Interface Algebra for Analysis of Hierarchical Real-Time Systems

Arvind Easwaran, Insup Lee,
Oleg Sokolsky

University of Pennsylvania

FIT 2008

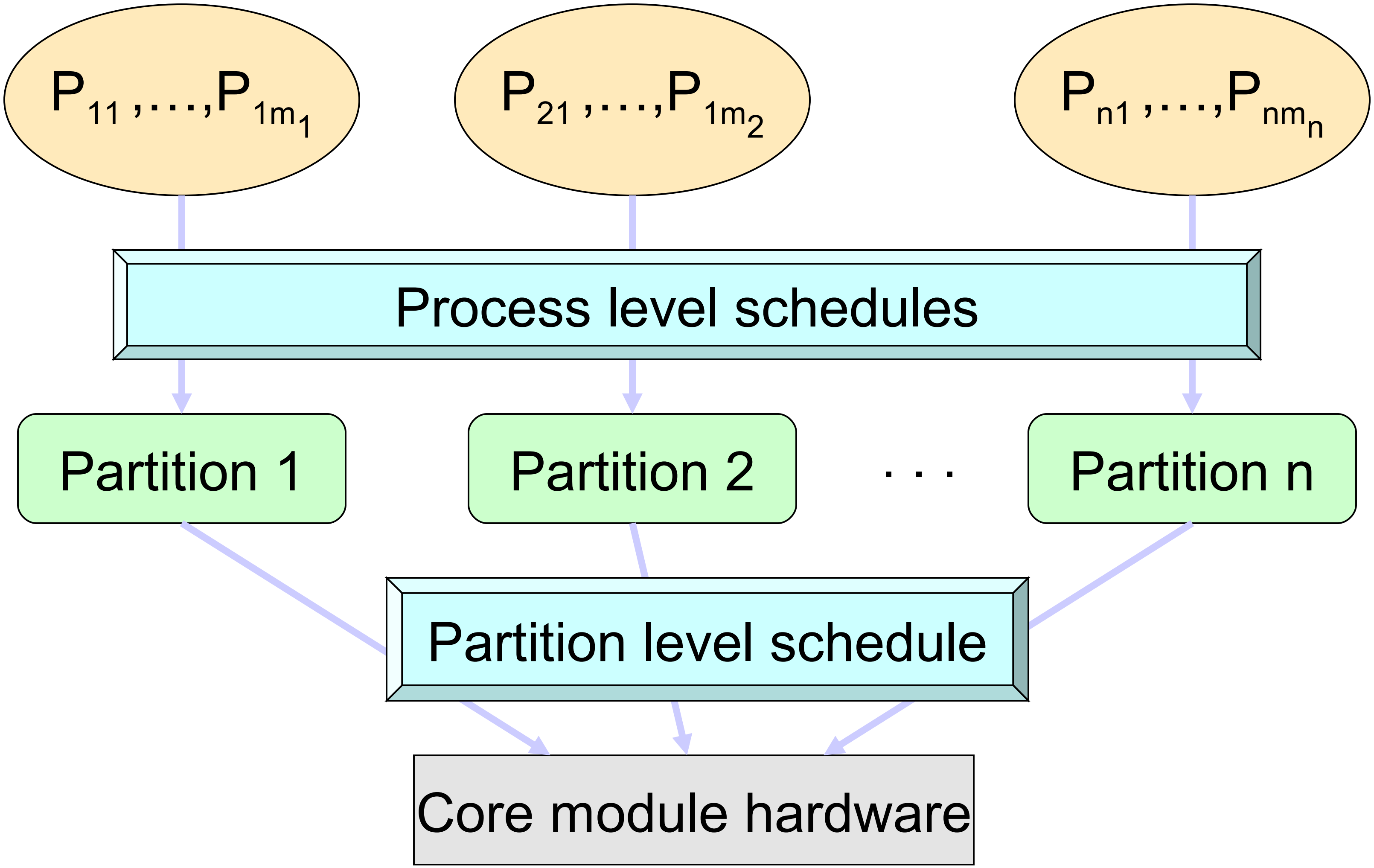
Motivation

- Application domain: embedded systems
  - complex, networked, and large-scale
- Important features of the domain:
  - module development followed by integration
  - rapid development cycles, module reuse
  - resource constraints are critical
- Component-based development helps contain complexity
- Goal: resource-centric component framework

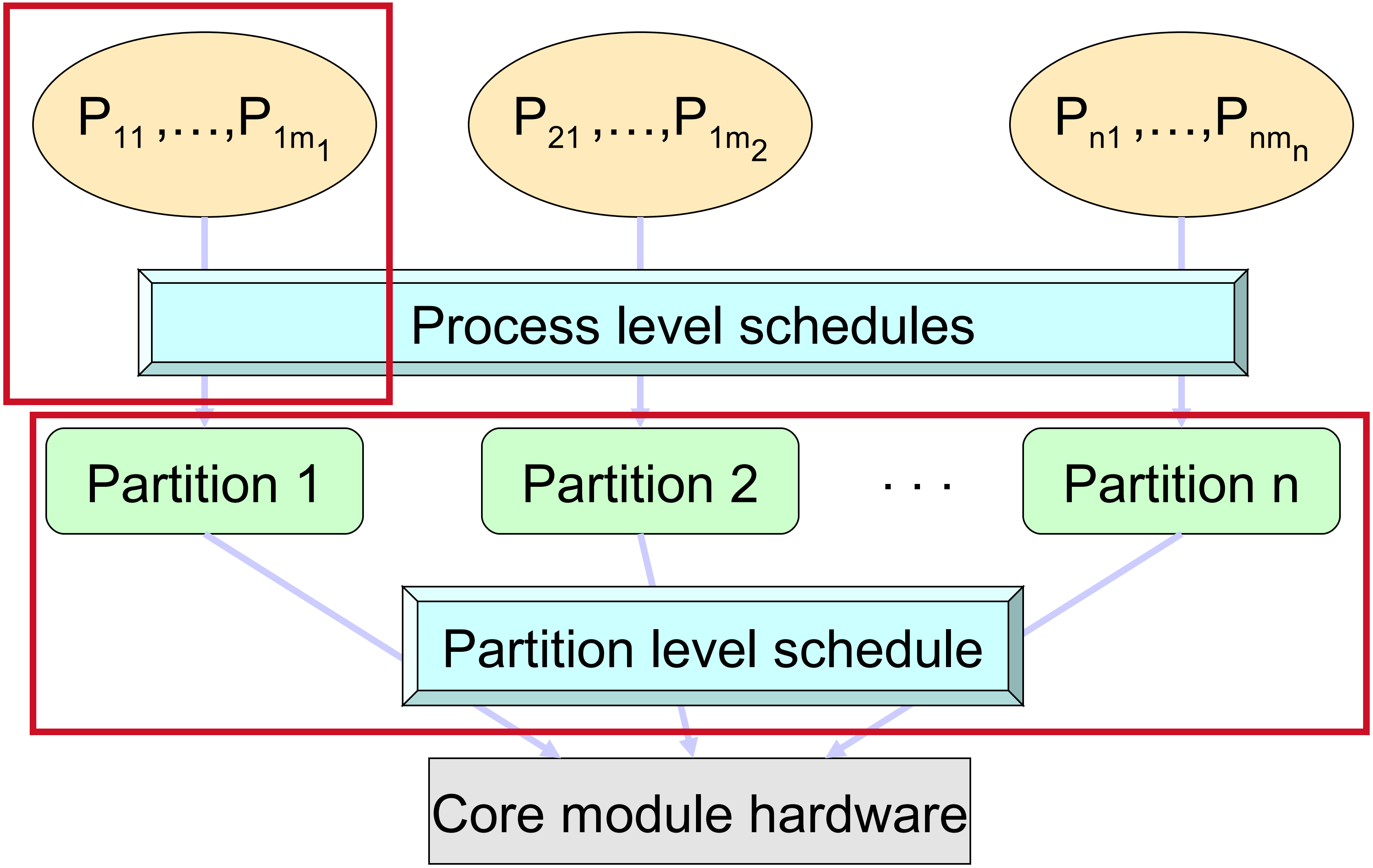
Component technologies

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

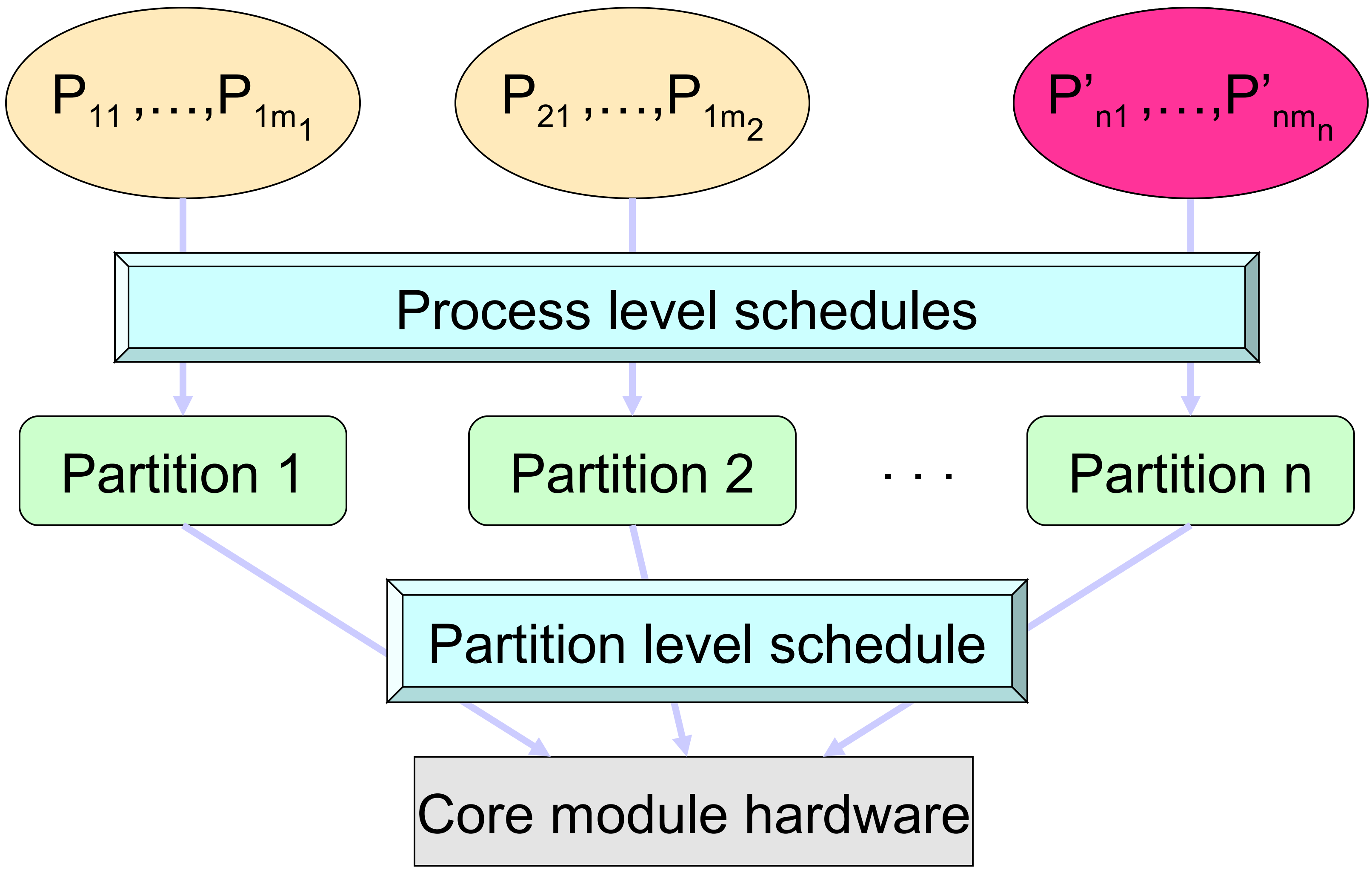
Motivating example: ARINC 653



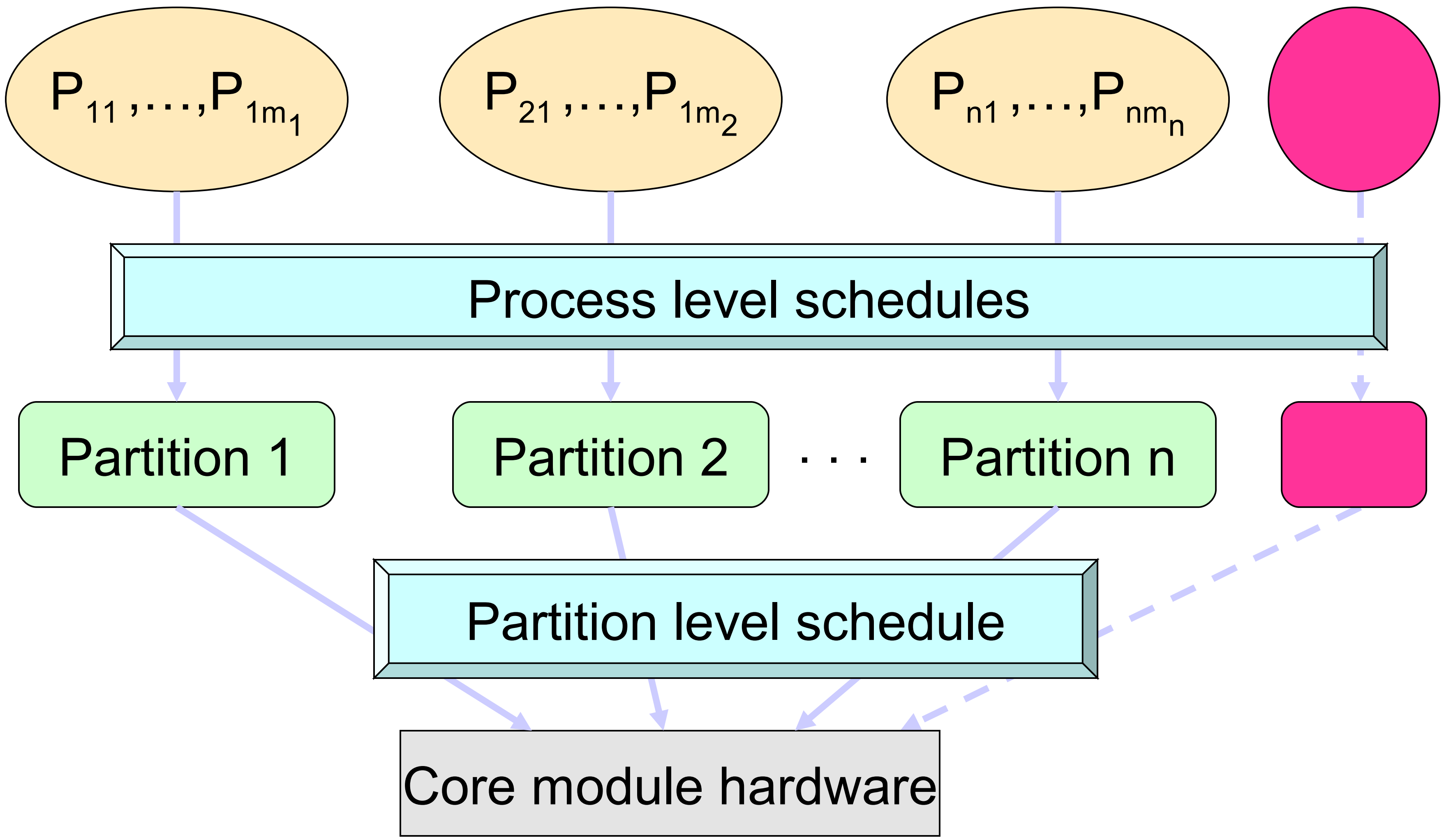
ARINC 653: Schedulability



ARINC 653: Refinement



ARINC 653: Incremental analysis



Real-time Components

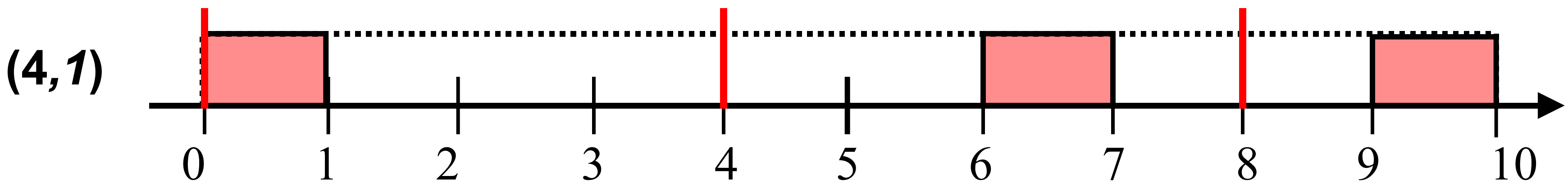
- Workload
  - Primitive: periodic or sporadic tasks
  - Composite: other components
- Scheduling algorithm
  - Earliest deadline first (EDF)
    - Always, job with earliest deadline executes
  - Deadline monotonic (DM) → Assume $D \leq T$
    - Always, job with smallest deadline executes

ARINC 653 Partition → Component

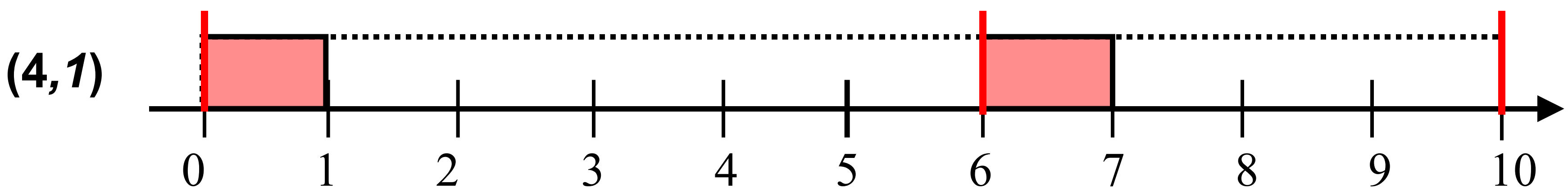
ARINC 653 Process → Periodic task

Real-Time Workload

- Set of real-time jobs with hard deadlines
- Periodic task specification: T = (p,e)

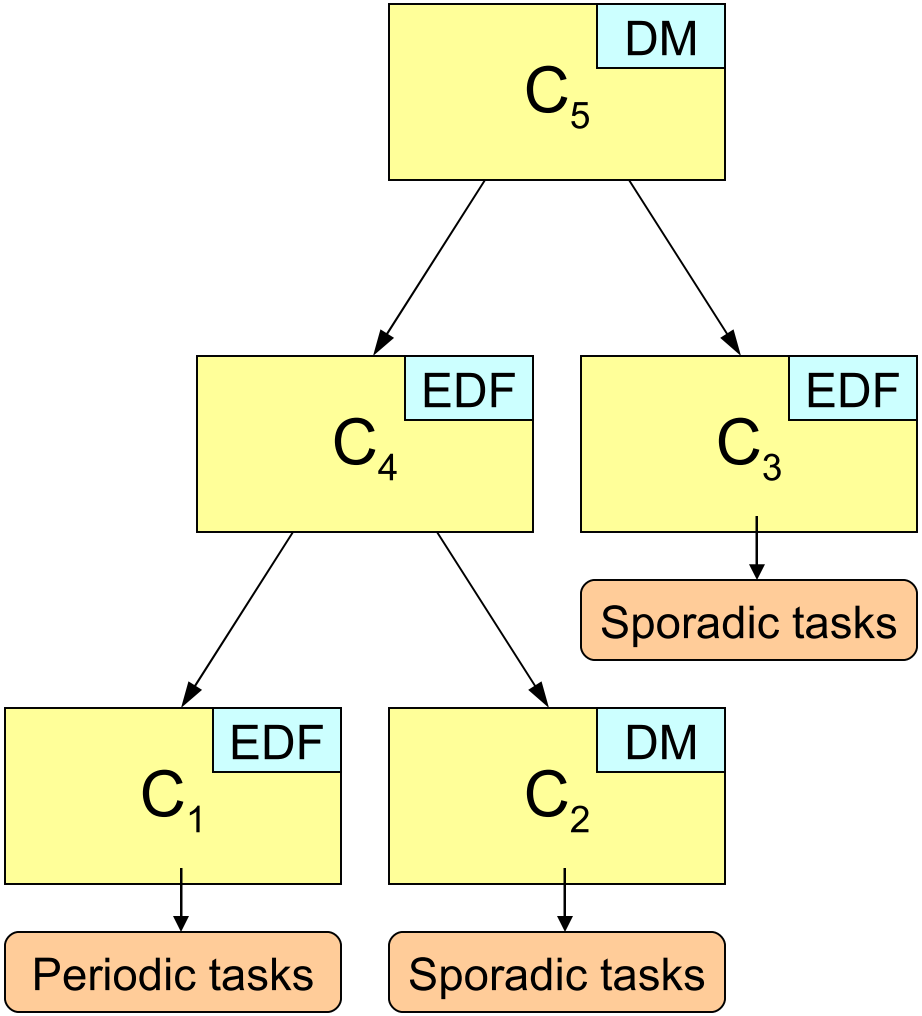


- Sporadic task specification: T = (p,e)



- In general: workload depends on contents of the component and scheduling algorithm

Hierarchical Scheduling Framework

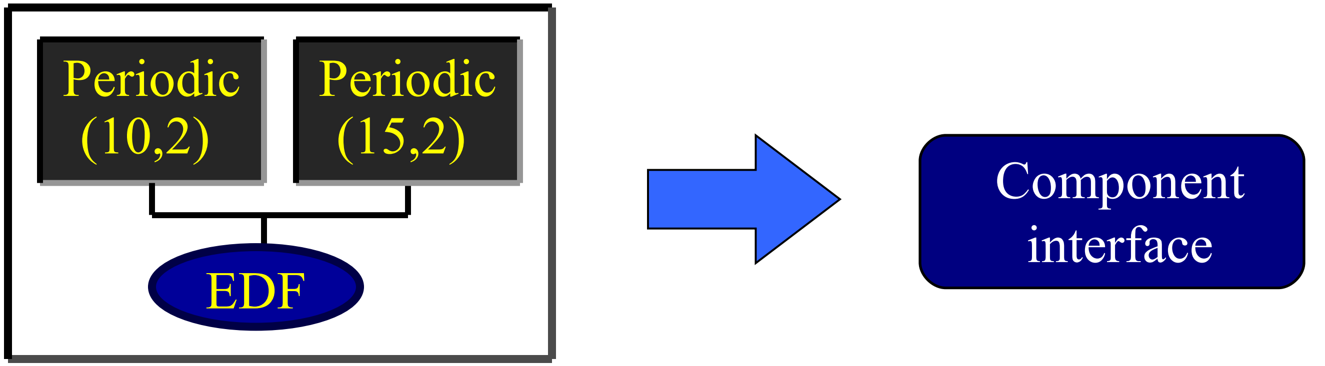


- Resource allocation from parent to child
- Notations
  - Leaf → C_1, C_2, C_3
  - Non-leaf → C_4, C_5
  - Root → C_5

ARINC 653 → Two-level hierarchical framework

Abstraction and Composition

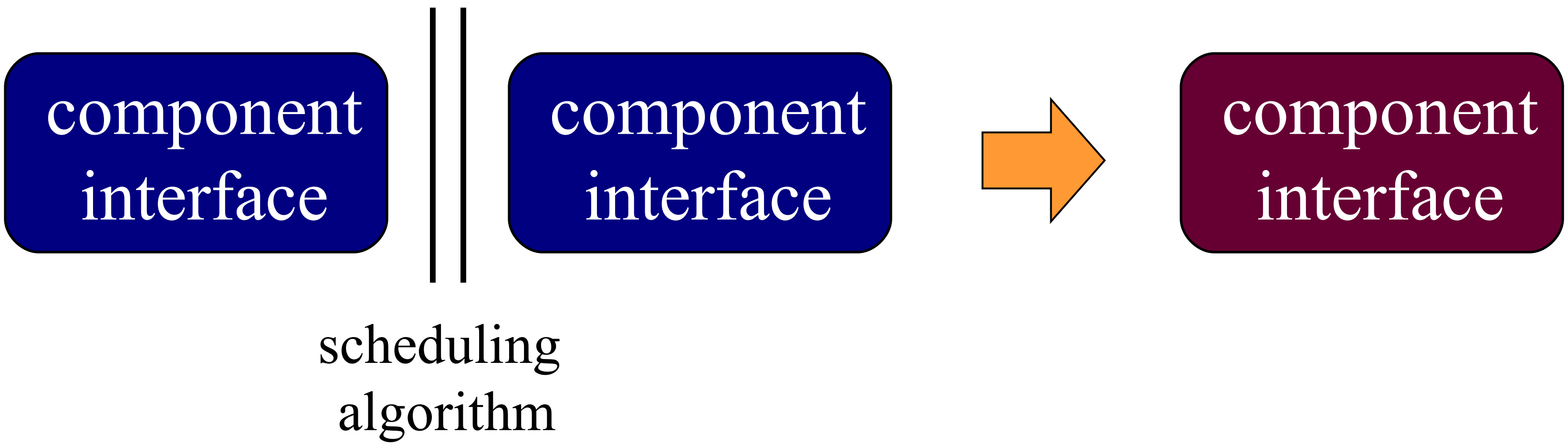
- Abstraction Problem: abstract the real-time application as a component with an interface



- 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



Demand Bound Function

- Characterizes resource demand
  - $dbf_W(t)$ is the maximum possible resource demand during a time interval of length t
- Used in schedulability analysis
  - W is schedulable on a resource R if

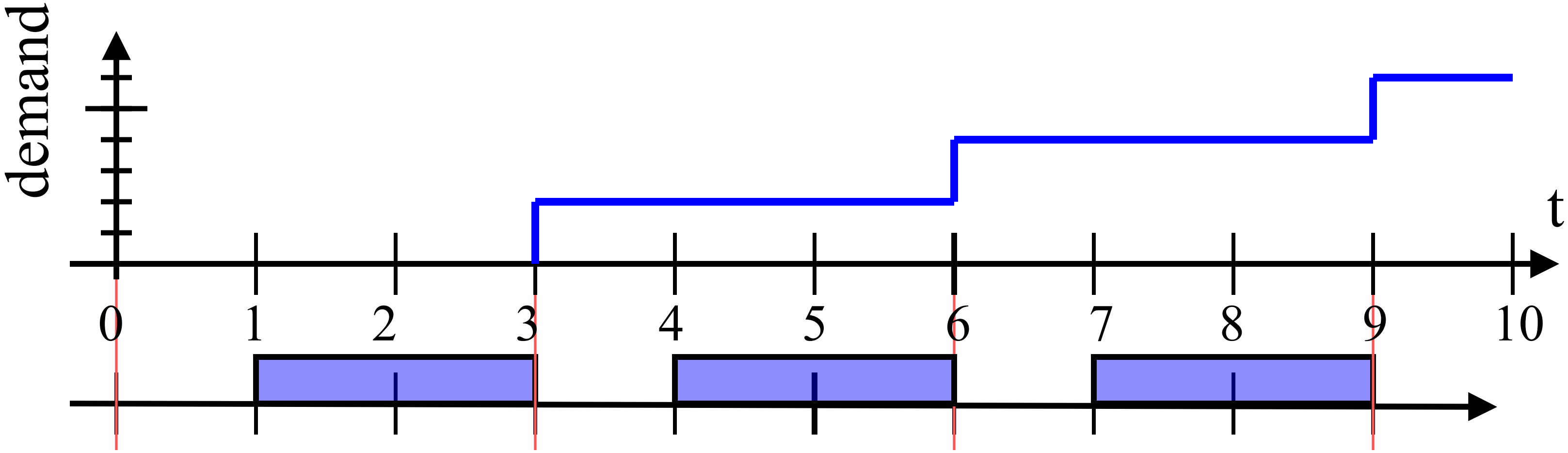
$$\forall t. dbf_W(t) \leq sbf_R(t)$$

  - $sbf_R(t)$: supply bound function
    - defined similarly to DBF

DBF of a single task

- Periodic task model T(p,e) [Liu & Layland, '73]
  - period p and execution time e
  - Ex: T(3,2)

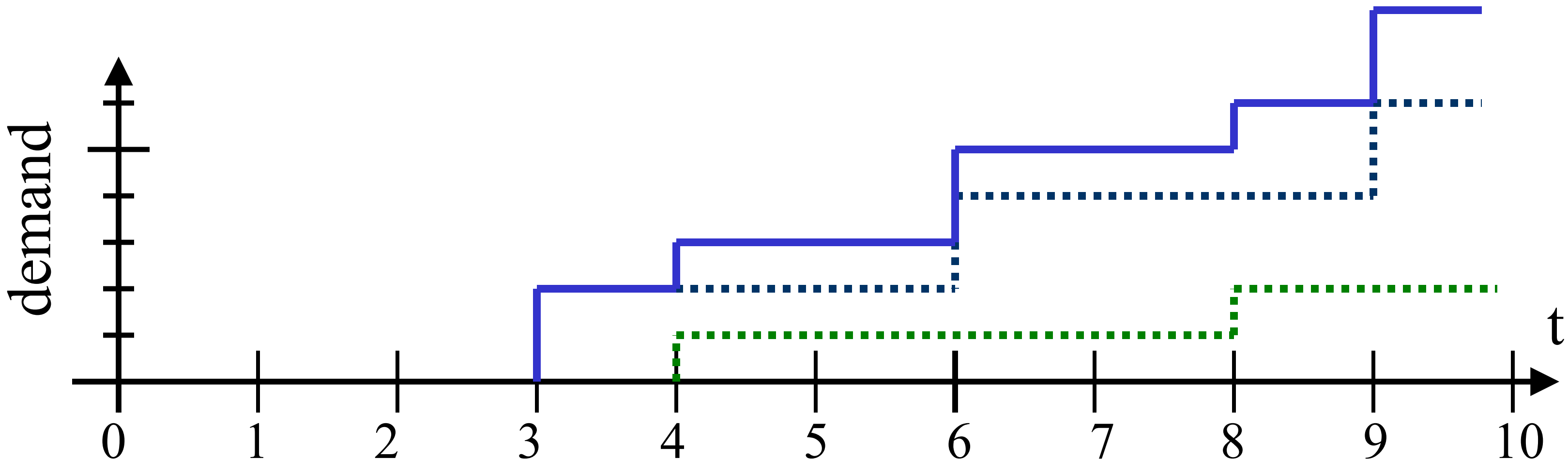
$$dbf_T(t) = \left\lfloor \frac{t}{p} \right\rfloor e$$



Demand Bound - EDF

- Periodic workload set *W* = {Ti(pi,ei),EDF},
  - $dbf_W(t)$ [Baruah et al.,'90]

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



Demand Interface

- General abstraction scheme for real-time workloads
- Specification D = <S,P,O>
  - S $\rightarrow$ Scheduling policy of interface
  - P = {P1,… , Pk} $\rightarrow$ Disjoint restrictions on output functions s.t. for each i, Pi $\subseteq$ DS
  - O = {O1,… , Ok} $\rightarrow$ Set of output functions (dbfs or sbfs) such that for each i, Oi $\in$ Pi
- Periodic or Sporadic Task T
  - DT = <FP, {PT = DS}, {OT = dbfT}>

Multiple outputs allow choice for abstraction

Interface Composition

- Technique to compositionally generate interfaces for real-time components
  - Generates demand interface for a component using interfaces of its workload
- For a component
  - Interface scheduler = component scheduler
  - Output restrictions are fixed by system designer
  - Composition generates outputs satisfying these restrictions

Property of Compositionality

- Requirement for interface composition:

  If the generated output is schedulable by some resource model, then workload outputs that were composed must also be schedulable by the same resource model under component's scheduler

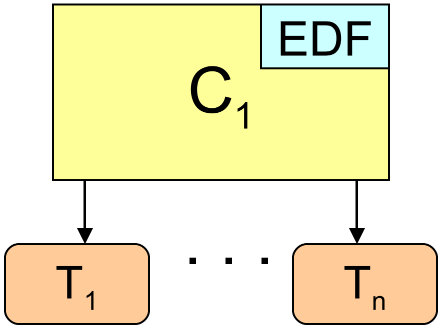
- Provided by existing schedulability conditions

Composition Process

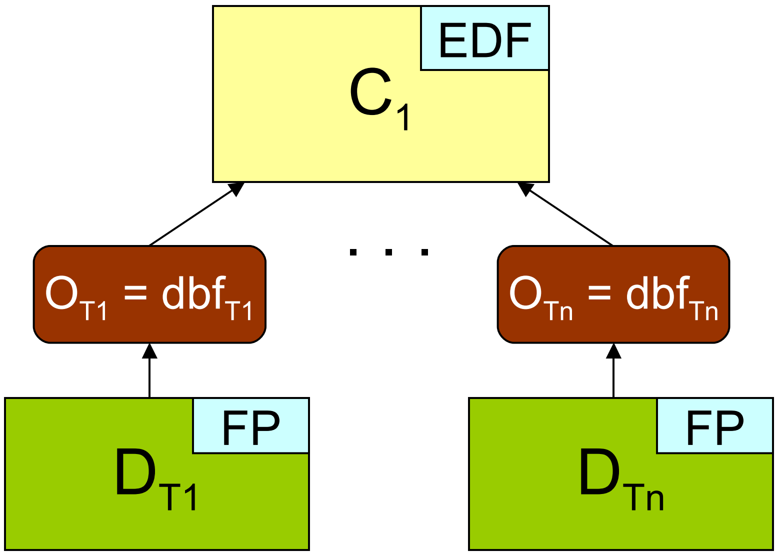
- Repeat the following steps for each output restriction in component interface
  - Choose one output from each workload interface
  - Compose the chosen outputs to generate output for component interface

- Each generated output
  - Satisfies compositionality as defined earlier
  - Satisfies corresponding output restriction

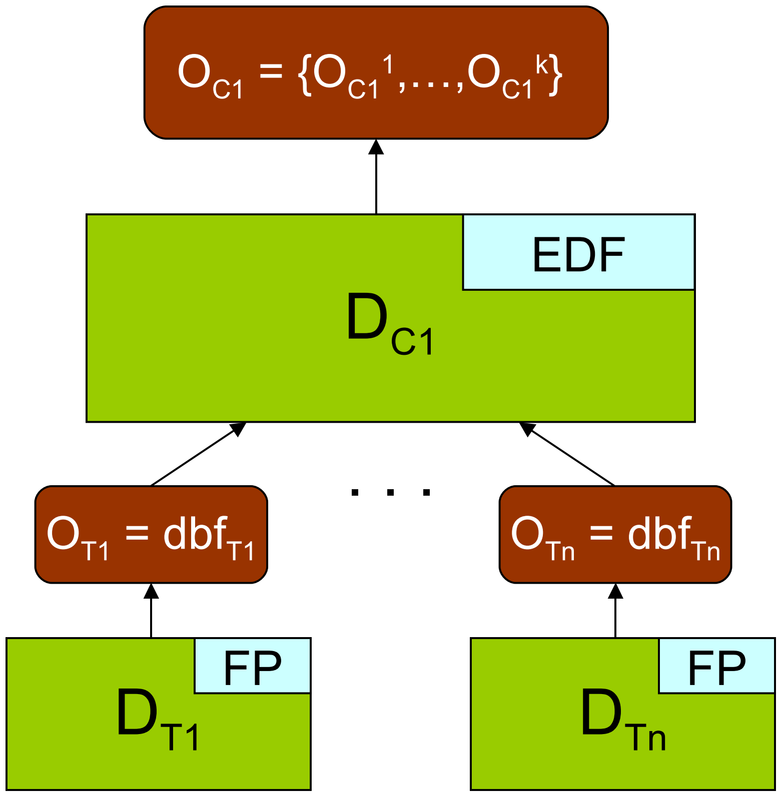
Example Composition (leaf)



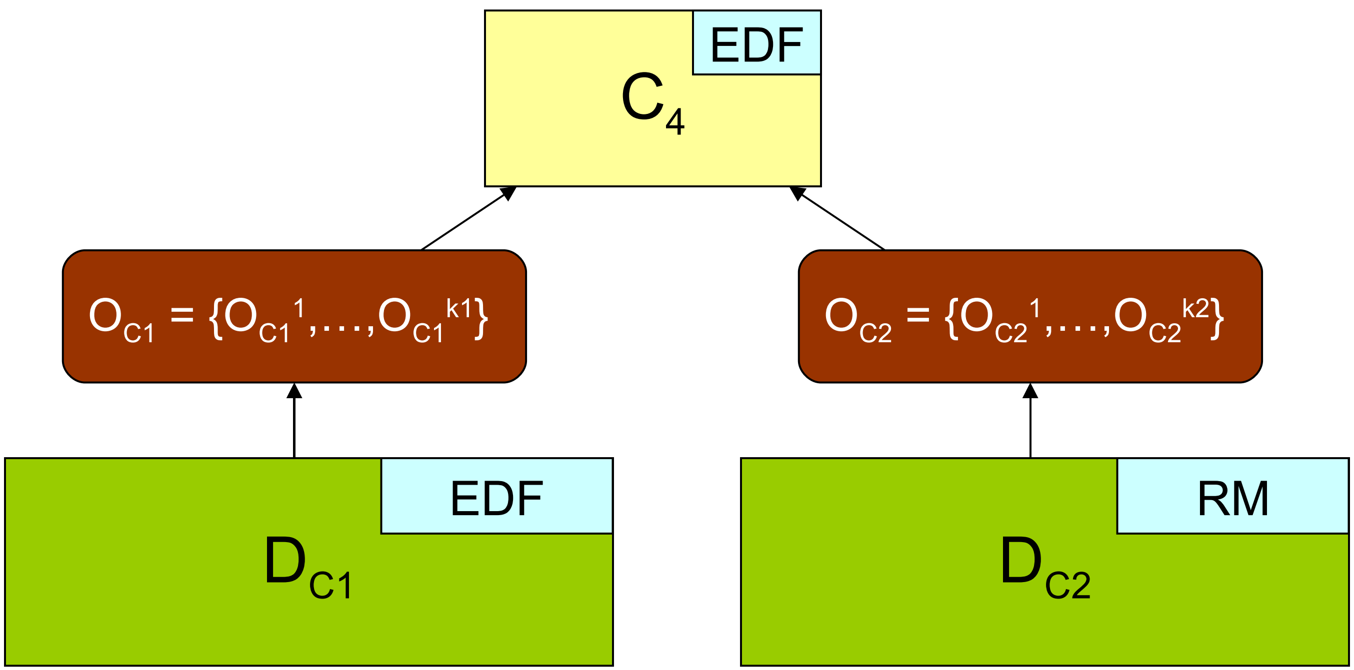
Example Composition (leaf)



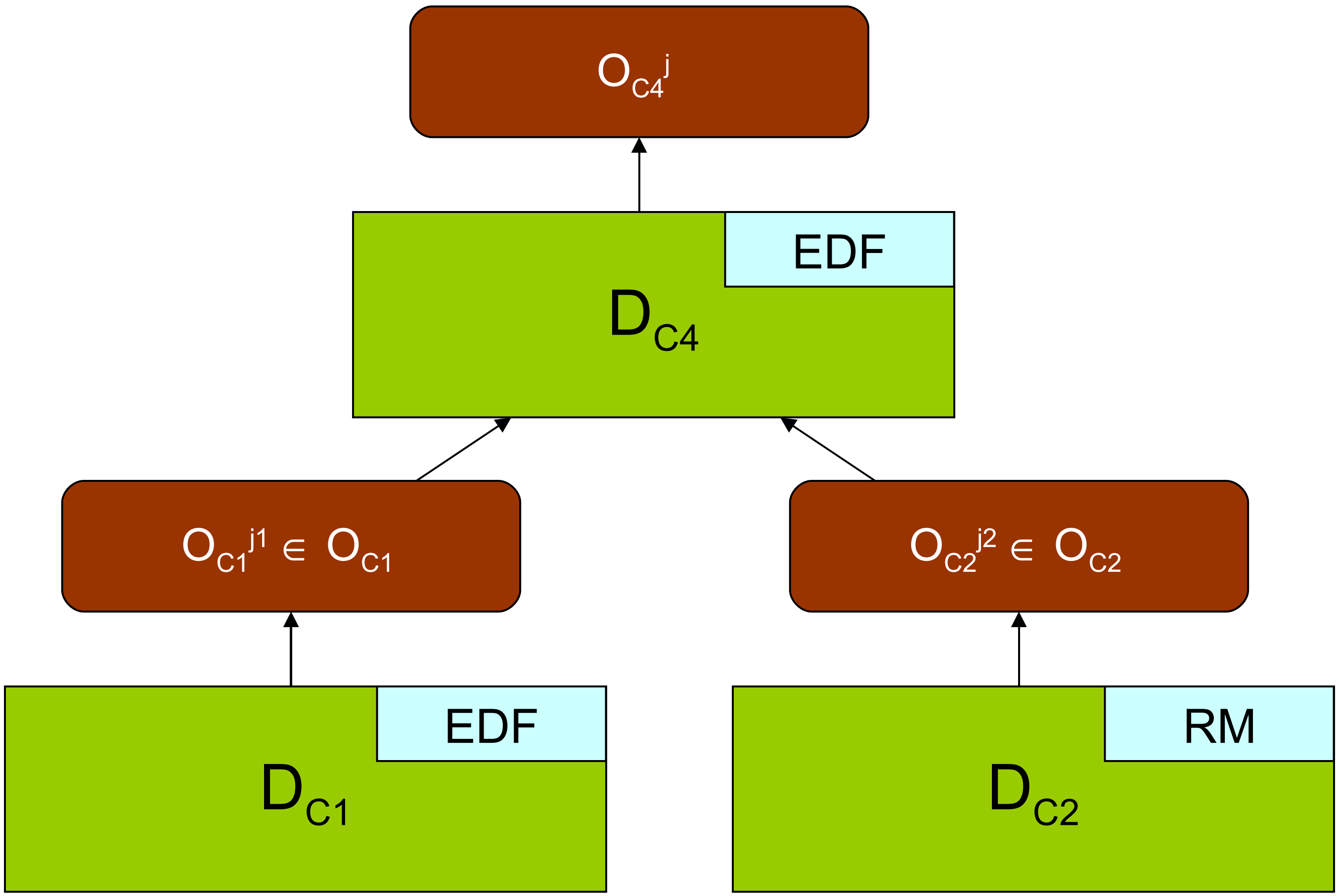
Example Composition (leaf)



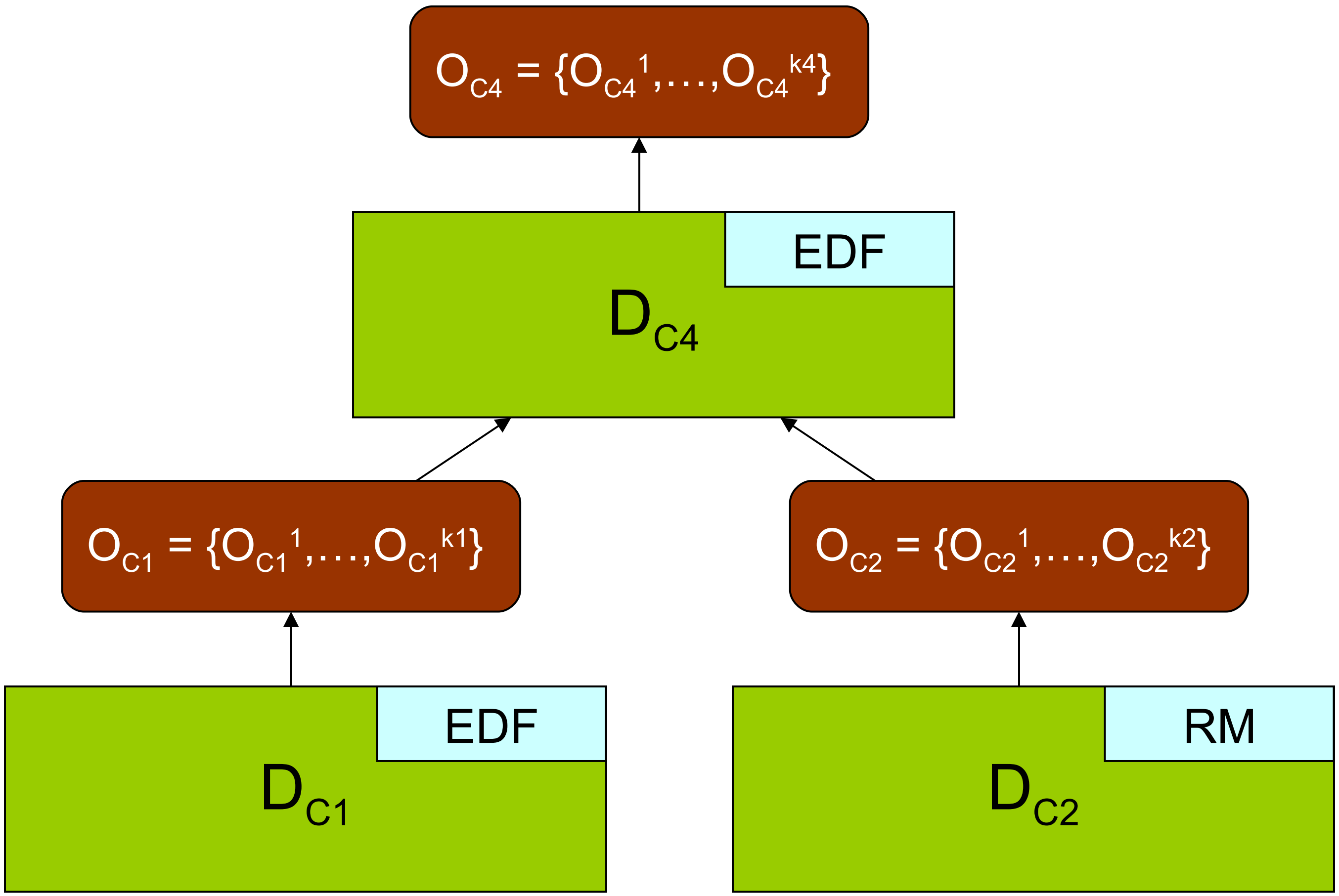
Example Composition (non-leaf)



Example Composition (non-leaf)



Example Composition (non-leaf)

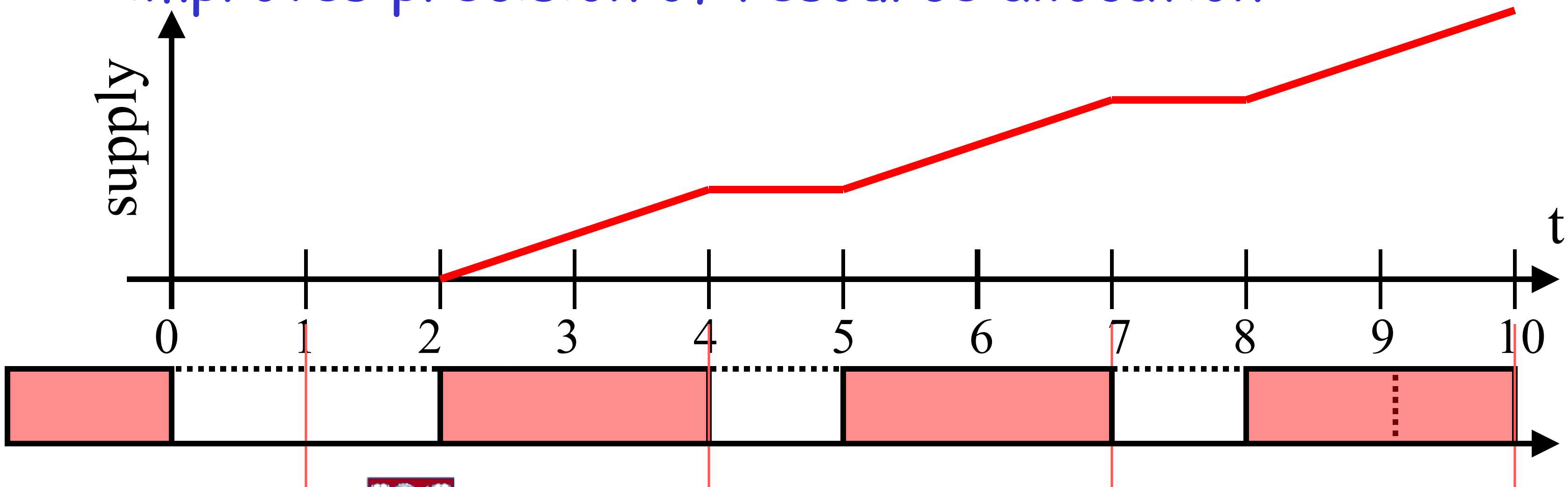


Framework Instantiation

- Definitions of compositionality and abstraction depend on the choice of a schedulability analysis technique
  - Specifically, a resource model
- Choice of a resource model determines
  - Types of outputs of the interface
  - Parameters of outputs
- Output restrictions are constraints on parameters of outputs

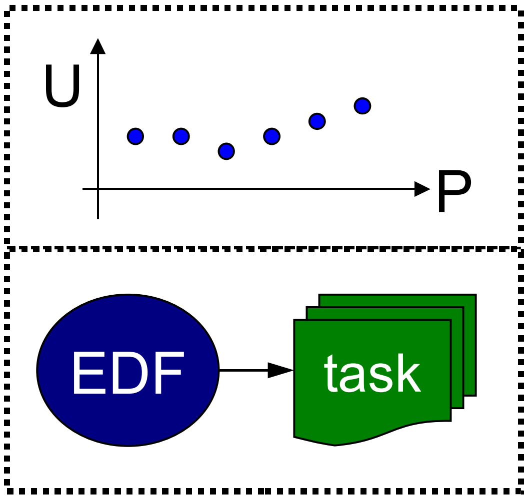
Resource Modeling

- Bounded-delay resource model [Mok et al., '01]
  - time-sharing resource w.r.t. a dedicated resource
- Periodic resource model Γ(Π,Θ) [ShinLee, '03]
  - characterizes periodic resource allocations
- EDP model [Easwaran et al., '07]
  - improves precision of resource allocation



Component Interface

- An output is (Π,Q), such that
  - Γ(Π,Q/Π) is an optimal resource model dominating dbf_W
- Output set covers range of periods
  - { (Π,Q) | 1 ≤ Π ≤ Π* }
    - Π* = LCM or can be user-defined



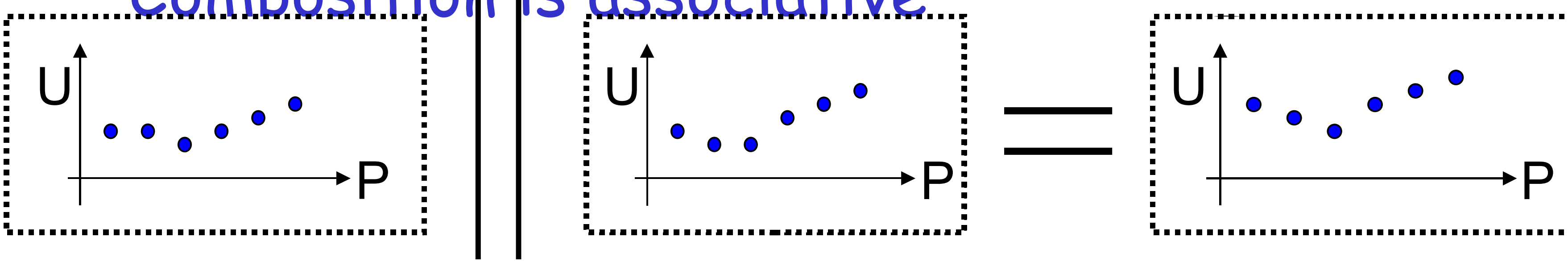
Interface Composition

- $P_C = \{\Pi=1,\ldots,\Pi=k\}$
- Abstraction function:
  - $A_{EDF,k} = \{\ (i, Q_i^C)\ \}_{i=1..k}$

    Such that given $(i, Q_i^{C_1}) \in O^{C_1}, (i, Q_i^{C_2}) \in O^{C_2}$

    $$Q_i^C = Q_i^{C_1} + Q_i^{C_2}$$

  - Composition is associative

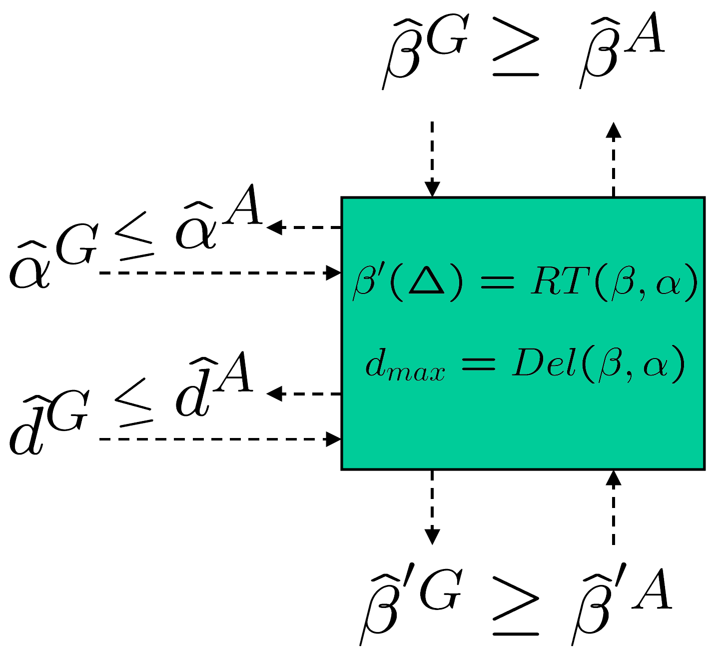


Related work

- Much work on hierarchical scheduling
  - schedulability conditions that are needed for instantiation
  - basis for abstraction and composition
    - Shin and Lee, '03 '04, Easwaran et al., '06
- Real-time interface frameworks
  - Henzinger and Matic, '06
  - Wandeler and Thiele, '06
- Behavioral timing interfaces
  - Primarily for future work

Related work

- Assume-guarantee interfaces with RT calculus
- Explicit representation of arrival and resource curves
- Target stream-processing systems



Adding behavioral information

- Stream interface?
  - Data does not change system state
  - Infrequent control events change state
  - Arrival curves for both? Problematic!
- Mode interface!
  - Mode automata
    - System modes as states
    - Mode switches by (infrequent) control events
  - Interface outputs can be mode-specific
    - Composition: mode product + mode schedulability

Conclusions

- Interface framework for real-time system
- Based on hierarchical schedulability analysis
- Supports
  - Independent implementation of components
  - Interface-based component composition
  - Component refinement
  - Incremental composition
- Instantiates to a variety of schedulability analysis methods