ARTIST2 Summer School 2008 in Europe
Autrans (near Grenoble), France
September 8-12, 2008

Multiprocessor real-time computing: formal foundations

Invited Speaker: Sanjoy Baruah
The University of North Carolina at Chapel Hill

Information Society

Multiprocessor real-time computing: formal foundations

Why multiprocessors?

- provide greater computing capacity, at lower cost
- many real-time applications are inherently parallelizable
- uniprocessor systems are becoming obsolete

-(multicore CPU's)

Goal: A theory of multiprocessor real-time scheduling

Outline of presentation

1. a multi-layered perspective
2. background and context
3. an illustrative example

A multi-layered perspective

Layer 1. Techniques & concepts for solving multiprocessor generalizations of uniprocessor problems

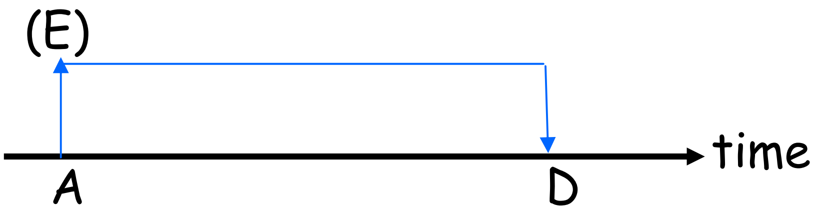
Layer 2. Metrics for quantifying multiprocessor problems

Layer 3. Task and machine models for representing multiprocessor systems

Task model

Jobs: basic units of work. J = (A, E, D)

- Preemptable
- Not parallelizable



Recurring tasks or processes

- finite (a priori known) number of them
- generate the jobs
- represent code within an infinite loop
- different tasks are assumed independent

The Liu & Layland task model

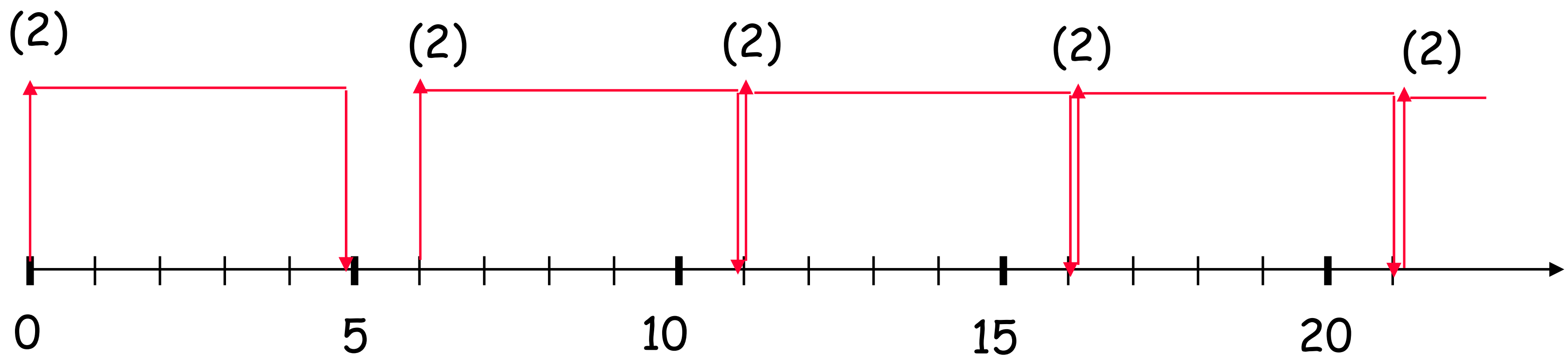
Task τ = (e,p)

- execution requirement
- minimum inter-arrival separation ("period")

Jobs

- first job arrives at any time
- consecutive arrivals $\geq$ p time units apart
- each task has execution requirement $\leq$ e
- each job has its deadline p time units after arrival

Example: τ =(2,5)



The sporadic task model

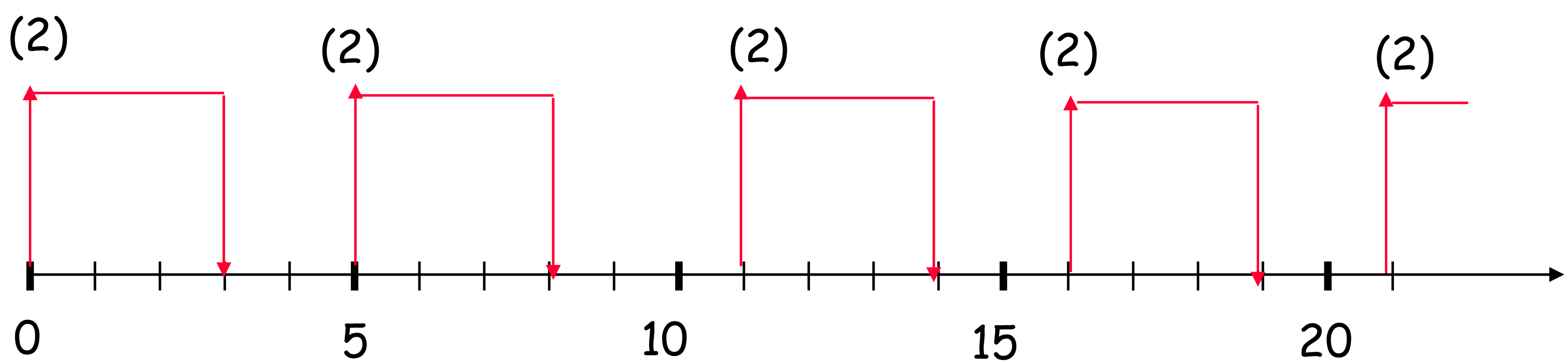
Task τ = (e,d,p)

- execution requirement
- relative deadline
- minimum inter-arrival separation ("period")

Jobs

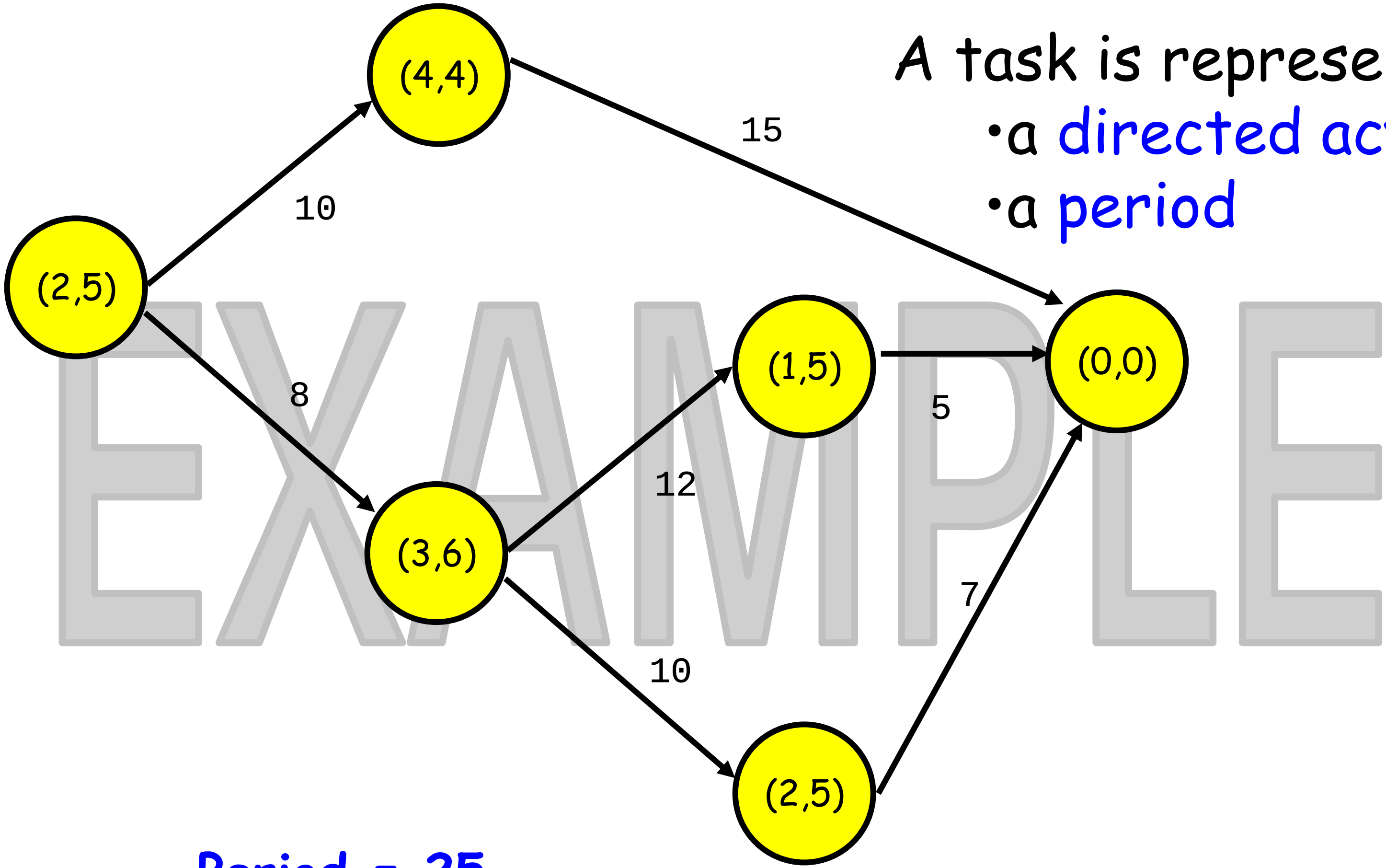
- first job arrives at any time
- consecutive arrivals $\geq$ p time units apart
- each task has execution requirement $\leq$ e
- each job has its deadline **d** time units after arrival

Example: τ =(2, 3, 5)



A DAG-based task model

A task is represented by
- a directed acyclic graph
- a period

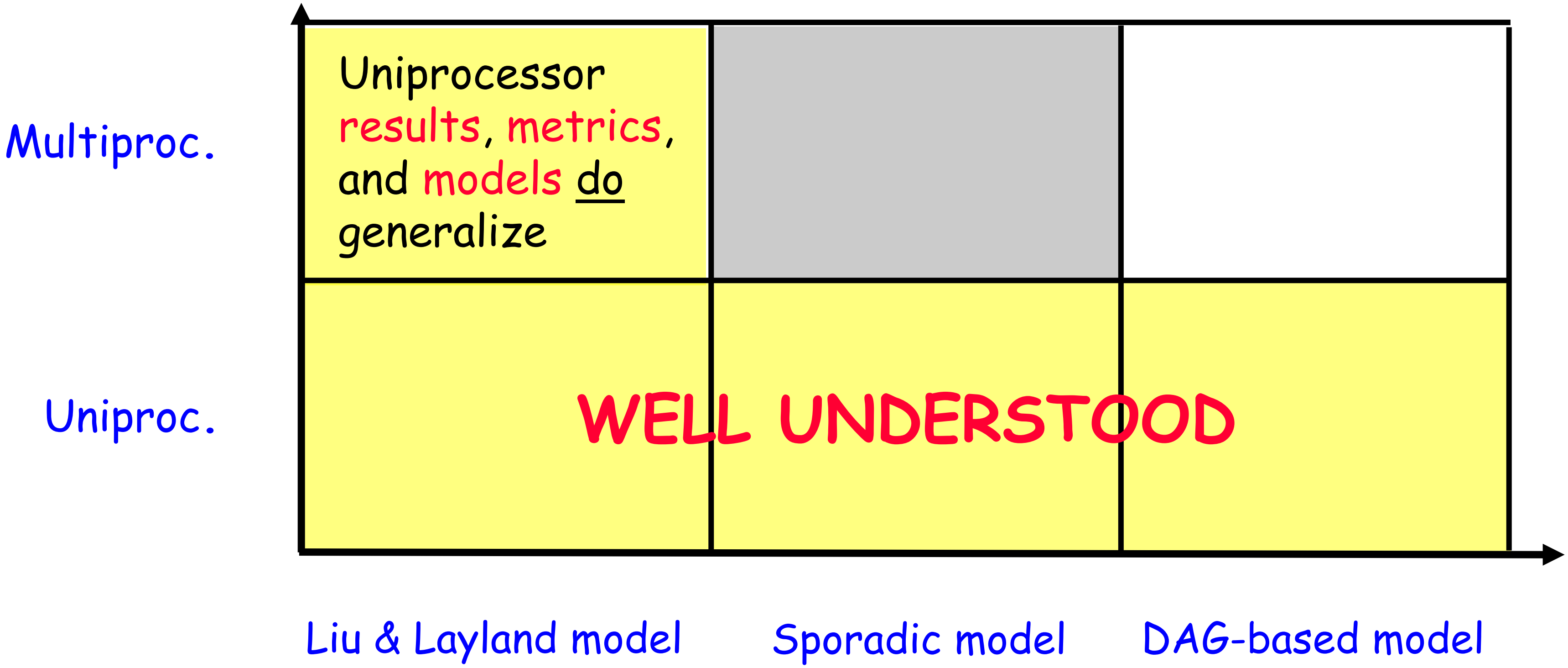


Period = 25

Multiprocessor scheduling: the current landscape

Example: Feasibility analysis of systems of sporadic tasks:

Can a specified system be scheduled to always meet all deadlines?



The multi-layered perspective

Layer 1. Techniques & concepts

Layer 2. Metrics

Layer 3. Task and machine models

Layer 1: Techniques and concepts

Example: Feasibility analysis of systems of sporadic tasks

INTUITIVELY APPEALING!!

On uniprocessors:

1. identify worst-case arrival sequence
   - each τ_i generates one job at t=0; subsequent arrivals exactly p_i time-units apart. **(SYNCHRONOUS ARRIVAL SEQUENCE)**

2. validate its schedulability
   - EDF is an optimal preemptive uniprocessor scheduling algorithm

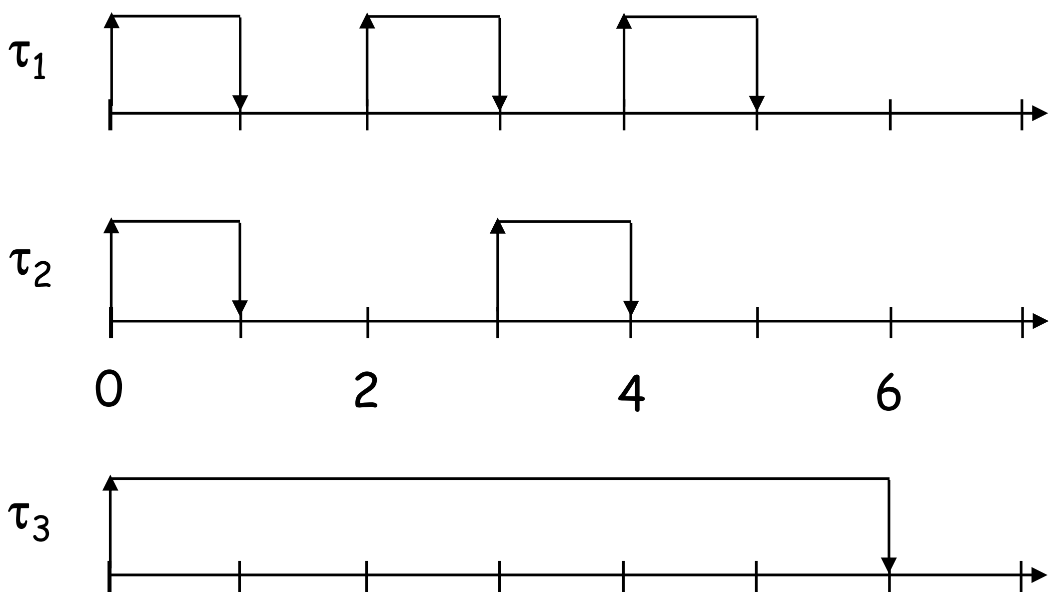
Feasibility analysis algorithm: Simulate EDF on the synchronous arrival sequence until (at most) lcm { p_1, p_2, …, p_n}

Layer 1: Techniques and concepts

RESULT: The Synch. Arrival Sequence not worst-case arrival sequence

$\tau_1 = (1,1,2)$, $\tau_2 = (1,1,3)$, $\tau_3 = (5,6,6)$, 2 Procs

$\tau_i = (e_i, d_i, p_i)$



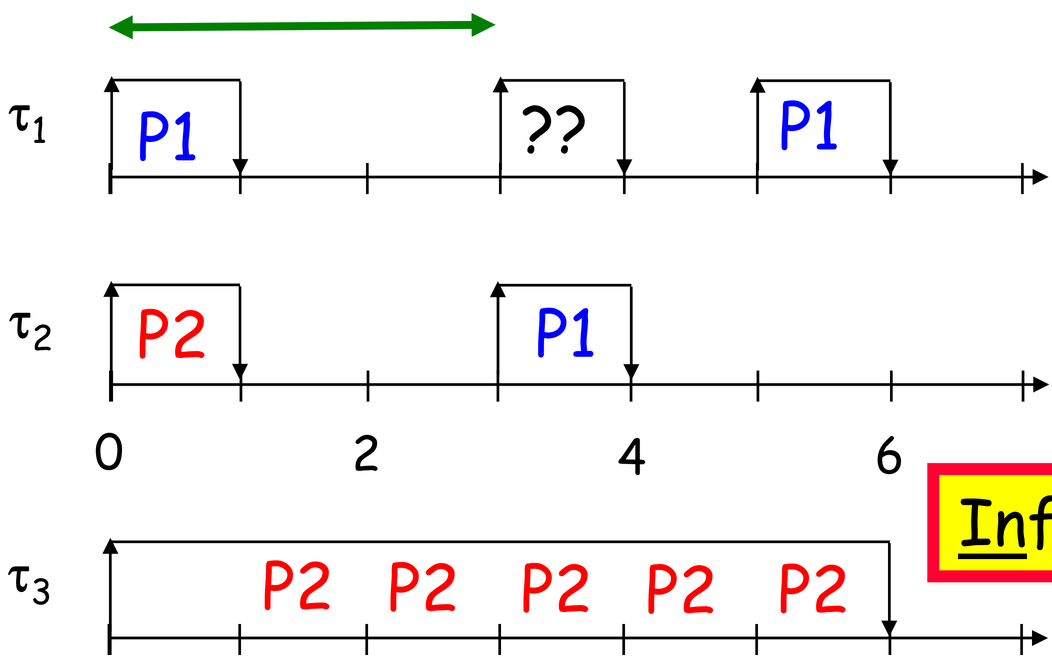
Synchronous Arrival Sequence

Layer 1: Techniques and concepts

RESULT: The Synch. Arrival Sequence <u>not</u> worst-case arrival sequence

$\tau_1 = (1,1,2)$, $\tau_2 = (1,1,3)$, $\tau_3 = (5,6,6)$, 2 Procs

$\tau_i = (e_i,\ d_i,\ p_i)$



<u>Infeasible</u> on 2 processors

<u>Synchronous Arrival Sequence</u>

Layer 1: Techniques and concepts

Feasibility analysis of systems of sporadic tasks

On multiprocessors:

1. identify worst-case arrival sequence:

Worst-case arrival sequence[s] remain unknown

2. validate its schedulability

All sporadic task systems can be shown to either be infeasible on m speed-1 processors, or feasible on m speed-(2 – 1/m) processors.

Bonifaci, Marchotti-Spaccamela, and Stiller. A constant-approx. feasibility test for multiprocessor real-time scheduling. (ESA-2008)

The multi-layered perspective

Layer 1. Techniques & concepts

Layer 2. Metrics

Layer 3. Task and machine models

Layer 2: Metrics

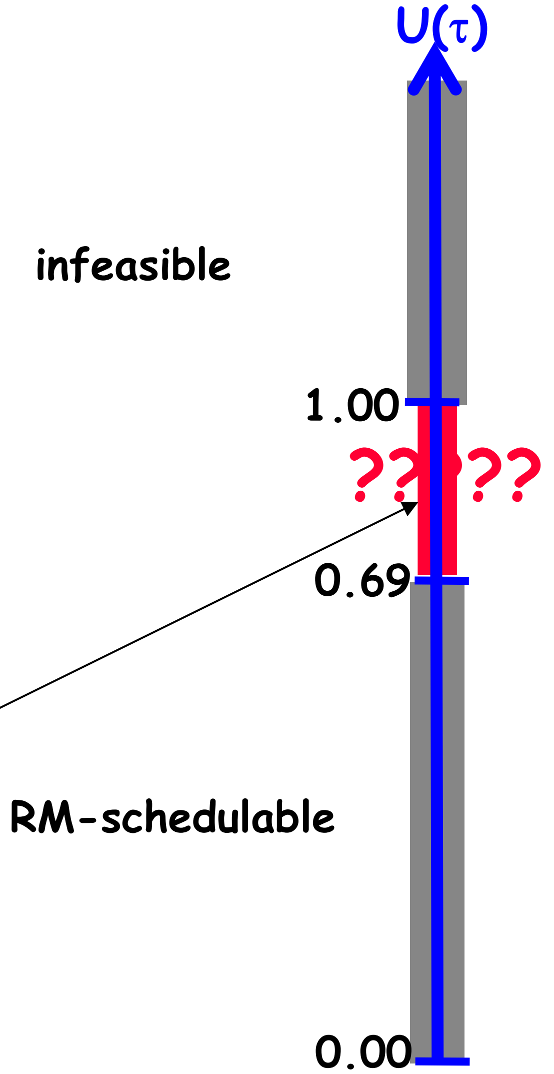
What makes a good metric?

(d_i=p_i for all tasks)

Example (Rate-Monotonic utilization bound). A Liu and Layland task system τ is Rate-Monotonic schedulable on a uniprocessor if $U(\tau) \le 0.69$

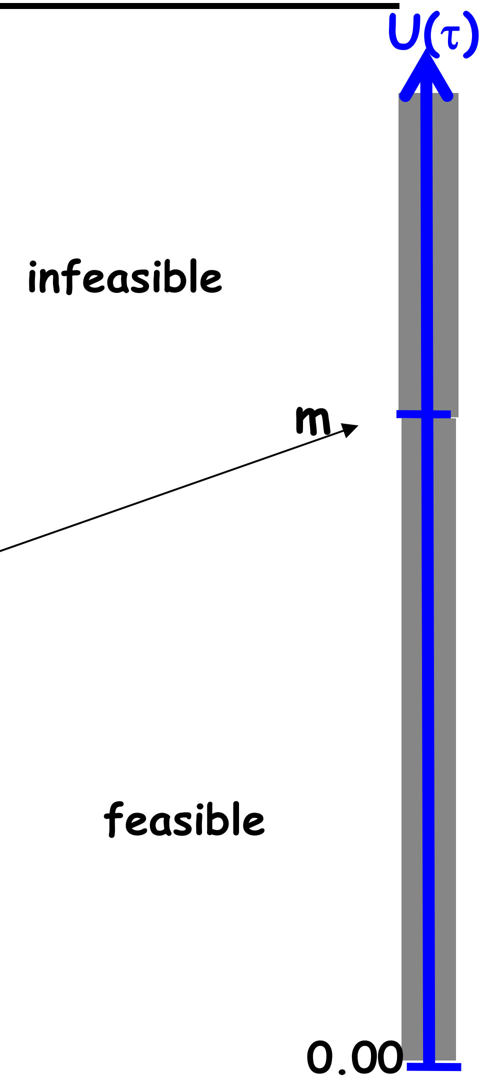
utilization: $e_1/p_1 + e_2/p_2 + \ldots + e_n/p_n$

A good metric minimizes the region of uncertainty



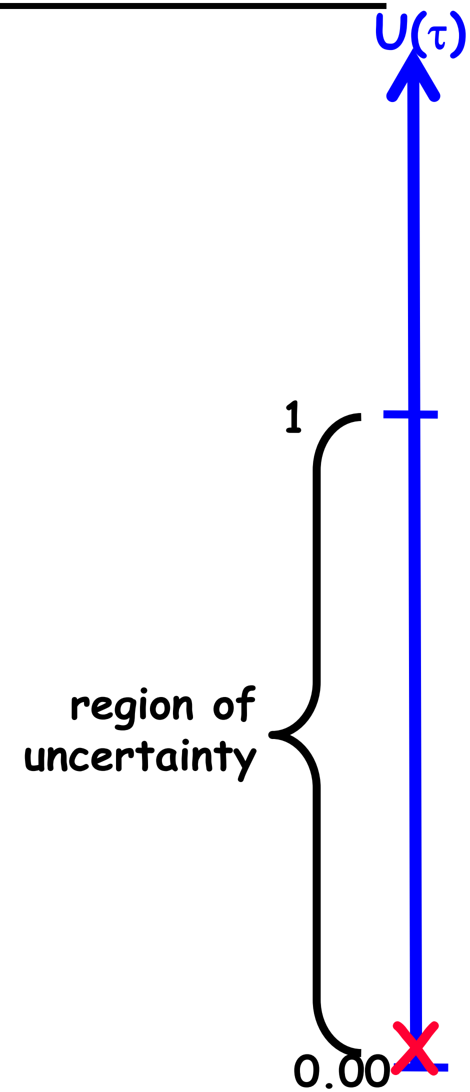
Layer 2: Metrics

Result. A Liu and Layland task system τ with $U(\tau) \leq m$ is feasible on m unit-capacity processors.



No region of uncertainty: utilization is a good metric for multiprocessor feasibility of Liu and Layland task systems

Layer 2: Metrics



Example: $\{\tau_1=(1, 1, p), \tau_2 = (\varepsilon, 1, p)\}$

has utilization = $(1 + \varepsilon)/p$

- (very small for large p)

but infeasible on a preemptive uniprocessor

But, utilization is a poor metric for sporadic task systems (even on uniprocessors)

Layer 2: Metrics

density: $e_1/d_1 + e_2/d_2 + \ldots + e_n/d_n$

Example: $\{\tau_1=(1, 1, p), \tau_2 = (\varepsilon, 1, p)\}$

density = $1/1 + \varepsilon/1 = (1+ \varepsilon) > 1$

has utilization = $(1 + \varepsilon)/p$

- (very small for large p)

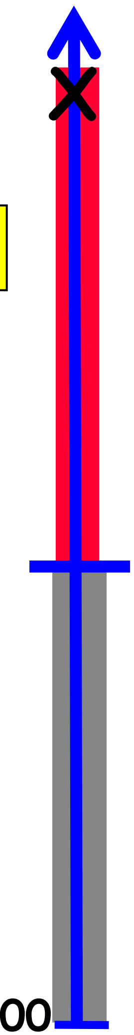
but infeasible on a preemptive uniprocessor

But, density is a poor metric for sporadic task systems (even on uniprocessors)

0.00

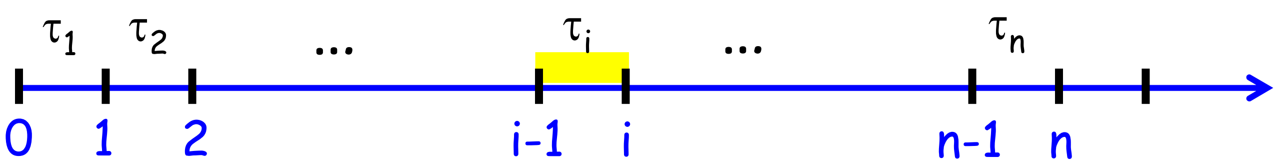
Layer 2: Metrics

density: $e_1/d_1 + e_2/d_2 + \ldots + e_n/d_n$



Example: $\{\tau_1=(1, 1, n), \tau_2 = (1, 2, n), \tau_3 = (1,3,n), \ldots \tau_i=(1,i,n),\ldots, \tau_n=(1,n,n)\}$

- has density = $1/1 + 1/2 + 1/3 + \ldots + 1/n \approx \log_e n$;
- but is feasible on a preemptive uniprocessor



But, density is a poor metric for sporadic task systems (even on uniprocessors)

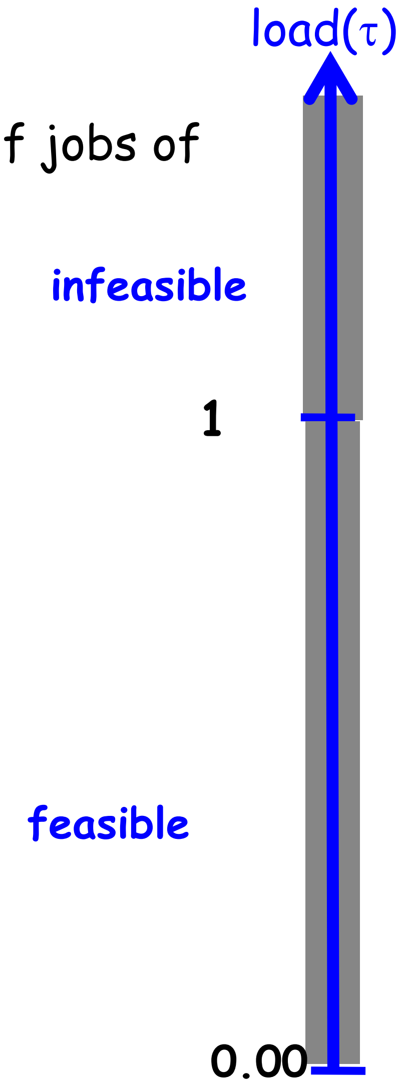
Layer 2: Metrics

DEMAND BOUND FUNCTION

$DBF(\tau_i, t) \equiv$ maximum cumulative execution requirement of jobs of sporadic task τ_i in any interval of length t

$$load(\tau) \equiv \max_{\text{all } t} \left(\sum_{\tau_i \in \tau} DBF(\tau_i, t) \Big/ t \right)$$

Maximum total execution requirement by jobs of sporadic task system τ over any time-interval of length t

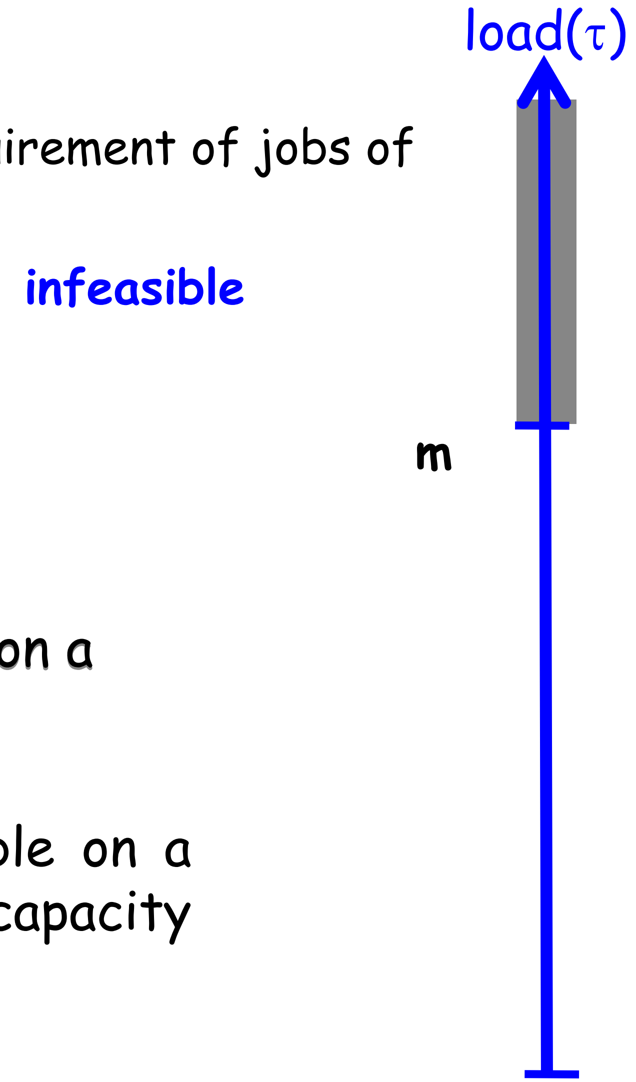


Layer 2: Metrics

DEMAND BOUND FUNCTION

$DBF(\tau_i, t) \equiv$ maximum cumulative execution requirement of jobs of sporadic task τ_i in any interval of length t

$$\text{load}(\tau) \equiv \max_{\text{all } t} \left(\sum_{\tau_i \in \tau} DBF(\tau_i, t) \Big/ t \right)$$



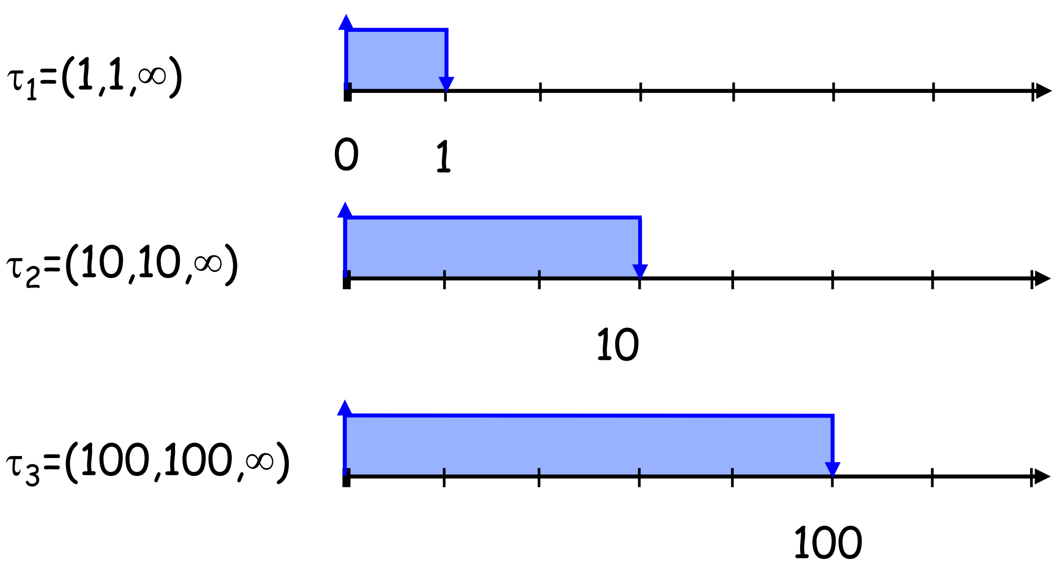
RESULT: Any sporadic task system τ is feasible on a preemptive uniprocessor if and only if $\text{load}(\tau) \leq 1$

RESULT: Any sporadic task system τ is feasible on a preemptive multiprocessor comprised of m unit-capacity procs only if $\text{load}(\tau) \leq m$

Layer 2: Metrics

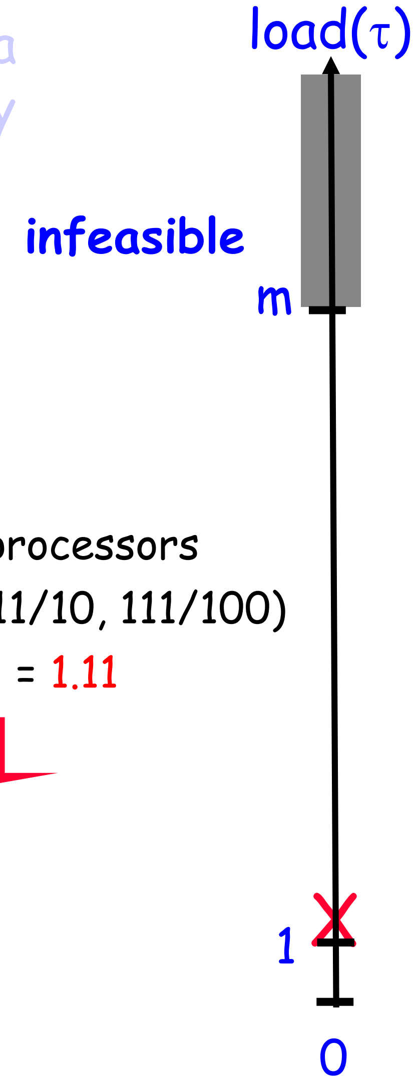
RESULT: Any sporadic task system τ is feasible on a preemptive multiprocessor comprised of m unit-capacity procs only if load(τ) ≤ m is not sufficient for feasibility

Example: $\{\tau_1=(1, 1, \infty), \tau_2 = (10, 10, \infty), \tau_3 = (100, 100, \infty), \}$



(Time-axis not to scale)

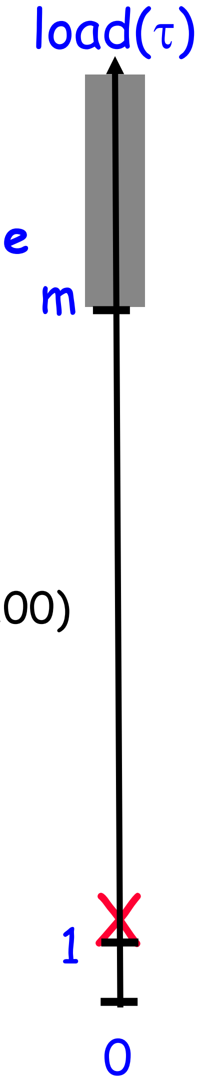
- infeasible on 2 processors
- load = max(1/1, 11/10, 111/100) = 1.11



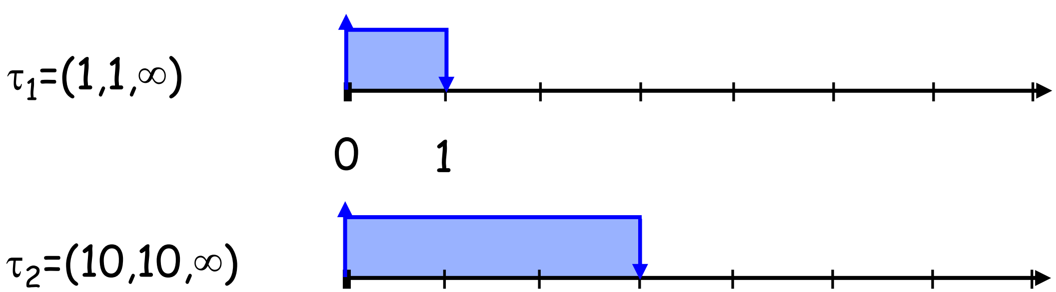
Layer 2: Metrics

RESULT: Any sporadic task system τ is feasible on a preemptive multiprocessor comprised of m unit-capacity procs only if load(τ) ≤ m is not sufficient for feasibility

infeasible



Example: $\{\tau_1=(1, 1, \infty), \tau_2 = (10, 10, \infty), \tau_3 = (100, 100, \infty), \}$



- infeasible on 2 processors
- load = max(1/1, 11/10, 111/100) = 1.11

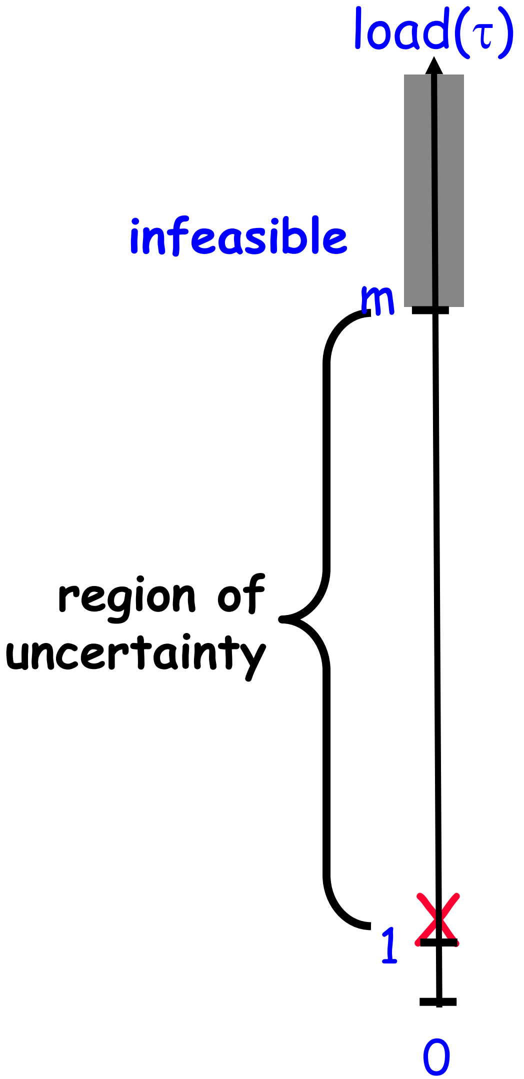
$\tau_3=($

For any number of processors, there are infeasible systems with load = (1+ε), for arbitrarily small ε

100

(Time-axis not to scale)

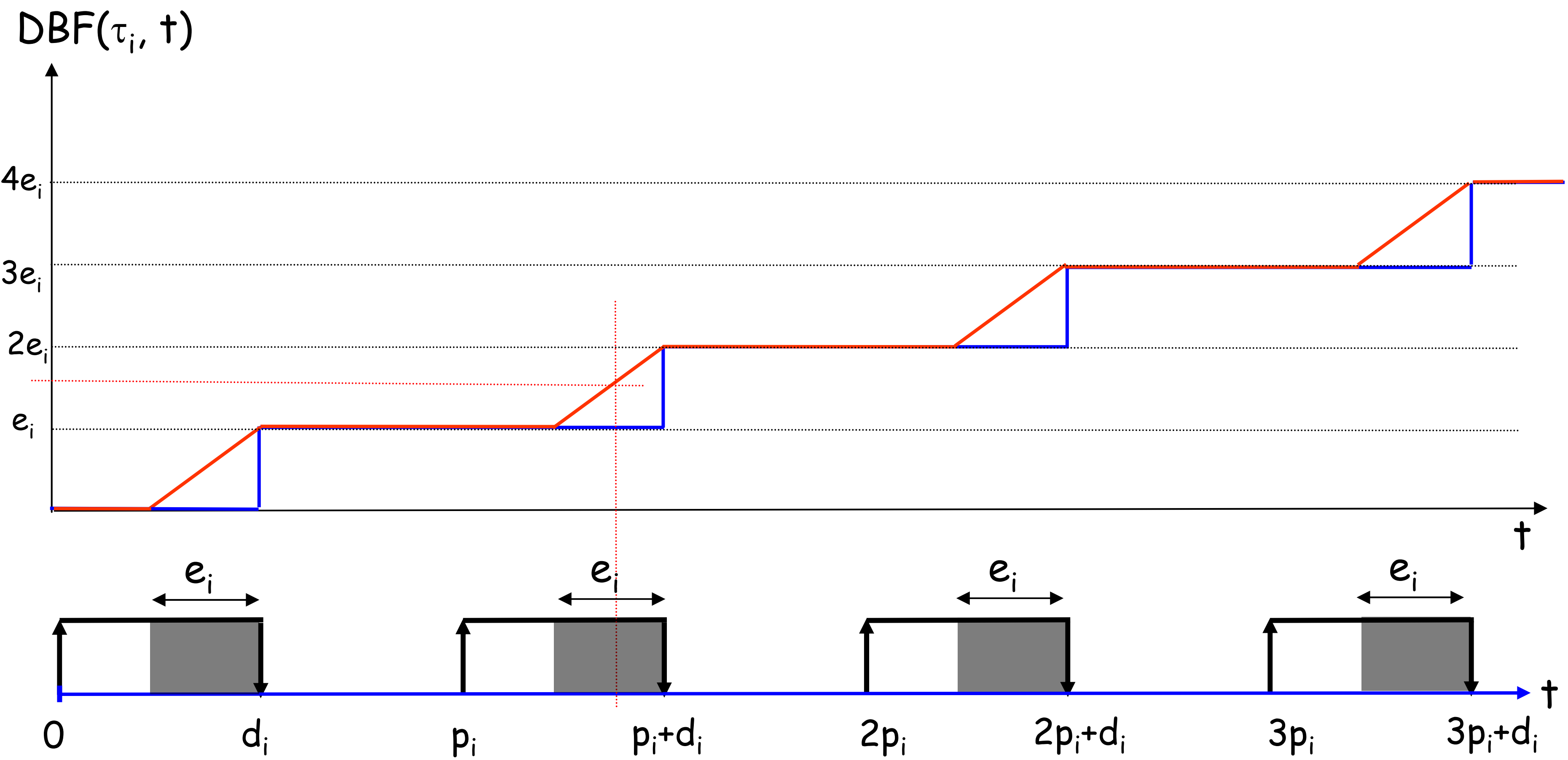
Layer 2: Metrics



Load is a poor metric for sporadic task systems on multiprocessors

Layer 2: Metrics - current status

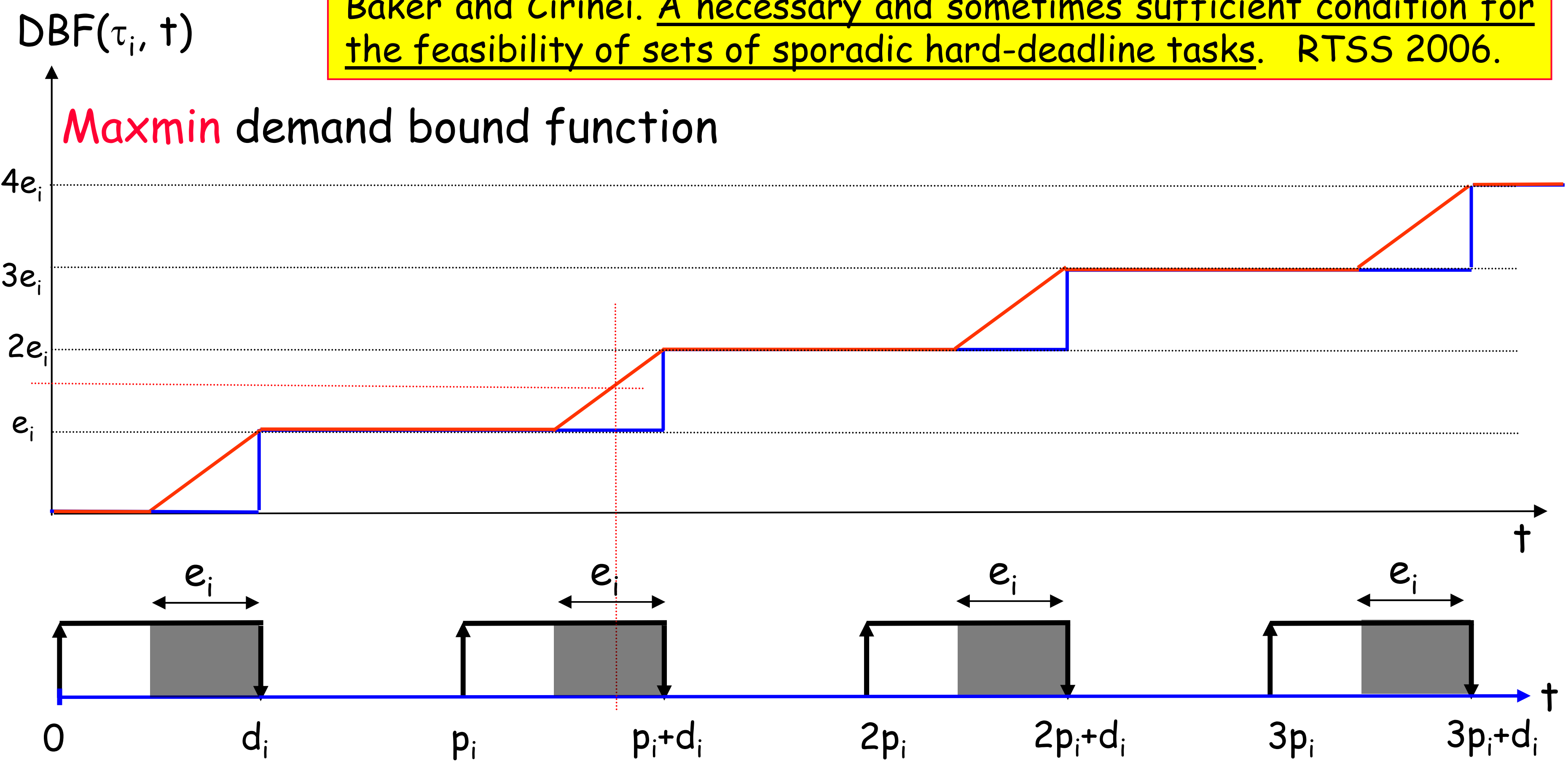
$DBF(\tau_i, t) \equiv$ maximum cumulative execution requirement of jobs of sporadic task τ_i in any interval of length t



Layer 2: Metrics - current status

MAXMIN LOAD

Baker and Cirinei. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. RTSS 2006.



The multi-layered perspective

Layer 1. Techniques & concepts

Layer 2. Metrics

Layer 3. Task and machine models

1. the "no-parallelism" assumption
2. multiple-instance workloads
3. cache considerations

Layer 3: Task and machine models

Assumption: No job-level parallelism

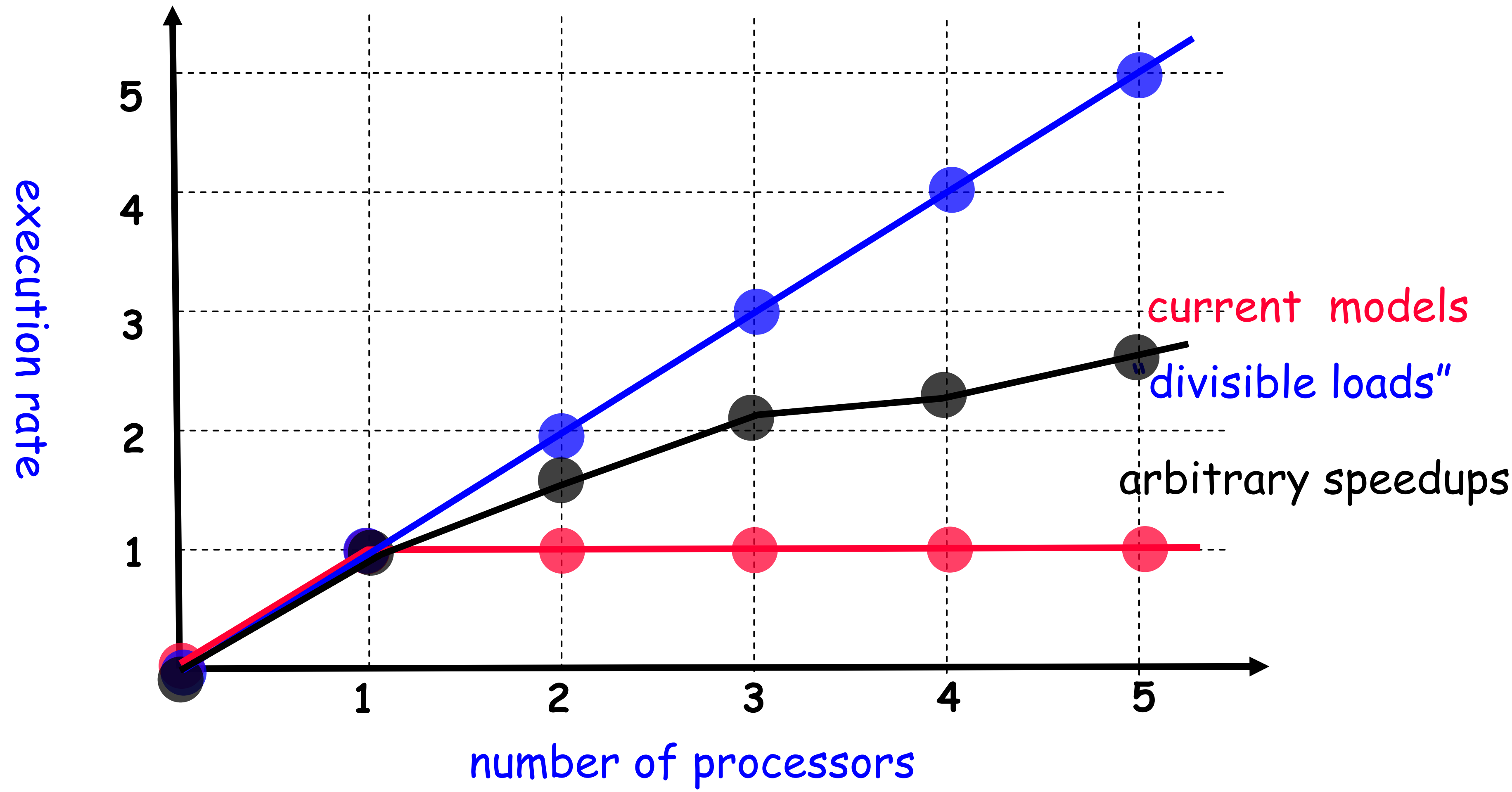
Contributes to difficulty of multiprocessor analysis:

> "The simple fact that a [job] can use only one processor even when several processors are free at the same time adds a surprising amount of difficulty to the scheduling of multiple processors."
>
> \- Liu (1969)

May not be valid any longer: extend the job model

Layer 3: Task and machine models

Incorporating job-level parallelism



Layer 3: Task and machine models

Incorporating job-level parallelism: current status

Steve Goddard and colleagues – University of Nebraska

* Anwar Mamat, Ying Lu, Jitender Deogun and Steve Goddard. Real-time divisible load scheduling with advance reservations. ECRTS 2008

Joel Goossens and colleagues – Université Libre de Bruxelles

* S. Collette and L. Cucu and J. Goossens. Integrating job parallelism in real-time scheduling theory. IPL 2008.

Layer 3: Task and machine models

1. the "no-parallelism" assumption

2. all tasks are distinct

3. cache considerations:

   - the hierarchical arrangement of processors

Summary

Multiprocessor systems are increasingly important

$\Longrightarrow$ need a theory of multiprocessor RT scheduling

Uniprocessors to Multiprocessors:

an evolutionary change? or a paradigm shift?

Extend uniprocessor scheduling theory to multiprocessors?

Yes (for an approximate theory)

- currently sufficient for practical purposes

Long term, probably not

- conjecture: fundamental new theory is needed