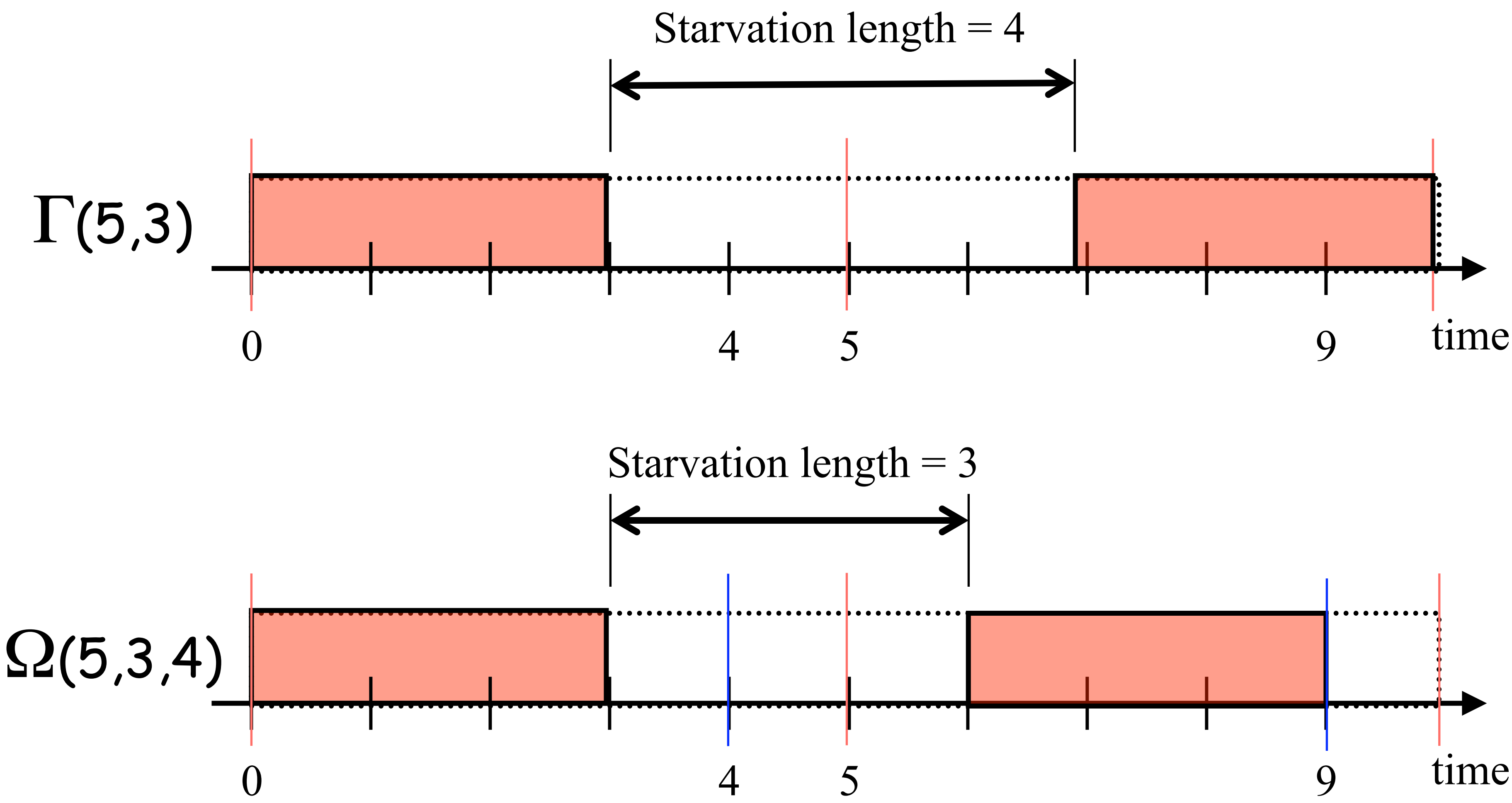
The EDP Resource Model

- Explicit Deadline Periodic resource

- Model: $\Omega = (\Pi, \Theta, \Delta)$
  - Explicit deadline Δ
  - Θ resource units in Δ time units
  - Repeat supply every Π time units

- Properties
  - Periodic resource model is a EDP model with $\Delta = \Pi$
  - Maximum slack of EDP model depends on Θ and Δ for a fixed Π
  - Slack can be controlled using Δ without changing bandwidth of model (within limits)
  - Smaller bandwidth required to schedule the same component, when compared to periodic resource models
  - improves precision of resource allocation

Supply bound function (sbf_{Ω})

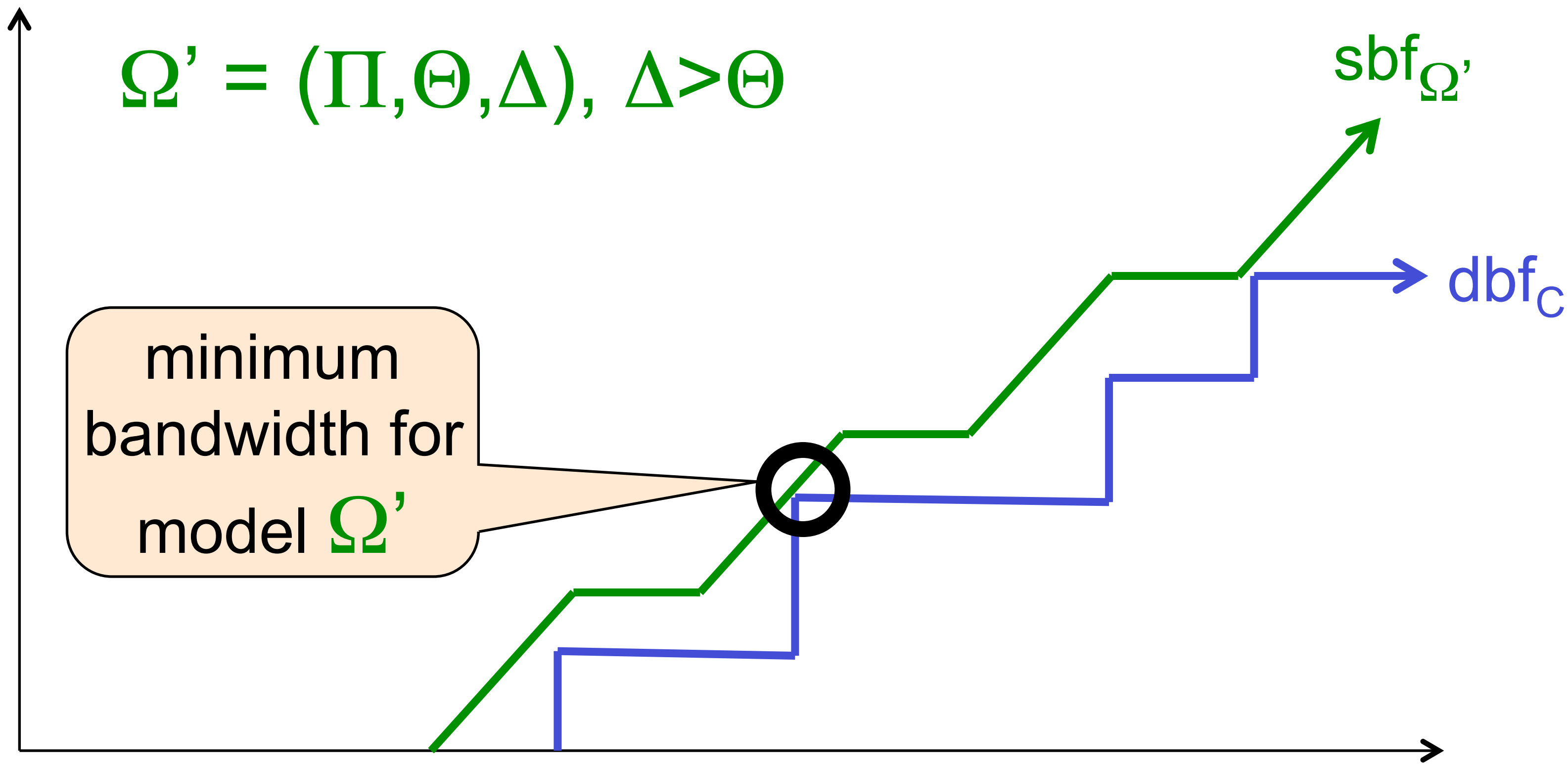


Bandwidth optimal interface

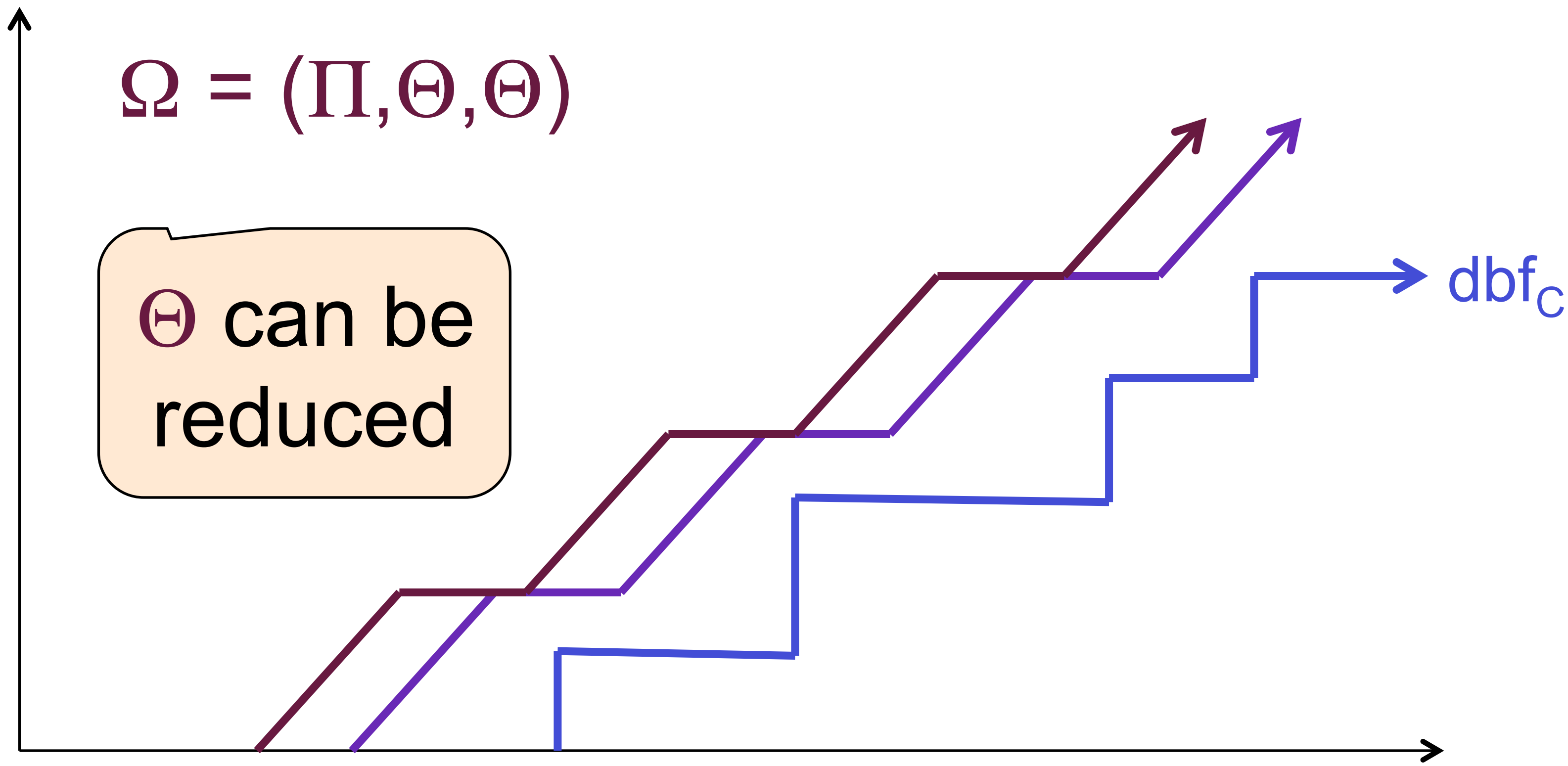
- Given component C and period Π
  - Compute Θ and Δ

- We use bandwidth optimality
  - Minimizes resource bandwidth Θ/Π
  - Occurs when $\Delta=\Theta$ (***Theorem 3.2 in RTSS'07***)

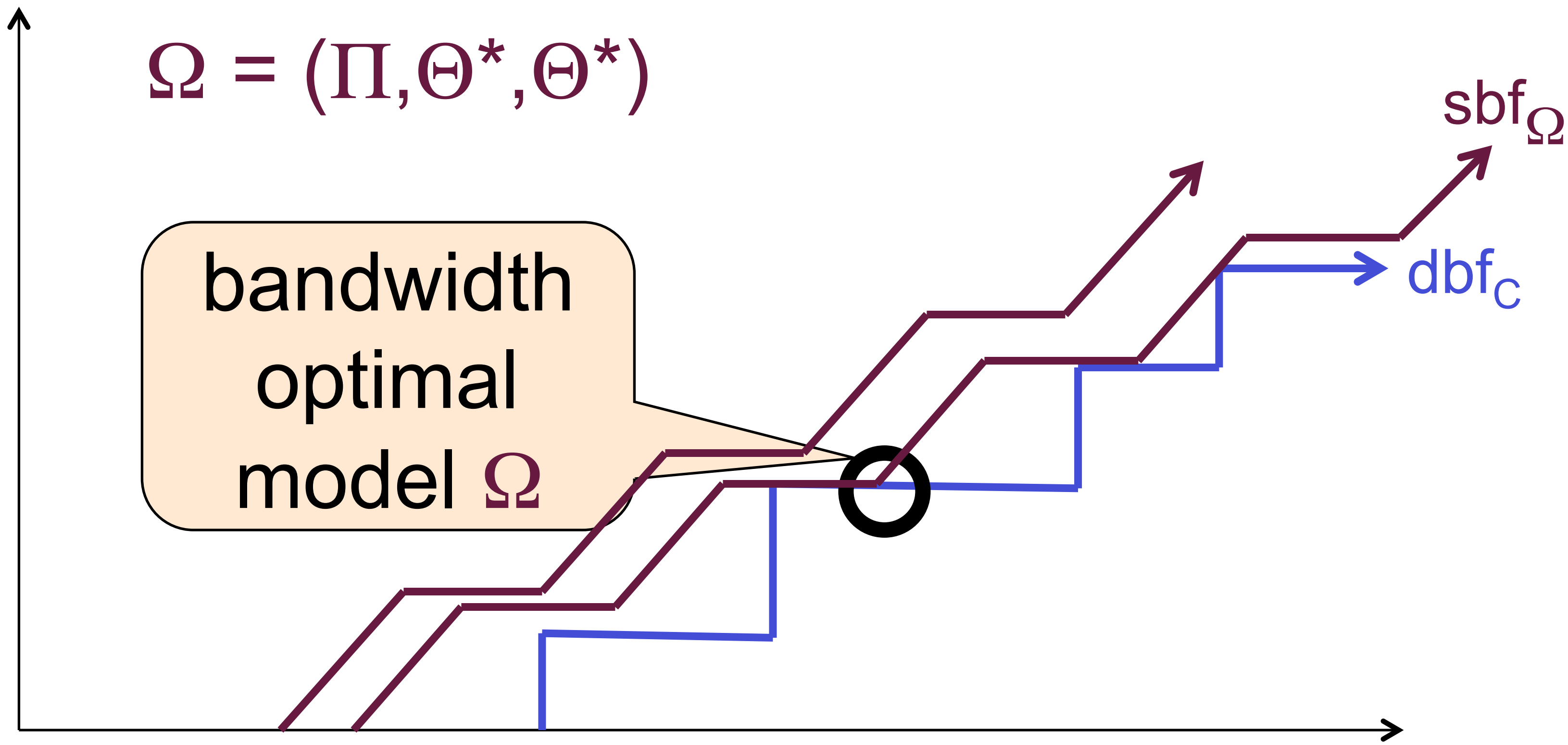
Bandwidth optimal interface



Bandwidth optimal interface



Bandwidth optimal interface



Bandwidth-deadline optimal

- Choose interface with Largest Δ among all bandwidth optimal interfaces
  - Reduced demand for composition

- Interface generation procedure
  - Set $\Delta=\Theta$, compute $\Omega = (\Pi,\Theta^*,\Theta^*)$
  - Set $\Theta=\Theta^*$, compute $\Omega^* = (\Pi,\Theta^*,\Delta^*)$

Applying to ARINC 653

- 2-level hierarchical scheduler
  - Partitions scheduled among themselves at higher level
  - Processes within each partition scheduled at lower level
- Uniqueness of ARINC 653
  - Harmonic partition periods
  - Preemption and blocking overheads
  - Communication dependencies across partitions
    - Process workload (dbf) depends on parameters which in turn are determined by these dependencies
- Appling to real ARINC workloads obtained from Honeywell
  - Preliminary results showed an improvement of up to 300% in bandwidth, depending on period of interfaces for 5-6 partitions, with 1-5 tasks each
- Tool (called CARTS) development underway to handle more extensive workloads

Example: ARINC workload

- Process parameters: (O,J,T,C,D)
  - O = Offset, J = Jitter, T = Period, C = Worst -case execution time, D = Deadline
  - T,C,D from workload, O added speculatively

- Example 1

  Partition 1: {(2,0,25,1.4,25), (3,0,50,3.9,50)}

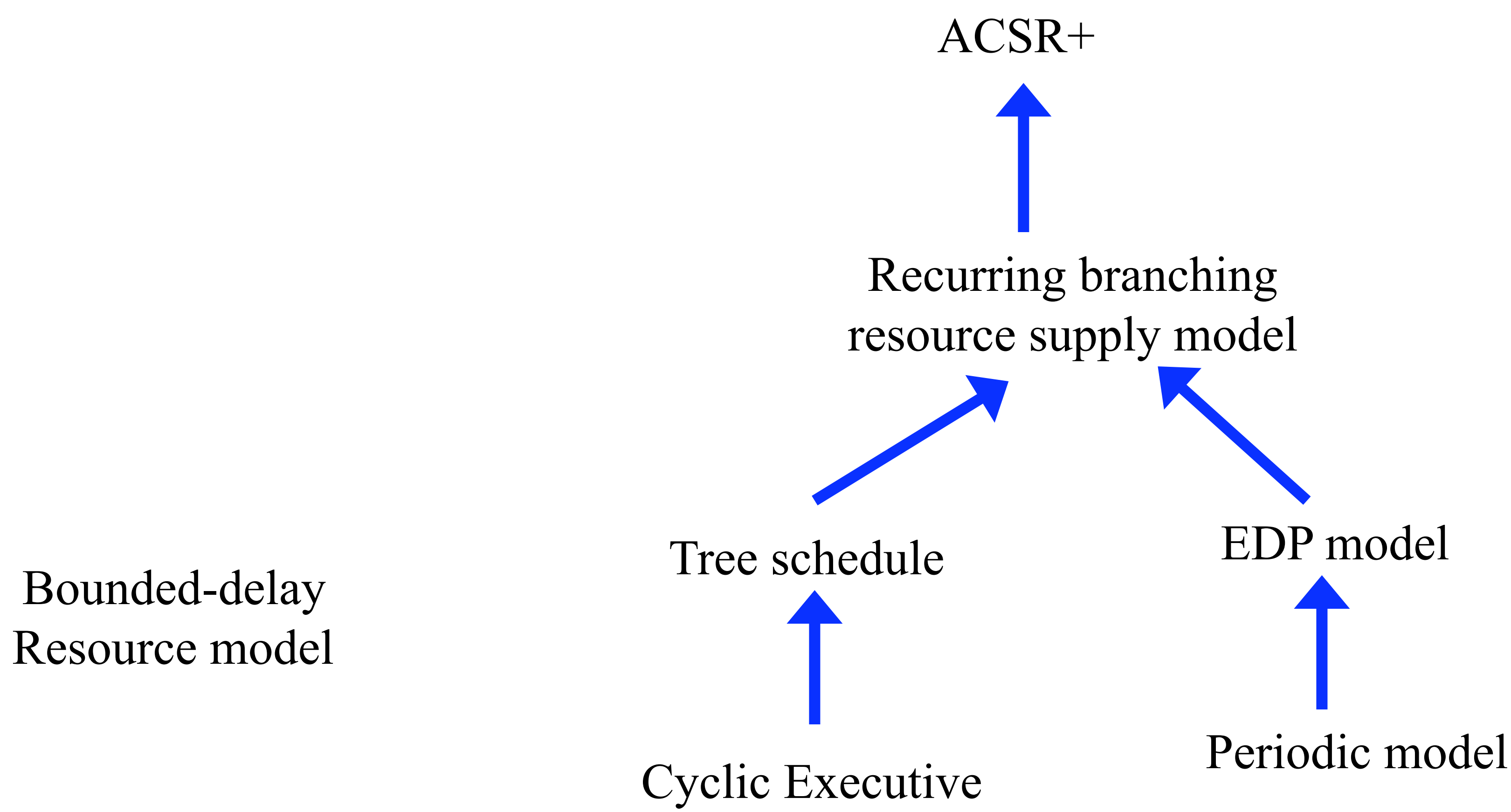
  Partition 2: {(0,0,50,2.8,50)}

  Partition 3: {(0,0,50,1.4,50)}

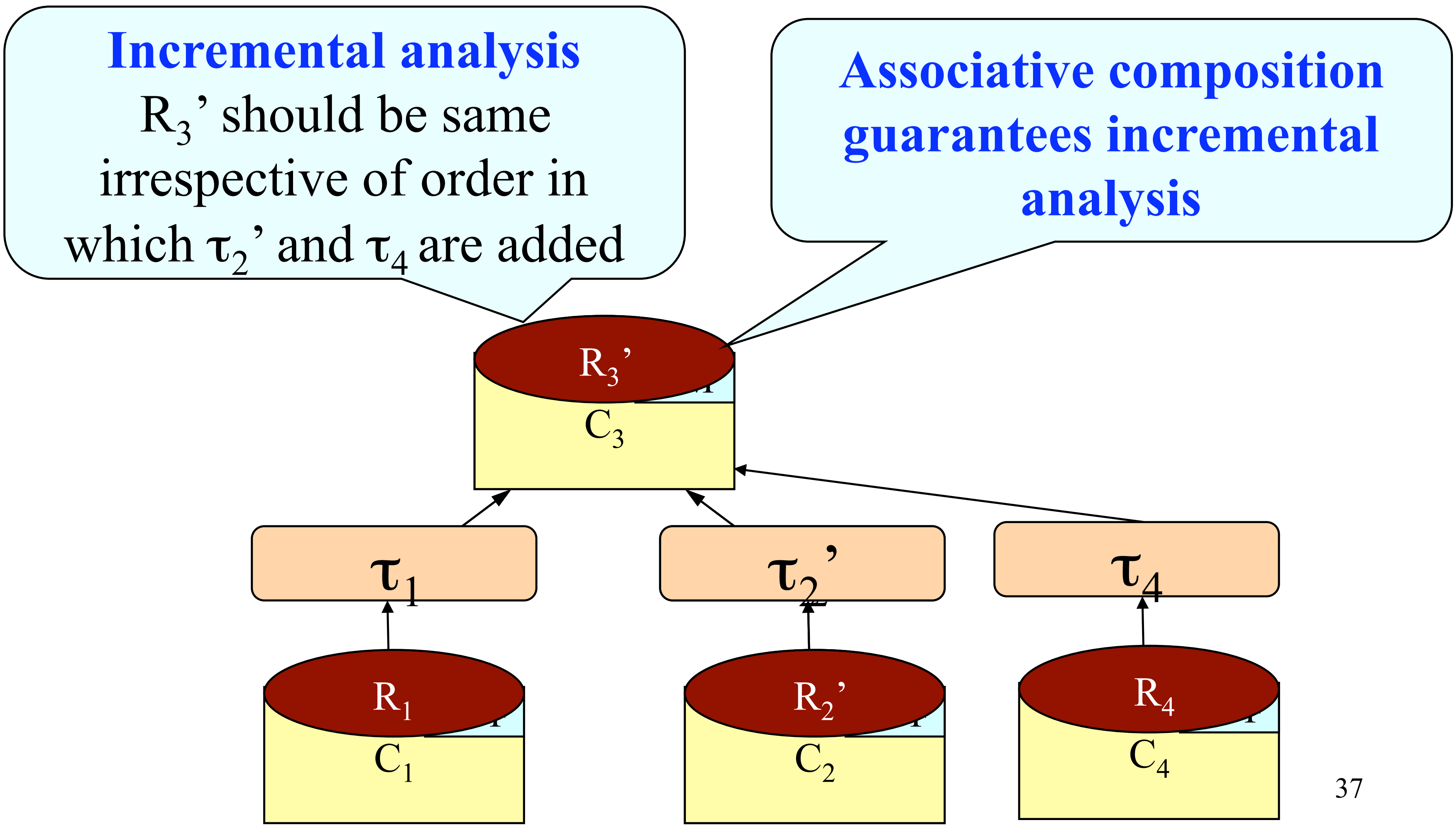
  Partition 4: {(3,0,25,1.1,25), (5,0,50,1.8,50), (11,0,100,2,100), (13,0,200,5.3,200)}

  Partition 5: {(2,0,50,1.3,50), (14,0,200,1.5,200)}

Resource Supply Models

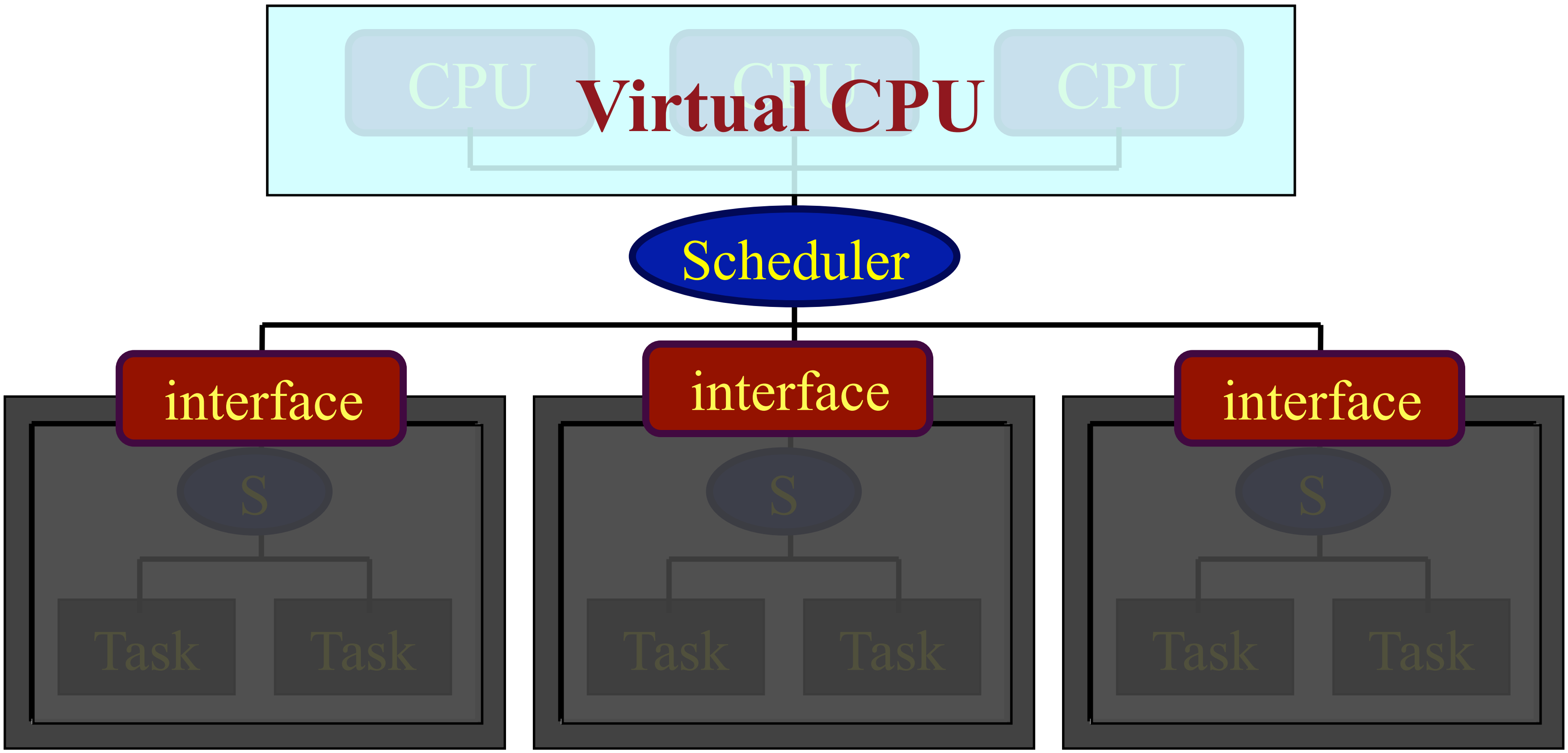


Incremental Analysis

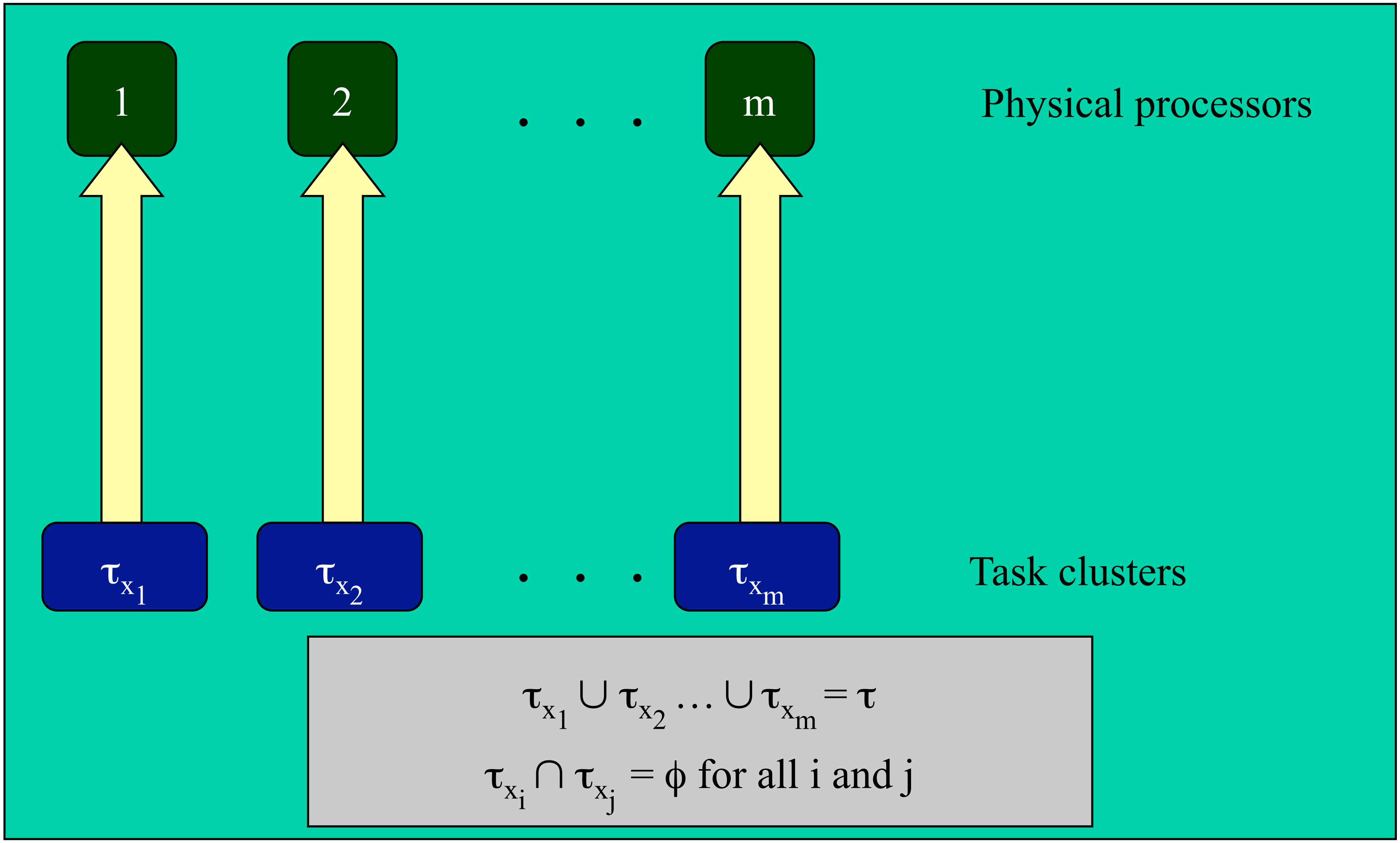


Multicore Processor Virtualization

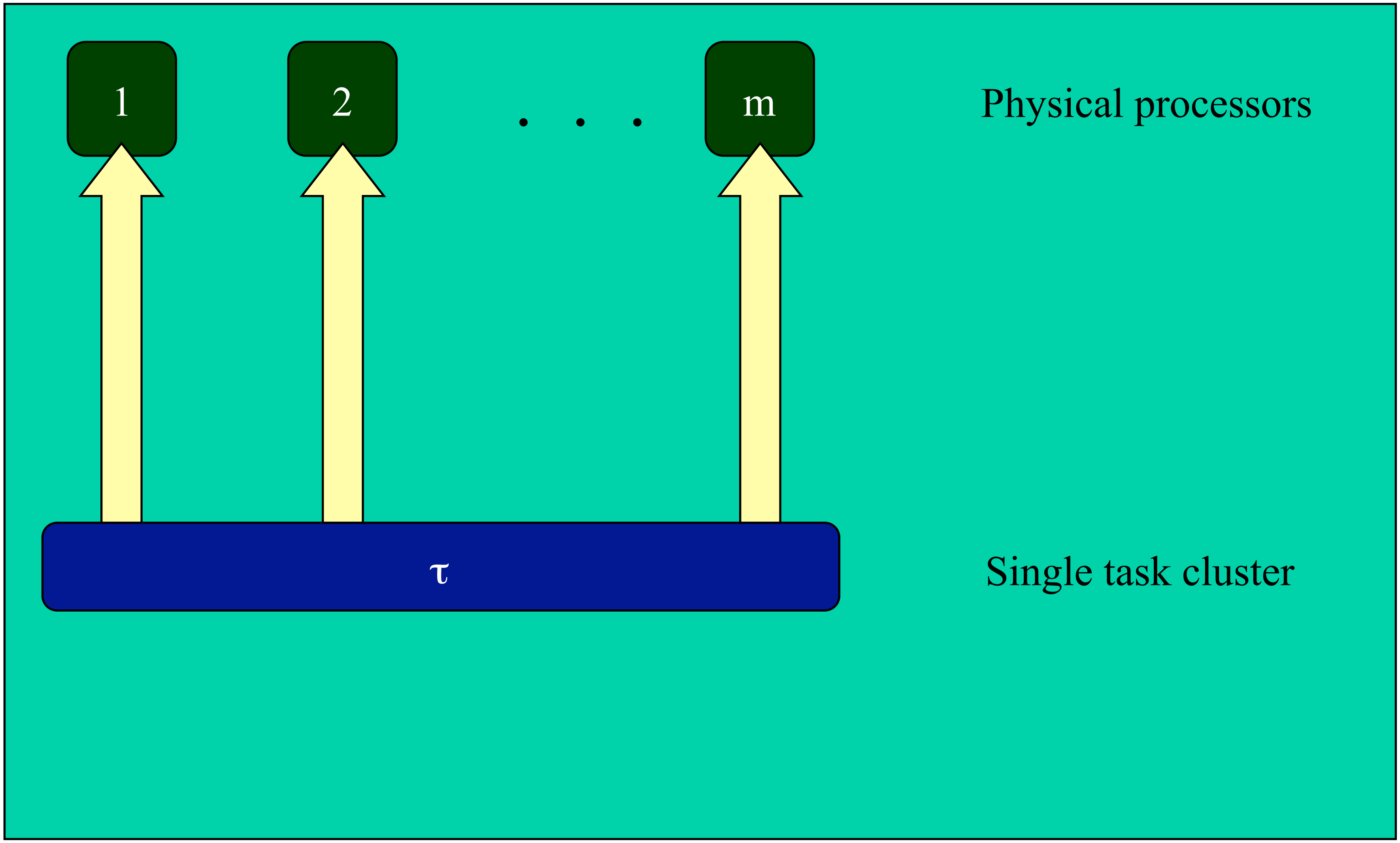
1. Compositional analysis of hierarchical multiprocessor real-time systems, through component interfaces
2. Using virtualization to develop new component interface for multiprocessor platforms



Partitioned Scheduling



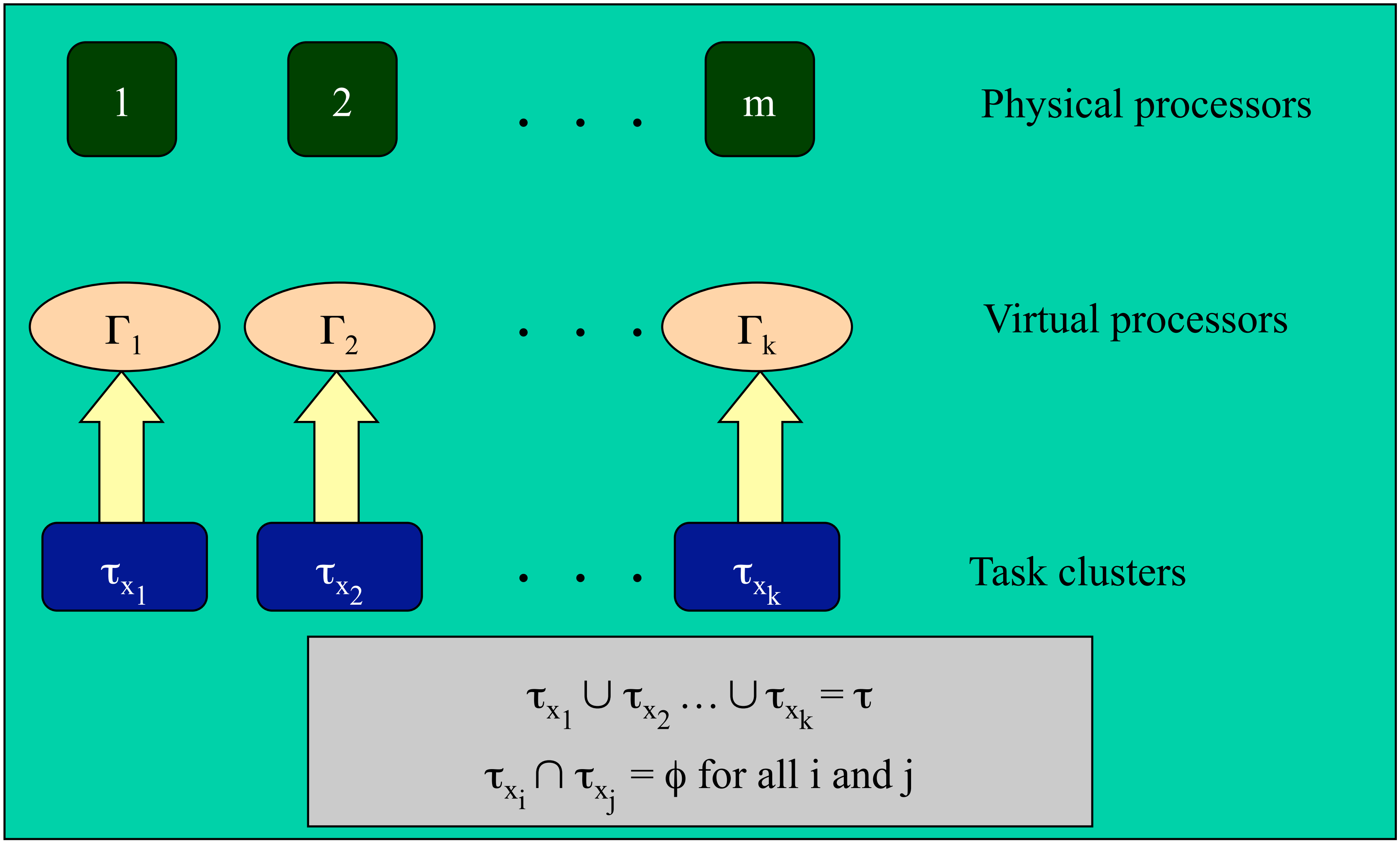
Global Scheduling



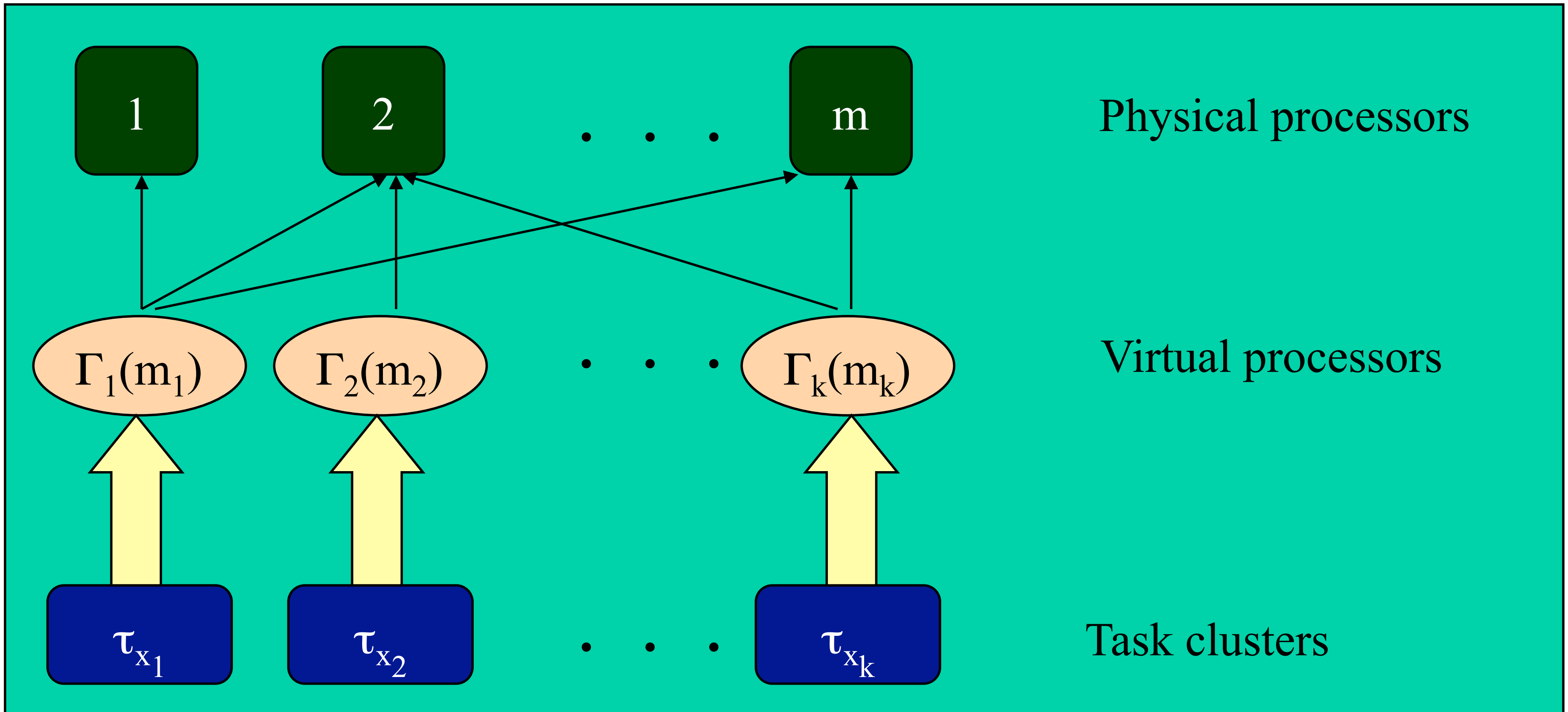
Multiprocessor Scheduling

- Goal: Optimal scheduling algorithms and their analysis techniques
- Partitioned vs. Global Scheduling
  - Shown using simulations[Baker05] that partitioned performs better
  - Exists task sets schedulable by global but not by any partitioned algorithm
  - EDF load bounds: $1/2(m - (m-1)\delta_{max})$[partitioned] vs. $(m - (m-1)\delta_{max})(1-\delta_{max})$[global]
- Our Approach: Framework for development of scheduling algorithms that support general task -processor mappings through virtualization

Virtual Clustering Interface



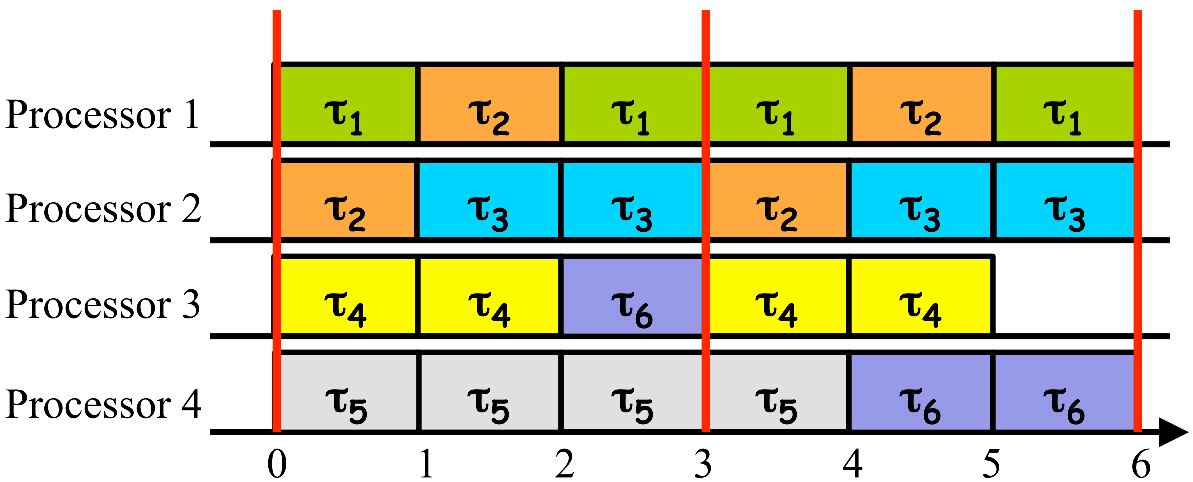
Virtual Clustering Interface



For each Γ_i, m_i ($\leq m$) is maximum number of physical processors that can be assigned to Γ_i at any instant

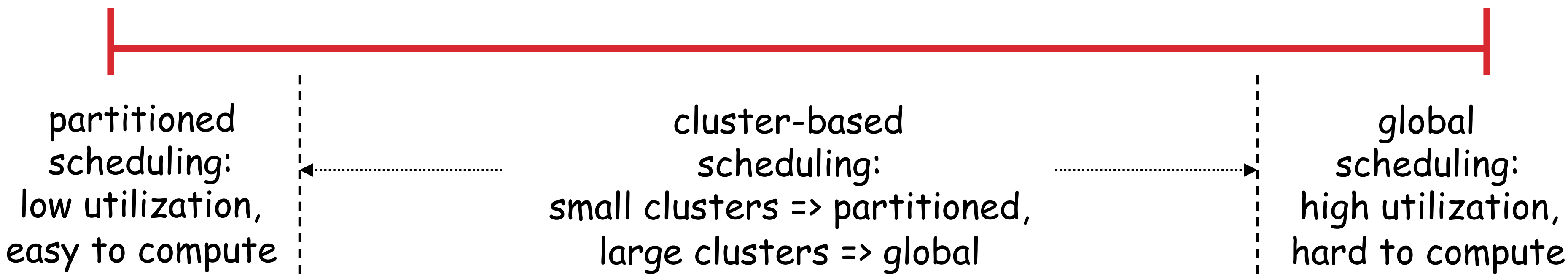
Virtual Clustering

- Task set and number of processors
  - $\tau_1=\tau_2=\tau_3=\tau_4=(3,2,3)$, $\tau_5=(6,4,6)$, and $\tau_6=(6,3,6)$, m=4
- Schedule under clustered scheduling
  - τ_1, τ_2, τ_3 scheduled on processors 1 and 2
  - τ_4, τ_5, τ_6 scheduled on processors 3 and 4



Virtual Clusters

- Use platform virtualization to provide a trade-off between resource utilization and scheduling complexity

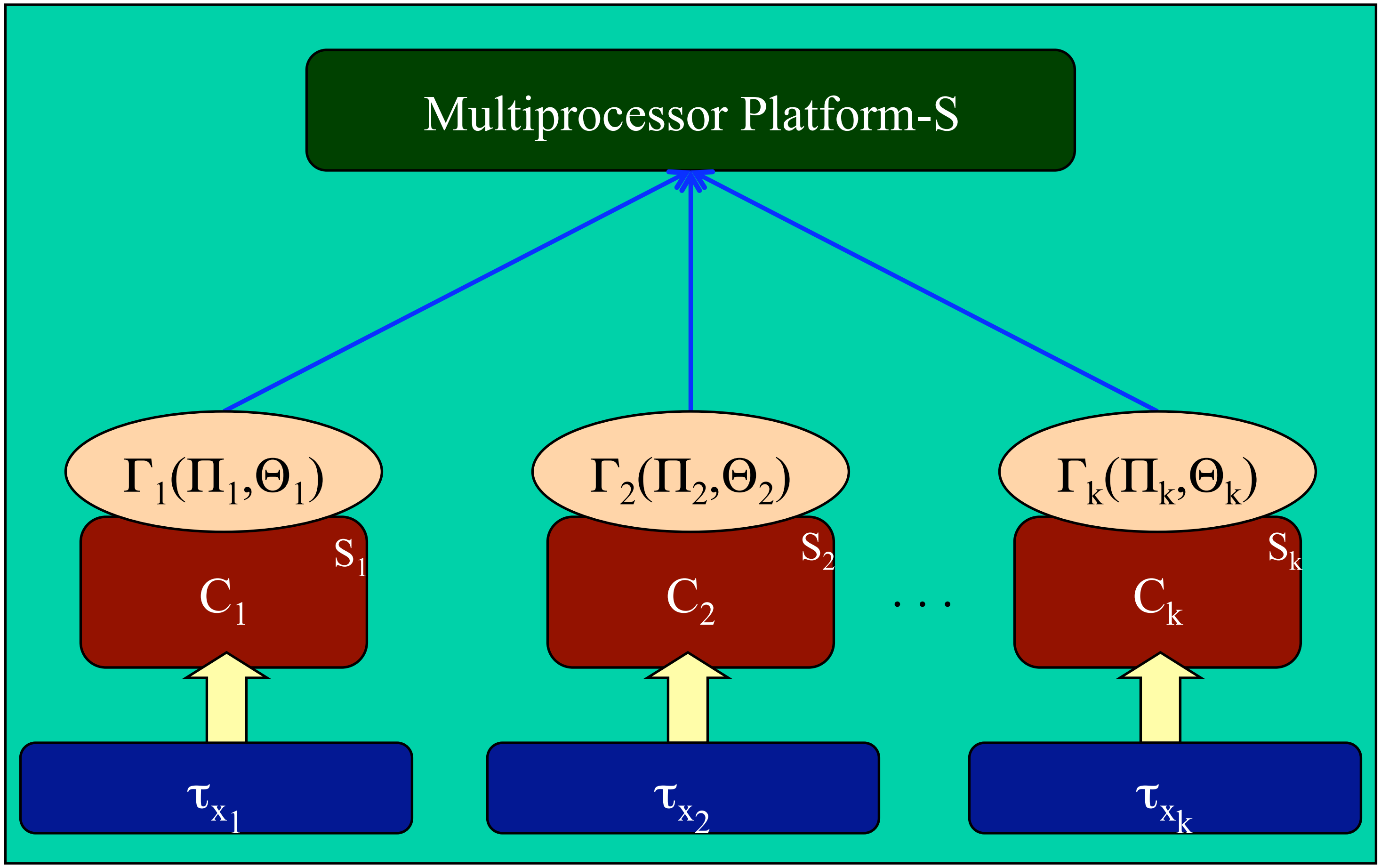


  – Cluster interface: (Γ,m)
    - Γ is the resource model, m is the maximum number of physical processors available
  – Inter-cluster scheduling is optimal

Virtual Clustering

- Two-level hierarchical scheduler
  - Intra-cluster schedulers for tasks within clusters
  - Inter-cluster schedulers for clusters on the platform (clusters can share some physical processors)
- Concurrency bound for each cluster
  - Abstract concurrency constraints of tasks within cluster
  - Minimizes overhead of schedulability analysis (e.g., Global EDF)
  - Helps regulate resource access (e.g., Caches?)
- Have virtual clusters been used before?
  - Supertasks[MoRa99], Megatasks[ACD06]
  - Results restricted to Pfair schedulers (not generalizable)

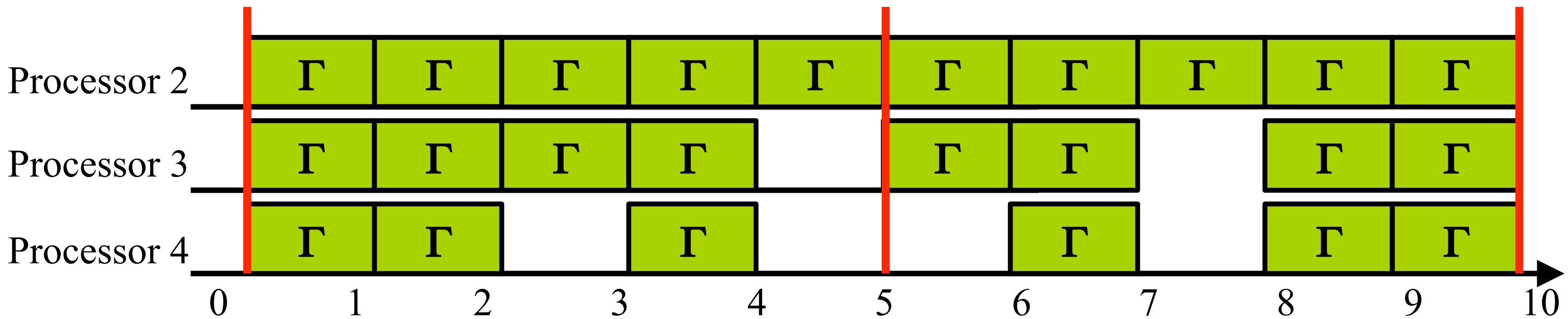
Need for Multiprocessor Periodic Resource (MPR) model



Multiprocessor Periodic Resource (MPR) model

- $\Gamma = (\Pi, \Theta, m')$
  - Θ units of resource supply guaranteed in every Π time units, with concurrency at most m' in any time instant

- Consider $\Gamma = (5, 12, 3)$



- Why MPR model?
  - Periodicity enables transformation of MPR model to periodic tasks which can be scheduled using standard algorithms

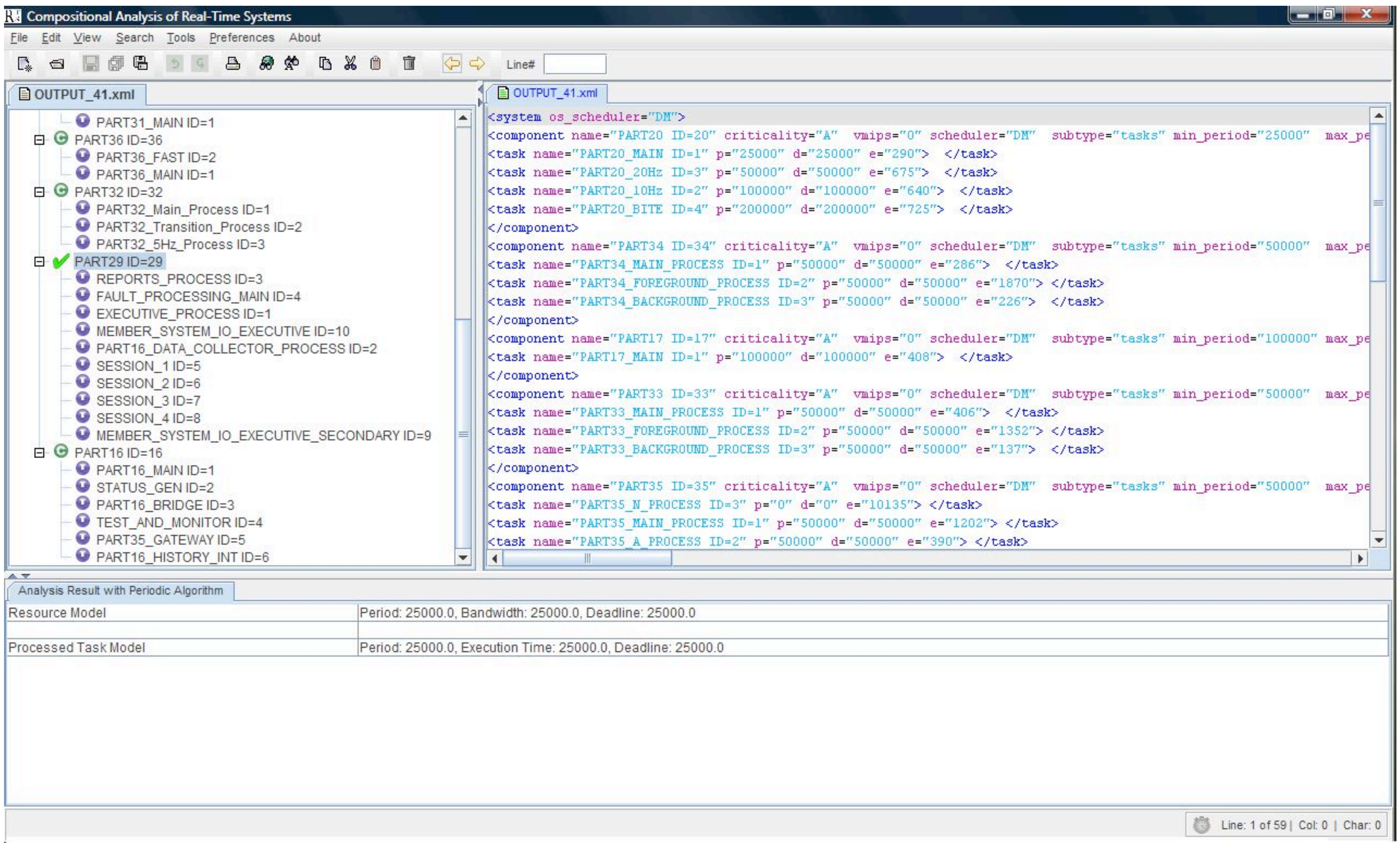
Virtual Cluster-based Scheduling

1. Split task set τ into clusters $\tau_{x_1}, \ldots, \tau_{x_k}$
2. Abstract τ_{x_i} into MPR interface Γ_i (for cluster ***VC***$_i$)
3. Transform each Γ_i into periodic tasks
   - Enables inter-cluster scheduler to schedule Γ_i

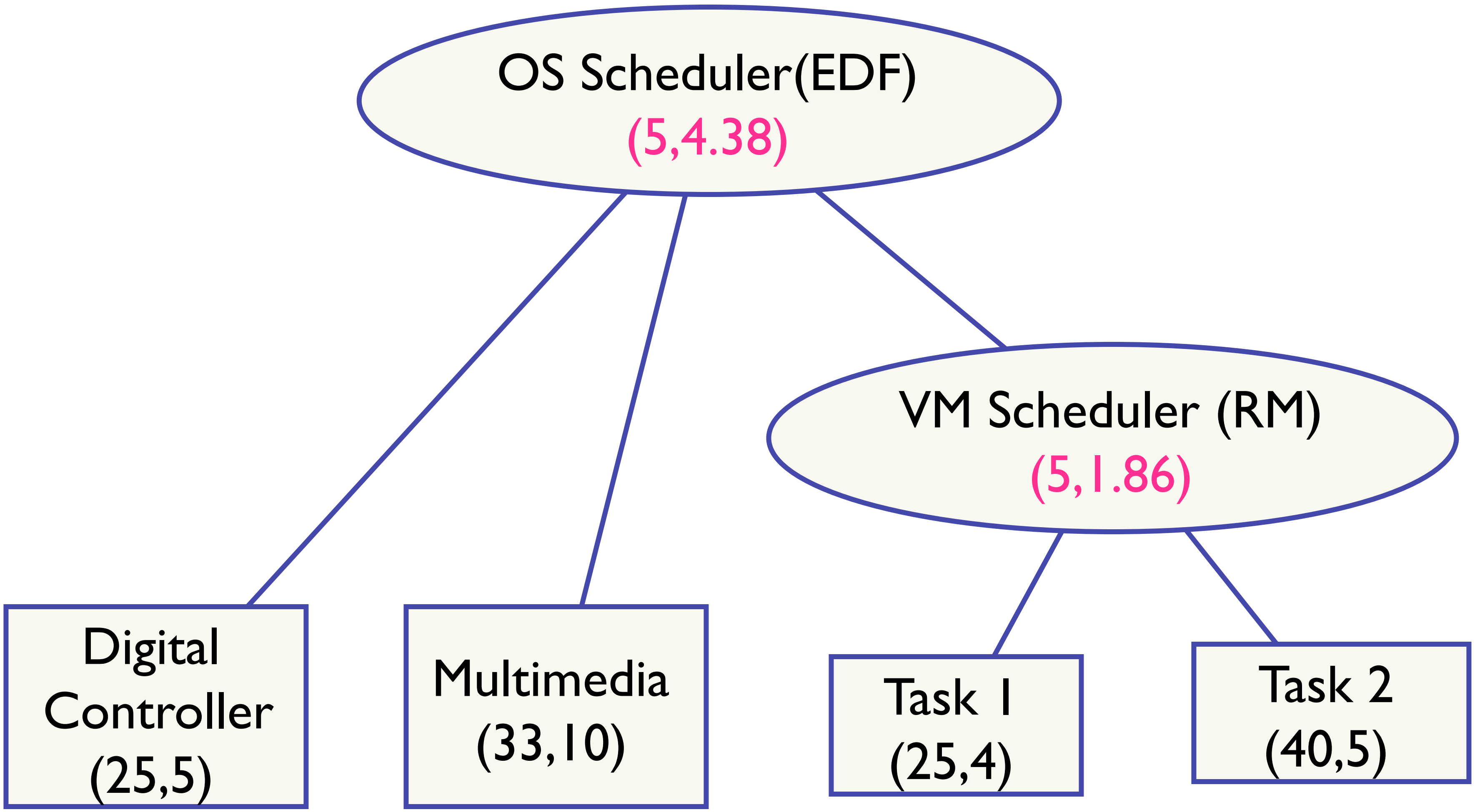
Summary on Virtual Clustering

- Virtual cluster-based multiprocessor scheduling
  - Transforms tasks from constrained to implicit deadline
    - Optimal inter-cluster scheduling techniques can be employed
  - Allows processor slack from one cluster to be used by another
  - Shows promise w.r.t. success of simple clustering techniques
- Open issues
  - Efficient clustering techniques for constrained and arbitrary deadline task systems
    - With an aim to solve the important open problem of optimal scheduling of arbitrary deadline periodic task systems
  - Including other resources such as caches

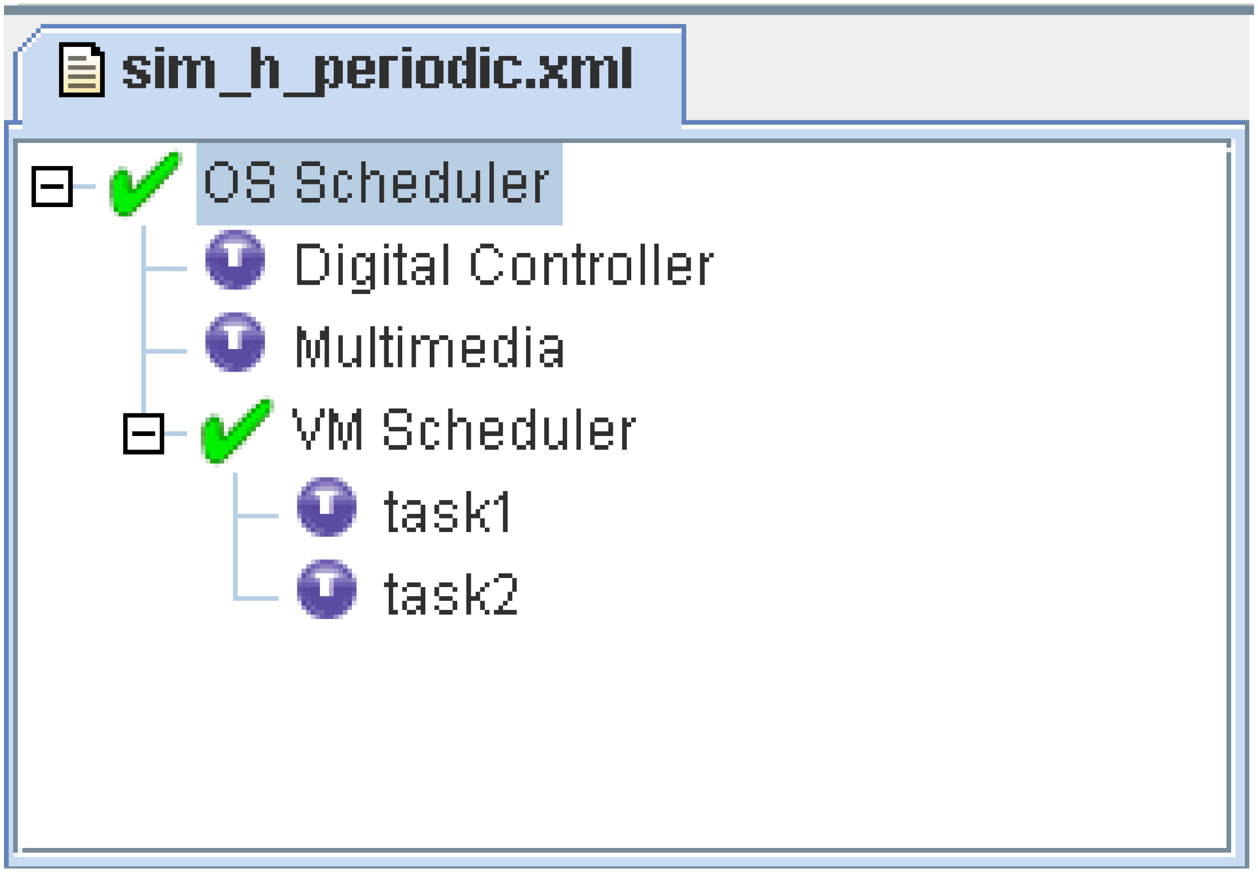
CARTS: Compositional Analysis of Real-Time Systems



Execution Demands for VM and OS Schedulers



System Modeling in CARTS



- Tree representation

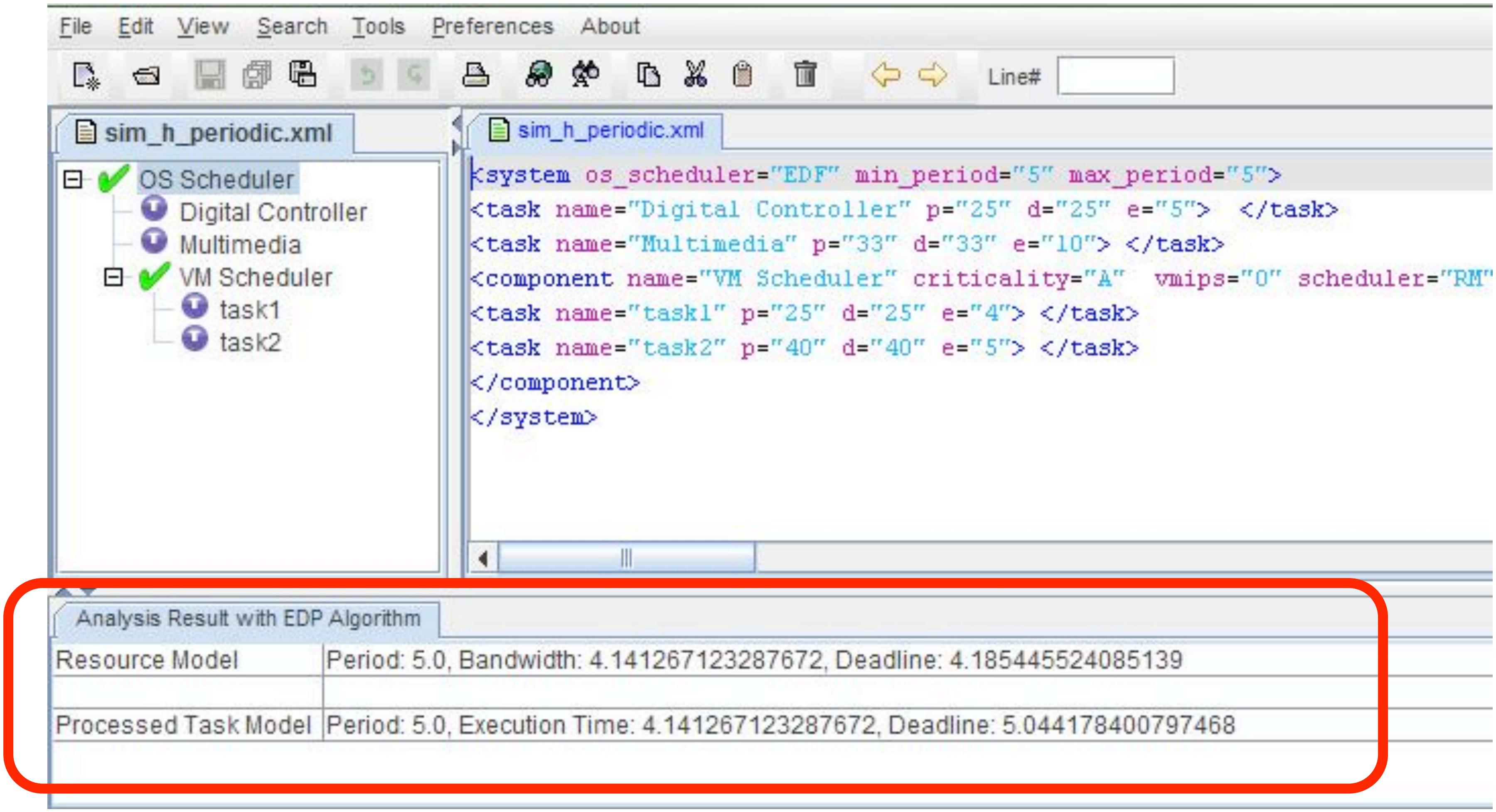
System Modeling in **CARTS**

sim_h_periodic.xml

```xml
<system os_scheduler="EDF" min_period="5" max_period="5">
<task name="Digital Controller" p="25" d="25" e="5">  </task>
<task name="Multimedia" p="33" d="33" e="10"> </task>
<component name="VM Scheduler" criticality="A"  vmips="0" scheduler="RM"  subtype="tasks" min_period="5"  max_p
<task name="task1" p="25" d="25" e="4"> </task>
<task name="task2" p="40" d="40" e="5"> </task>
</component>
</system>
```

- XML representation

Analysis in CART



CARTS

Supports

- Task & Resource Models
  - Periodic
  - Explicit Deadline Periodic (EDP)
- Scheduling Policies
  - Rate monotonic
  - Earlies Deadline First (EDF)
  - Others planned
- Open architecture

Features

- Editor for demand-supply XML files
- Tree representation of components and tasks
- Editing components/tasks in the tree
- Conversion from XML to tree representation and vice versa
- AADL output planned

Summary

- Periodic Resource Model
- Explicit Deadline Resource (EDP) Model
- Incremental Analysis
- Resource Optimality
- Virtual Clustering for Mutlicore Processors
- Toolset: CARTS
- Compositionality in Multimode Real-Time Systems
- Looking for Case Studies

References

- A Compositional Scheduling Framework for Digital Avionics Systems, Arvind Easwaran, Insup Lee, Oleg Sokolsky, Steve Vestal, to appear in RTCSA 2009.
- Optimal Virtual Cluster-based Multiprocessor Scheduling, Arvind Easwaran, Insik Shin, and Insup Lee, to be published in Real-Time Systems Journal (RTSJ).
- On the complexity of generating optimal interfaces for hierarchical systems, Arvind Easwaran, Madhukar Anand, Insup Lee, Oleg Sokolsky, Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS 2008).
- Hierarchical Scheduling Framework for Virtual Clustering of Multiprocessors, Insik Shin, Arvind Easwaran, Insup Lee, ECRTS, Prague, Czech Republic, July 2-4, 2008 (Runner-up in the best paper award)
- Robust and Sustainable Schedulability Analysis of Embedded Software, Madhukar Anand and Insup Lee, LCTES, Tucson, AZ, Jun 12-13, 2008
- Compositional Feasibility Analysis for Conditional Task Models, Madhukar Anand, Arvind Easwaran, Sebastian Fischmeister, and Insup Lee, ISORC, Orlando, Florida, May 5-7, 2008
- Compositional Real-Time Scheduling Framework with Periodic Model, Insik Shin and Insup Lee, ACM Transactions on Embedded Computing Systems (TECS), vol 7, no 3, April 2008
- Interface Algebra for Analysis of Hierarchical Real-Time Systems, Arvind Easwaran, Insup Lee, Oleg Sokolsky, Foundations of Interface Technologies (FIT) workshop, Budapest, Hungary, April 5, 2008
- Compositional Analysis Framework using EDP Resource Models, Arvind Easwaran, Madhukar Anand, and Insup Lee, Tucson, Arizona, Dec 3-6, 2007
- Resources in process algebra, Insup Lee, Anna Philippou, Oleg Sokolsky, Journal of Logic and Algebraic Programming, Vol. 72, pp. 98-122,2007

This work was supported in part by AFOSR, NSF and ONR.

Thank You!

Questions?