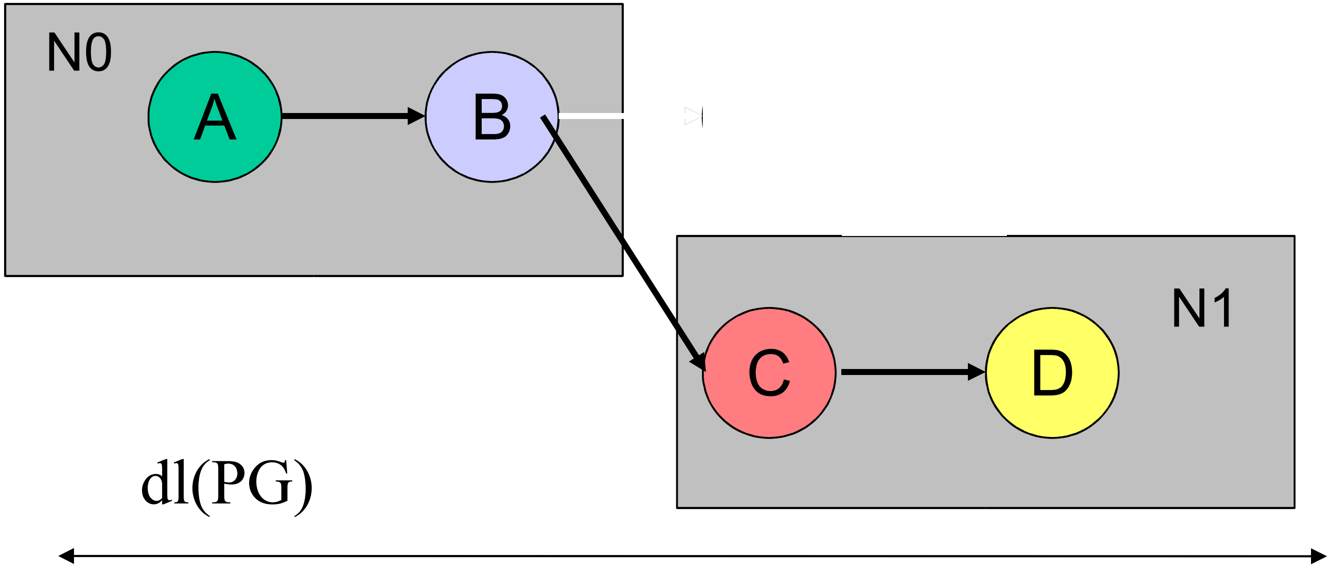
Pre Run-time Scheduling – Flexibility Integration Offline - Online

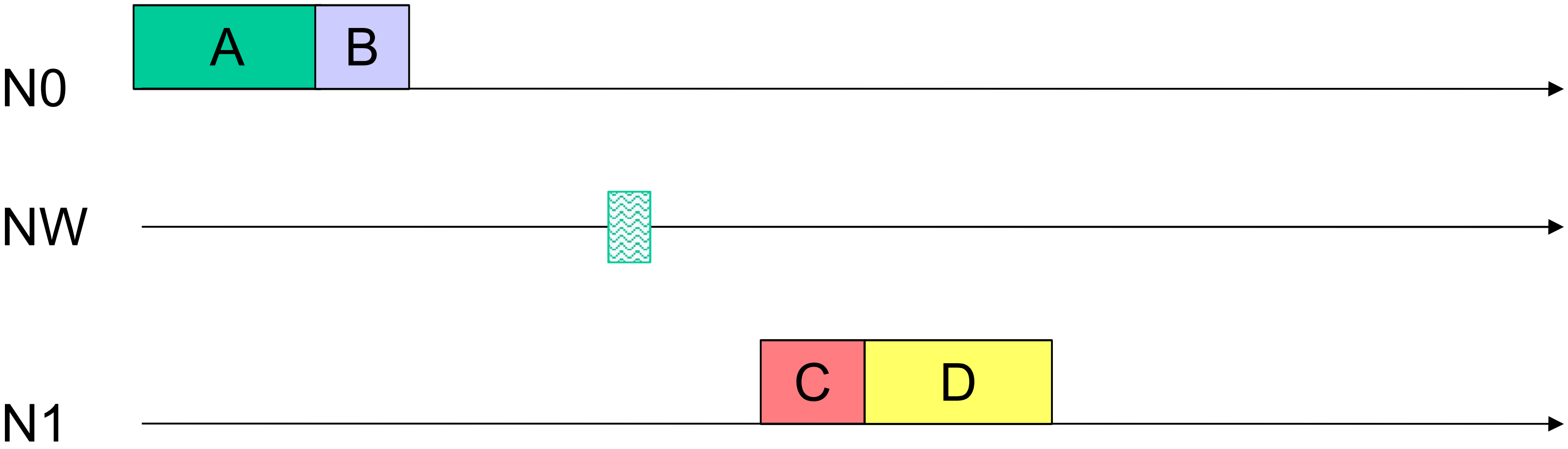
Gerhard Fohler
Mälardalen University, Sweden
gerhard.fohler@mdh.se

Offline schedules

- general timing constraints
- offline scheduler
  - resolves constraints
  - constructs one solution which meets all constraints
- fixed (blind) runtime execution
- no flexibility

- how can we
  - increase flexibility
  - add dynamic tasks
  - integrate with online scheduling methods


N0
A
B
N1
C
D
dl(PG)


A
B
N0
NW
C
D
N1

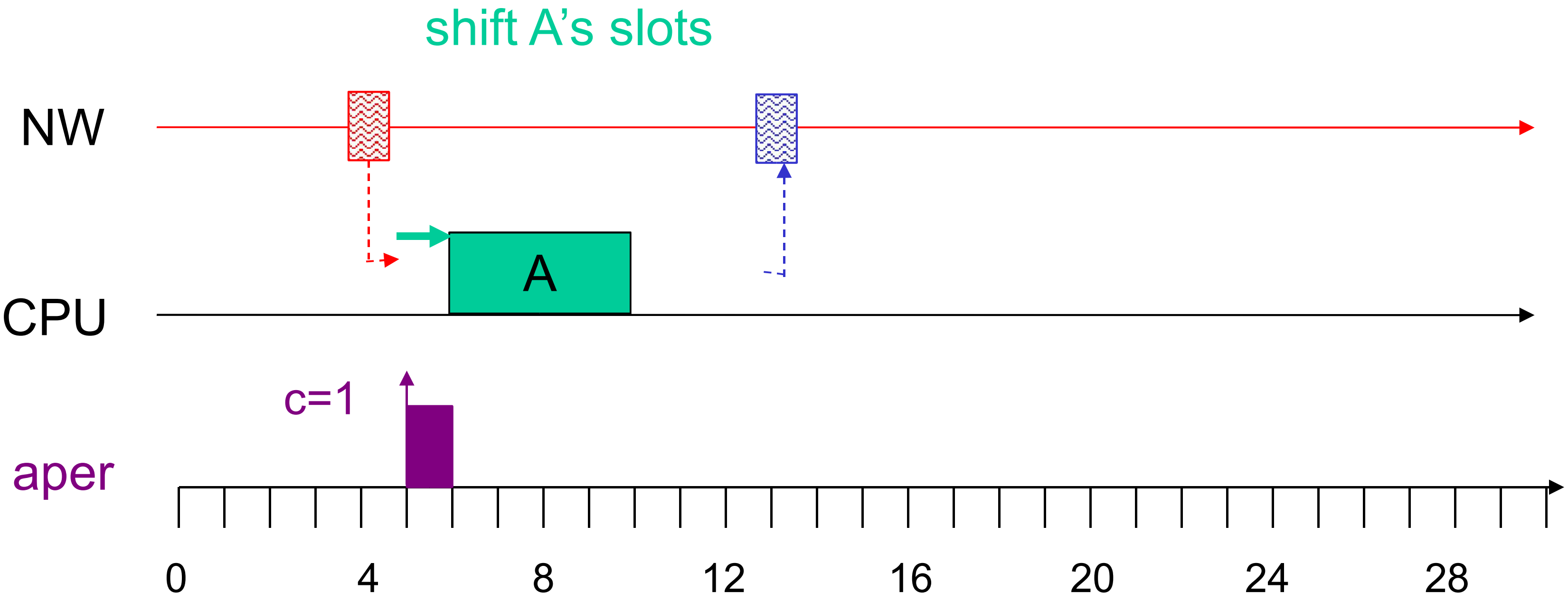
Slot Shifting…

Offline

- timing constraints ✓
- offline schedule ✓

- we have
  - offline constructed schedule
- we want
  - include dynamic tasks
  - schedule them online
- what can we do?
  - include in offline schedule (e.g., pseudo periodic)
    inefficient
  - fit into empty slots   no guarantees
  - we can do better!

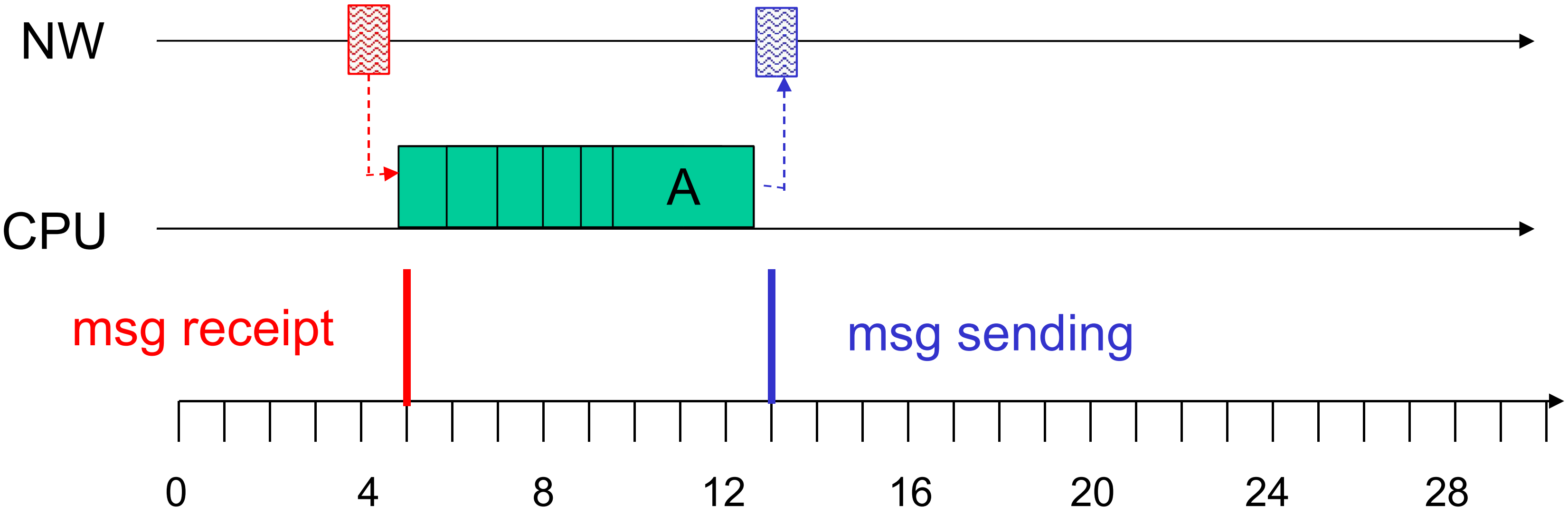
Basic Idea



Shifting pre-runtime tasks

- pre runtime schedule assigns fixed times for execution
  - although different times possible
  - *overconstrains* schedule
    - we have to select *one out of several possible* times
    - …for the *sake of algorithm* only
- we know, that we can shift A
  - execute the aperiodic task at once
  - feasibility of tasks not violated
  - how much and where can we shift?
  - what are boundaries?

Shifting tasks



Limitations on Shifting

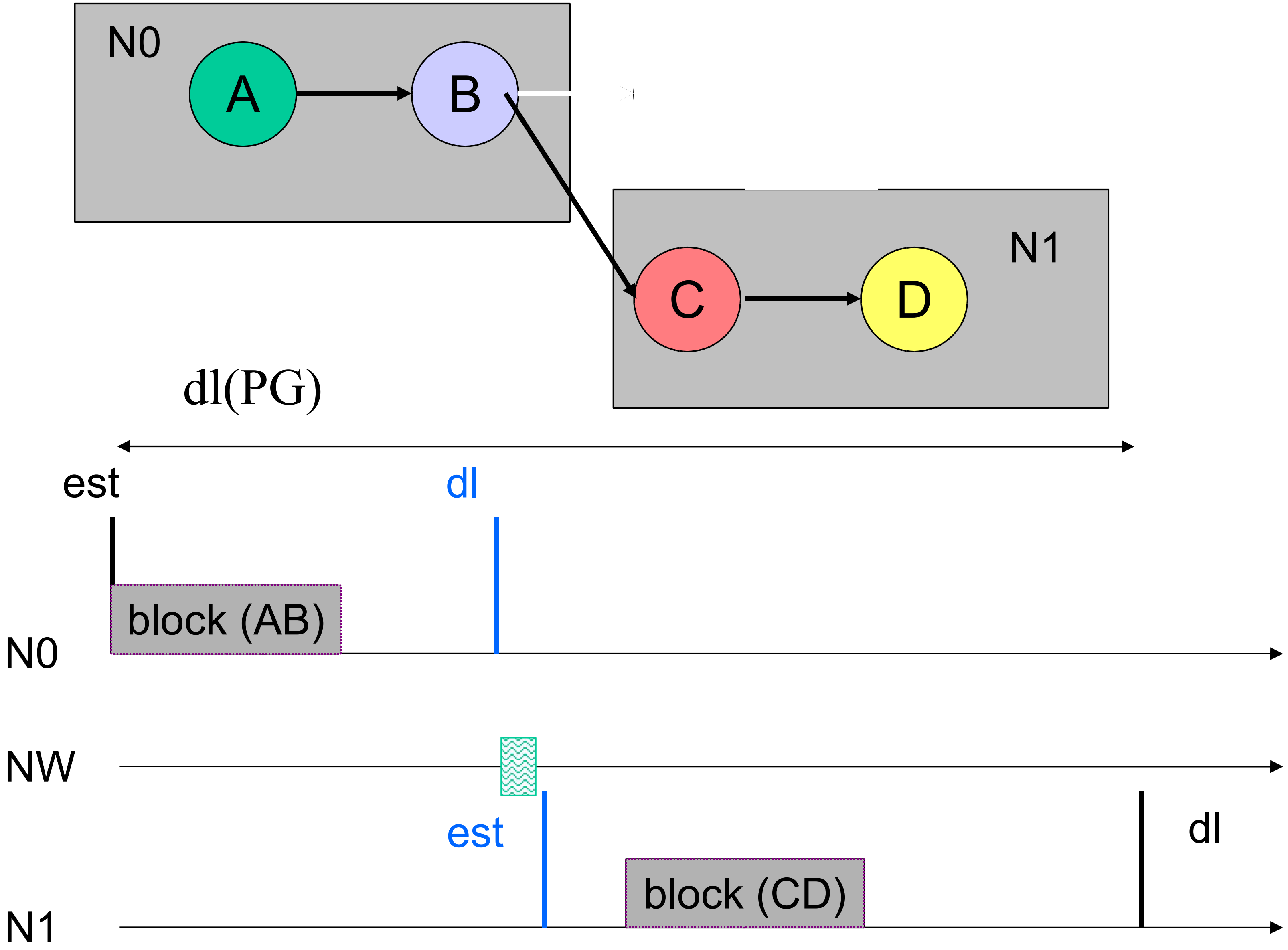
we can shift tasks

limitations

- receipt of message
- sending of message
- earliest start time of precedence graph, end-to-end constraints, task chain
- deadline of -”-

calculate start time, deadline pairs for tasks

- expresses flexibility of task
- reduces overconstraining
- fit in aperiodic task by shifting as long as these constraints met


N0
A
B
N1
C
D
dl(PG)
est
dl
block (AB)
N0
NW
est
dl
block (CD)
N1

These tasks are assigned fixed *starttimes* or *deadlines.*

(Subgraphs of precedence graphs allocated to nodes combined.)

independent tasks with starttimes, deadlines on single nodes

simple EDF runtime scheduling

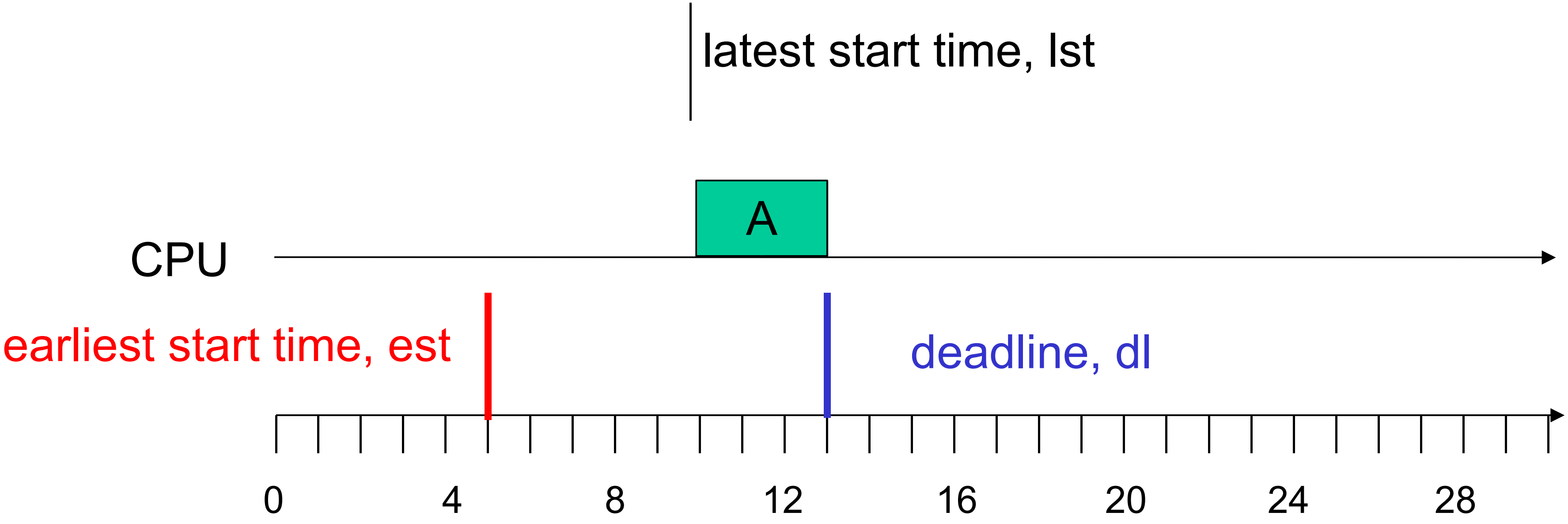
Slot Shifting …

Offline

- timing constraints
- offline schedule
- earliest start times, deadlines ✓

How much shifting?

- know what is
  - earliest time to start task
  - latest time to finish
- aperiodic arrives: how far can we shift static task?



- latest start time
  start no later or violate deadline
- have to ensure when executing aperiodics
  how?
- more complex dispatching
  still next task, but check for constraints
- more memory - 3 integers per task

Slot Shifting

Offline

- timing constraints
- offline schedule
- earliest start times, deadlines
- latest start times ✓

Insert how much? Where?

so far soft aperiodics

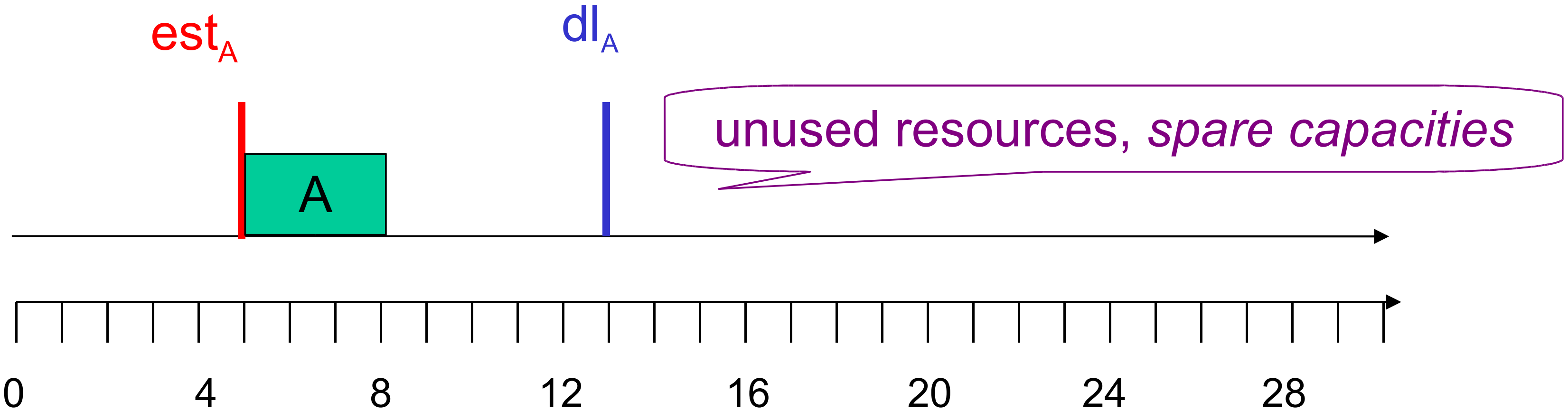
can we give guarantees for firm aperiodics?

- worst case execution time
- deadline
- before start, want to guarantee that we can complete them

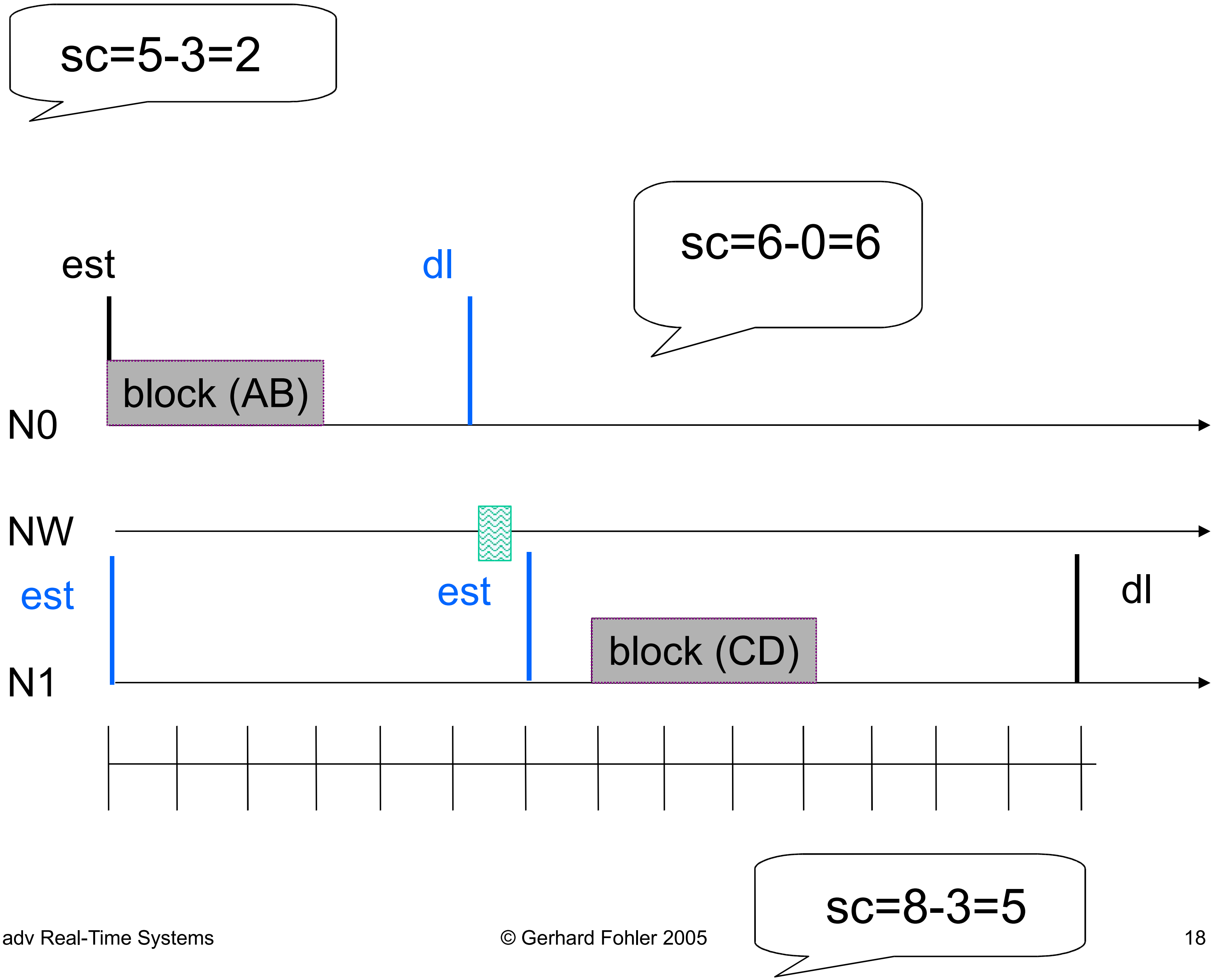
how can we decide?

- need idle resources for aperiodics
- before deadline of aperiodic
- which resources can we use?

Spare Capacities



- spare capacities, sc = length of execution interval
  - execution times
- available for aperiodic tasks
- know amount and location from schedule!


sc=5-3=2
est
dl
sc=6-0=6
block (AB)
N0
NW
est
est
dl
block (CD)
N1
sc=8-3=5

Slot Shifting

Offline

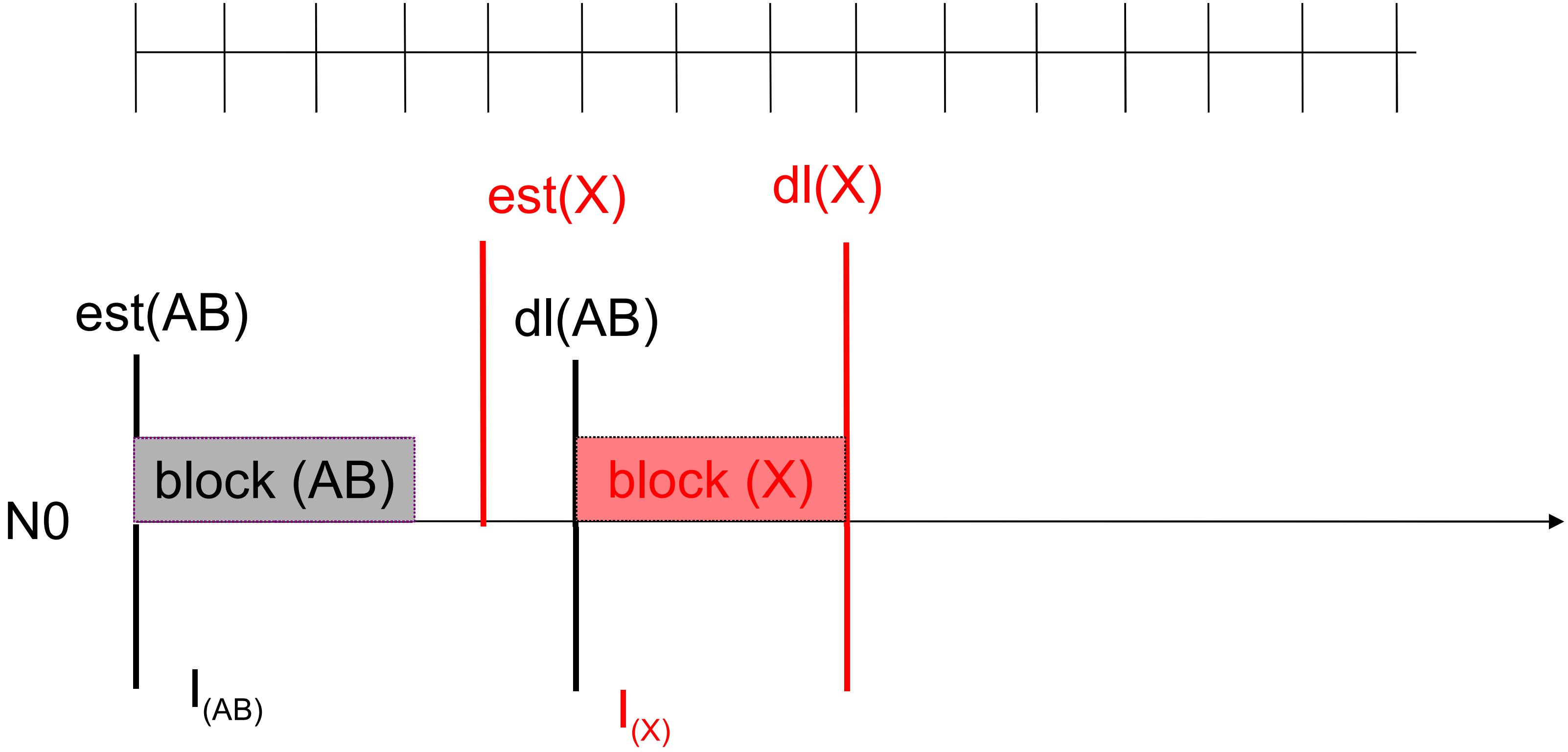
- timing constraints
- offline schedule
- earliest start times, deadlines
- latest start times
- ~~spare capacities~~ ✓ not yet…

Intervals

sort deadlines *disjoint invervals:*

- *end*: deadline of task(s)
- tasks with that deadline
- *spare capacity, sc*
  the amount of idle resources in that *interval*
- *start*: max of est of task(s) and end previous interval

- empty intervals:
  - $\mathrm{end}(I_{i-1}) < \mathrm{start}(I_i)$
  - wcet = 0



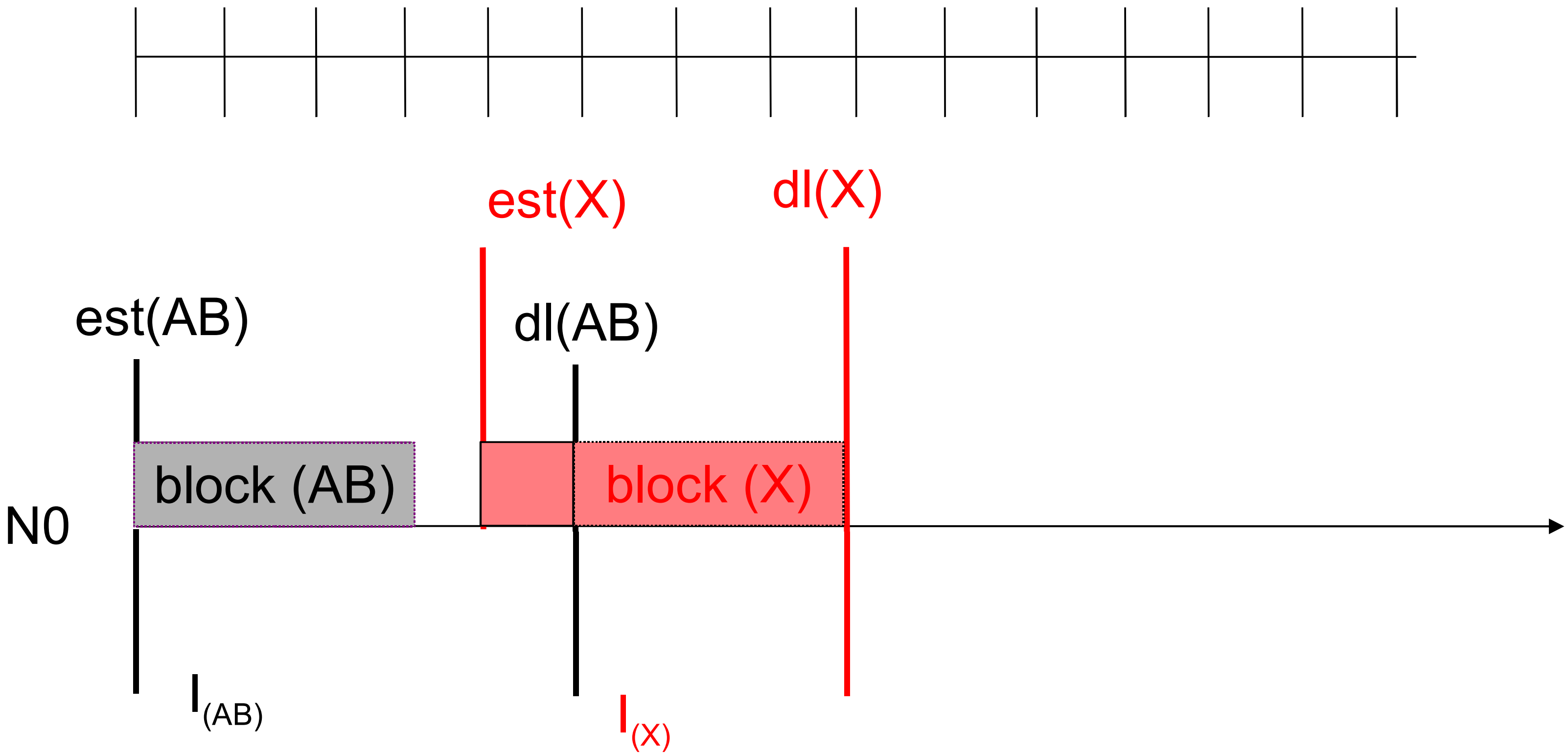
$sc(I_{(AB)})=5-3=2$

$sc(I_{(X)})=3-3=0$

$$\boxed{sc(I) \quad |I| \qquad \underset{T \quad I}{} \quad wcet(T)}$$

..almost the truth…

- intervals   execution intervals!



~~$sc(I_{(AB)})=5-3=2$~~

$sc(I_{(X)})=3-4=-1$

$sc(I_{(AB)})=5-3\boxed{-1}=1$

“borrowing”

$$\boxed{sc(I_i) \quad |I_i| \qquad \underset{T \quad I_i}{} \; wcet(T) \quad \min(sc(I_{i \quad 1}),0)}$$

borrowing mechanism:

- if tasks in subsequent interval need more resources than available in it:
  execute in other interval, use resources from there “borrow”
- run-time mechanisms resolve negative spare capacity
- only for calculation and flexibility
- start of interval can be   earliest start time
- earliest start time checked separately

Slot Shifting

Offline

- timing constraints
- offline schedule
- earliest start times, deadlines
- latest start times
- intervals ✓
- spare capacities ✓

Online Mechanisms- Scheduling

online scheduler invoked at each node after each slot

- check for new aperiodic tasks
- guarantee algorithm
- take scheduling decision
- update spare capacities
- execute scheduling decision

earliest deadline first

- after each slot, scheduling decision taken locally at each node
  - no ready task:
    CPU idle
  - $sc(I_c) > 0$, soft aperiodic task A:
    execute A
  - $sc(I_c) = 0$:
    an offline or guaranteed task has to be executed or deadlines are missed
    takes care that no latest start time is missed!
    no other mechanism needed, eg, watchdog, etc
    implicit invocation, no extra memory needed
  - $sc(I_c) > 0$, soft aperiodic task:
    offline or guaranteed task executed

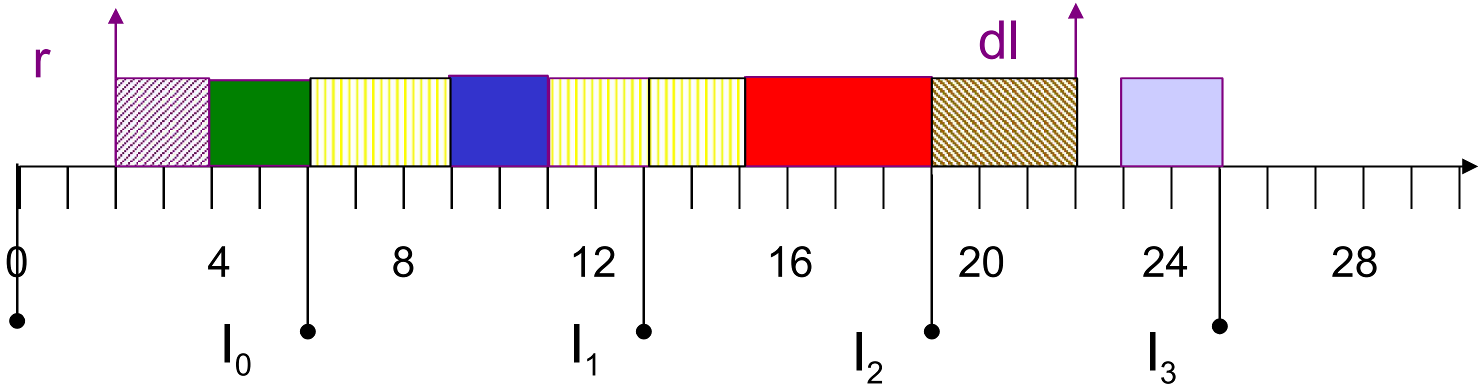
Acceptance of aperiodics

- aperiodics (without deadline):
  sc > 0: one slot can be given to it
- firm aperiodics (wcet and deadline):
  want them executed either completely or not at all

guarantee algorithm

O(N)

- aperiodic task A (r,wcet, dl)
- three parts of spare capacities available



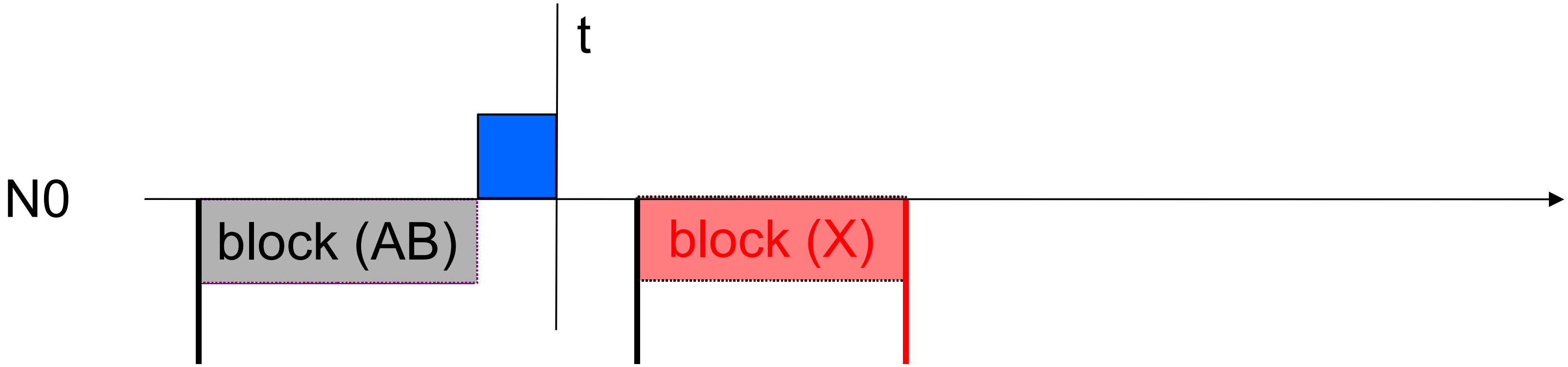
- ▨ $sc(I_c)$: remaining sc in current interval
- ▥ $sc(I_i)$: $sc(I_i)>0$, $c < i$ l, $end(I_l)$ $dl(A)$, $end(I_{i+1}) > dl(A)$, sc in all *full* intervals between r and dl
- ▨ $\min(sc(I_{l+1}), dl(A) - dl(I))$, minimum spare capacities of last interval or up to the deadline of aperiodic in last interval
- possibly interval split

Guarantee

- if sum of total sc between dl and r are larger or equal wcet, guarantee
- need to ensure guarantees resources are not used otherwise
- after guarantee:
  - update interval l
  - update interval l-1
  - …
  - update interval c

Spare capacities at runtime

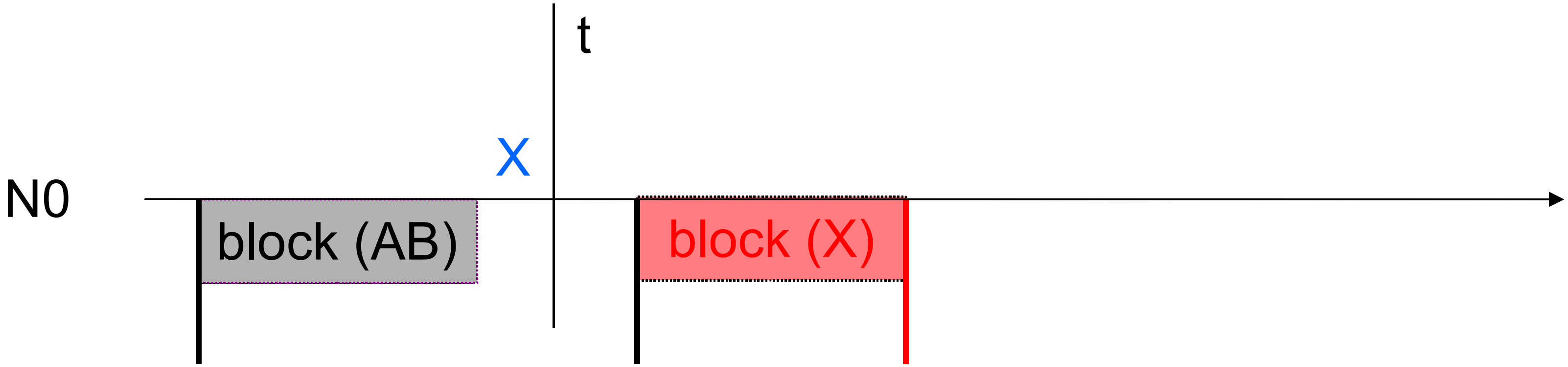
- aperiodic execution
  - decrease spare capacity of current interval



$I_{(AB),}\ sc(I_{(AB)})=2$ $I_{(X),}\ sc(I_{(X)})=0$

at t: $sc(I_{(AB)})=2-1=1$ $I_{(X),}\ sc(I_{(X)})=0$

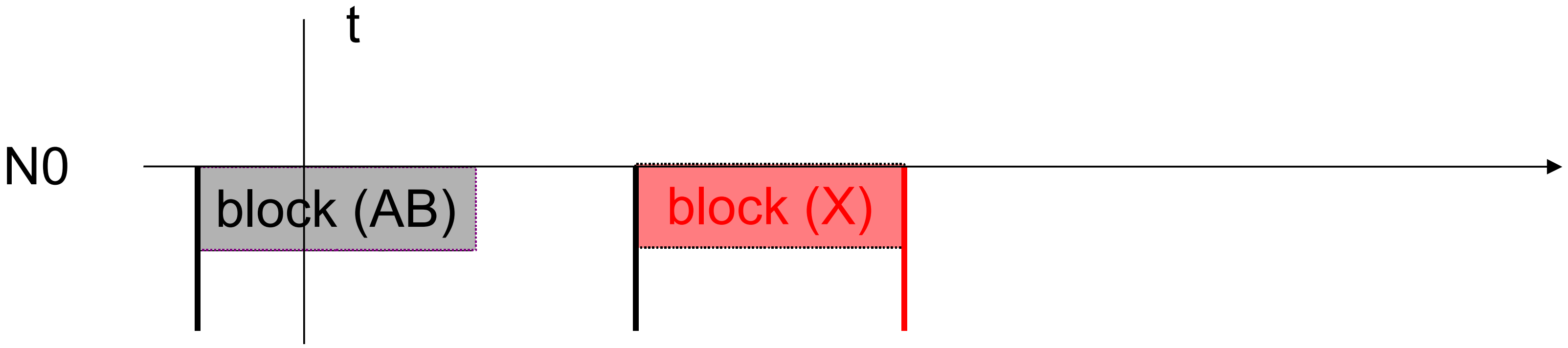
- no execution
  - decrease spare capacity of current interval



$I_{(AB),}\ sc(I_{(AB)})=2$ $I_{(X),}\ sc(I_{(X)})=0$

at t: $sc(I_{(AB)})=2-1=1$ $I_{(X),}\ sc(I_{(X)})=0$

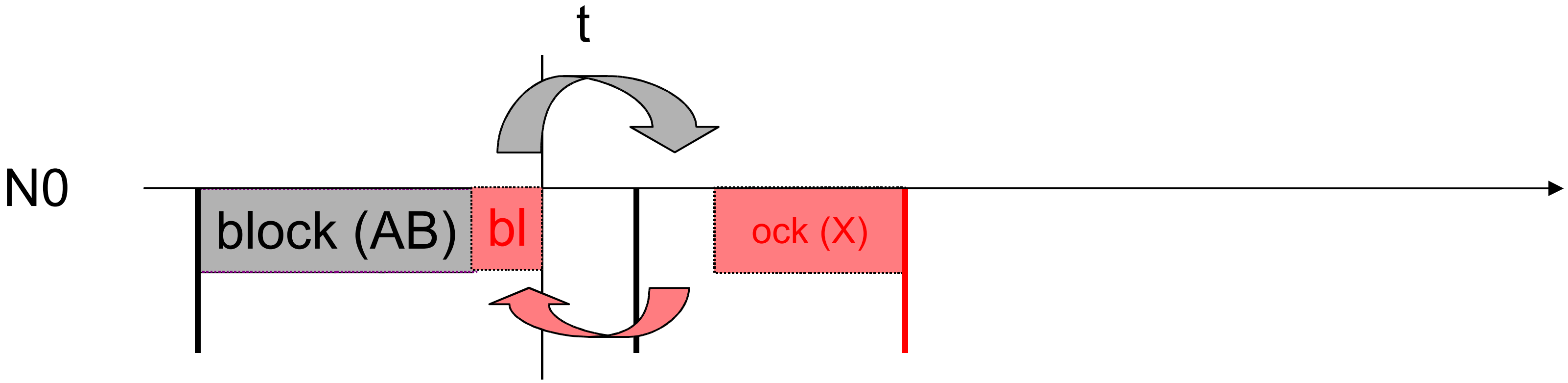
- execution of offline task T
  - T current interval I_c
    spare capacity stays the same



$I_{(AB),}$ $sc(I_{(AB)})=2$ $I_{(X),}$ $sc(I_{(X)})=0$

at t: $sc(I_{(AB)})=2$ $I_{(X),}$ $sc(I_{(X)})=0$

- execution of offline task T
  - T future interval I_f
    - spare capacity I_c decreased
    - spare capacity I_f increased



$I_{(AB),}$ $sc(I_{(AB)})=2$ $I_{(X),}$ $sc(I_{(X)})=0$

at t: $sc(I_{(AB)})=2-1=1$ $I_{(X),}$ $sc(I_{(X)})=0+1=1$

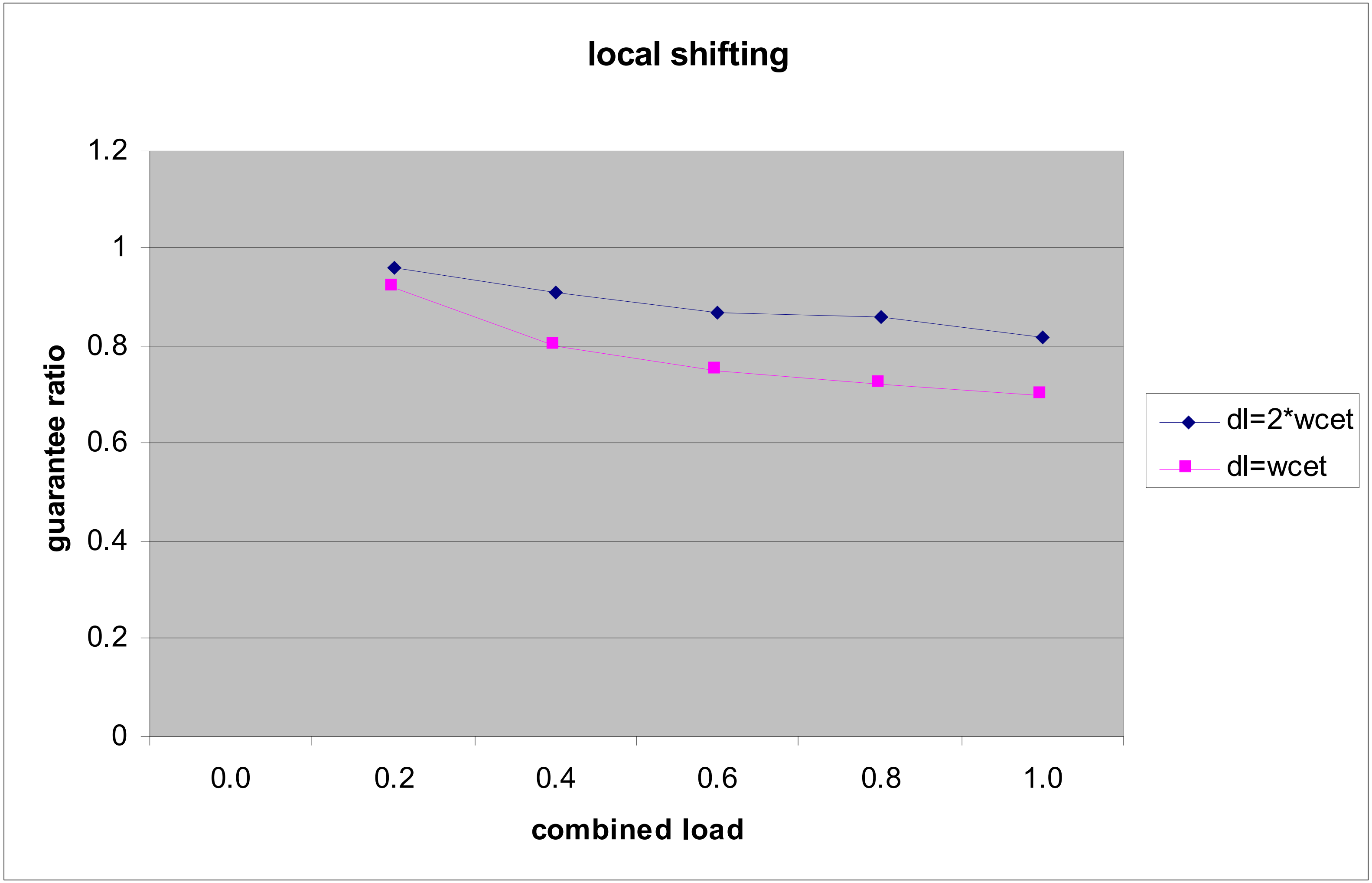
- update capacity of I_f
  - if   0 …done
  - if < 0 … need to update previous interval I_{f-1}
- sc(I_{f-1})
  - if   0 …done
  - if < 0 … need to update previous interval I_{f-2}
- ….
- until sc   0 or I_c

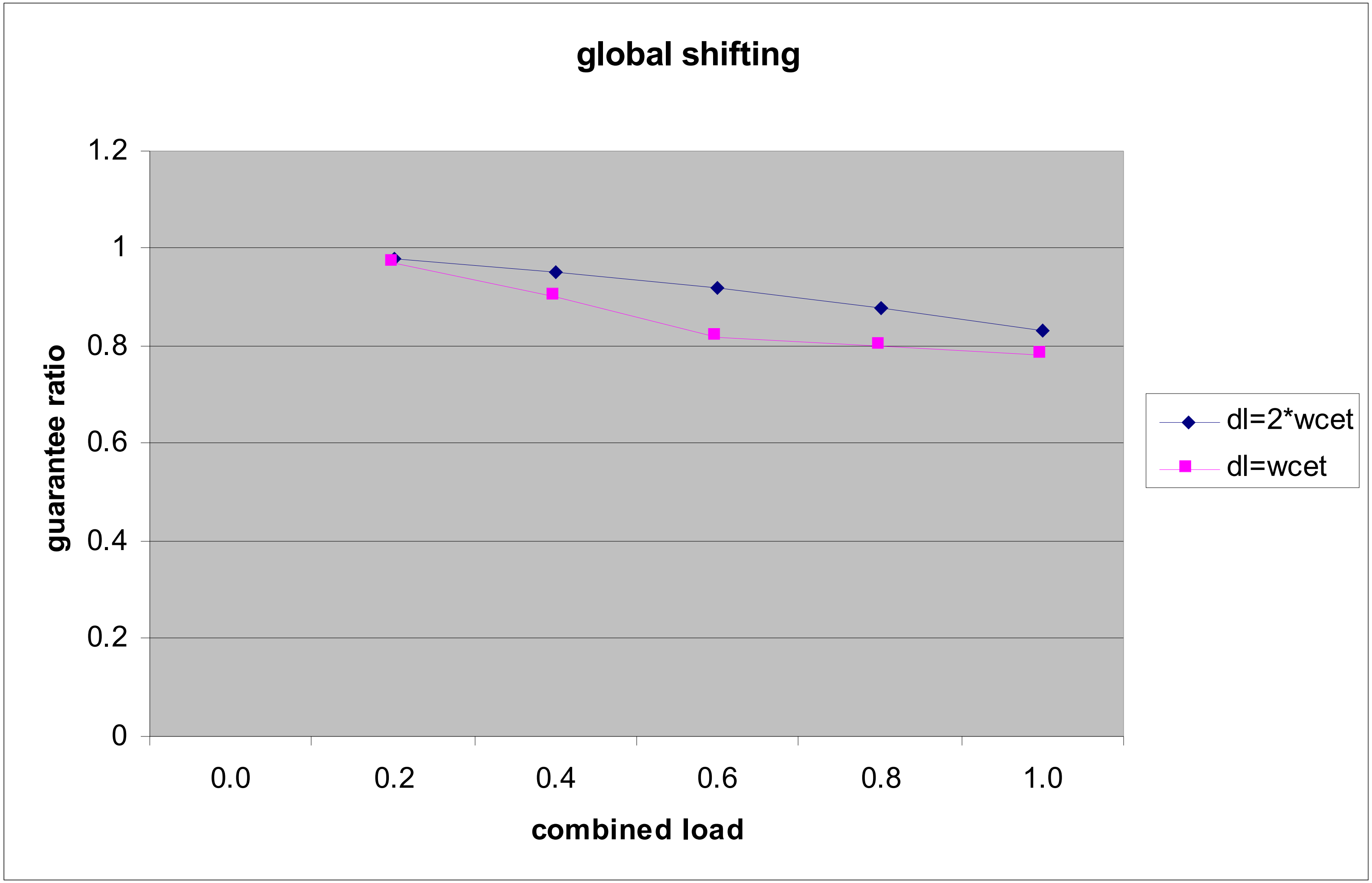
Shifting Messages

- communication medium resource like CPU from scheduling perspective
- shift messages as well
- restriction to sending messages *earlier*
  - no receiver synchronization necessary
  - may increase spare capacities at receiver
  - when message received - spare capacities updated
  - else same

Analysis

- MARS
- 4 CPUs
- TDMA network
- ~1600 task sets generated and pre runtime scheduled
- randomly generated aperiodic tasks
- each point in plots 700-1000 task sets
- 0.95 confidence intervals < 5%


local shifting
guarantee ratio
1.2
1
0.8
0.6
0.4
0.2
0
0.0
0.2
0.4
0.6
0.8
1.0
combined load
◆ dl=2*wcet
■ dl=wcet


global shifting
1.2
1
0.8
0.6
0.4
0.2
0
guarantee ratio
0.0
0.2
0.4
0.6
0.8
1.0
combined load
dl=2*wcet
dl=wcet

“Slot shifting nouveau”

- further acceptance test
- integration with TBS
- ….

		Periodic with constraints		Sporadic	Aperiodic	
		Simple	Complex		Firm	Soft
		• Periods • Deadlines • Start times	• End-to-end dl • Inst. separation • Distribution • Jitter etc.	Minimum separation between instances	• Deadlines • Guarantee	• No dl
Offline	Sch	x	x			
	Test			x		
Online	Sch	x	x	x	x	x
	Test				x	

Slot Shifting - Summary

- handle online tasks while maintaining feasibility of offline scheduled tasks
- offline reduction of complexity
- simple runtime handling
- “interface” for integration of offline and online scheduling

- offline scheduled system for critical activities
- restrict amount of shifting
- flexibility for rest

predictable flexibility

Articles

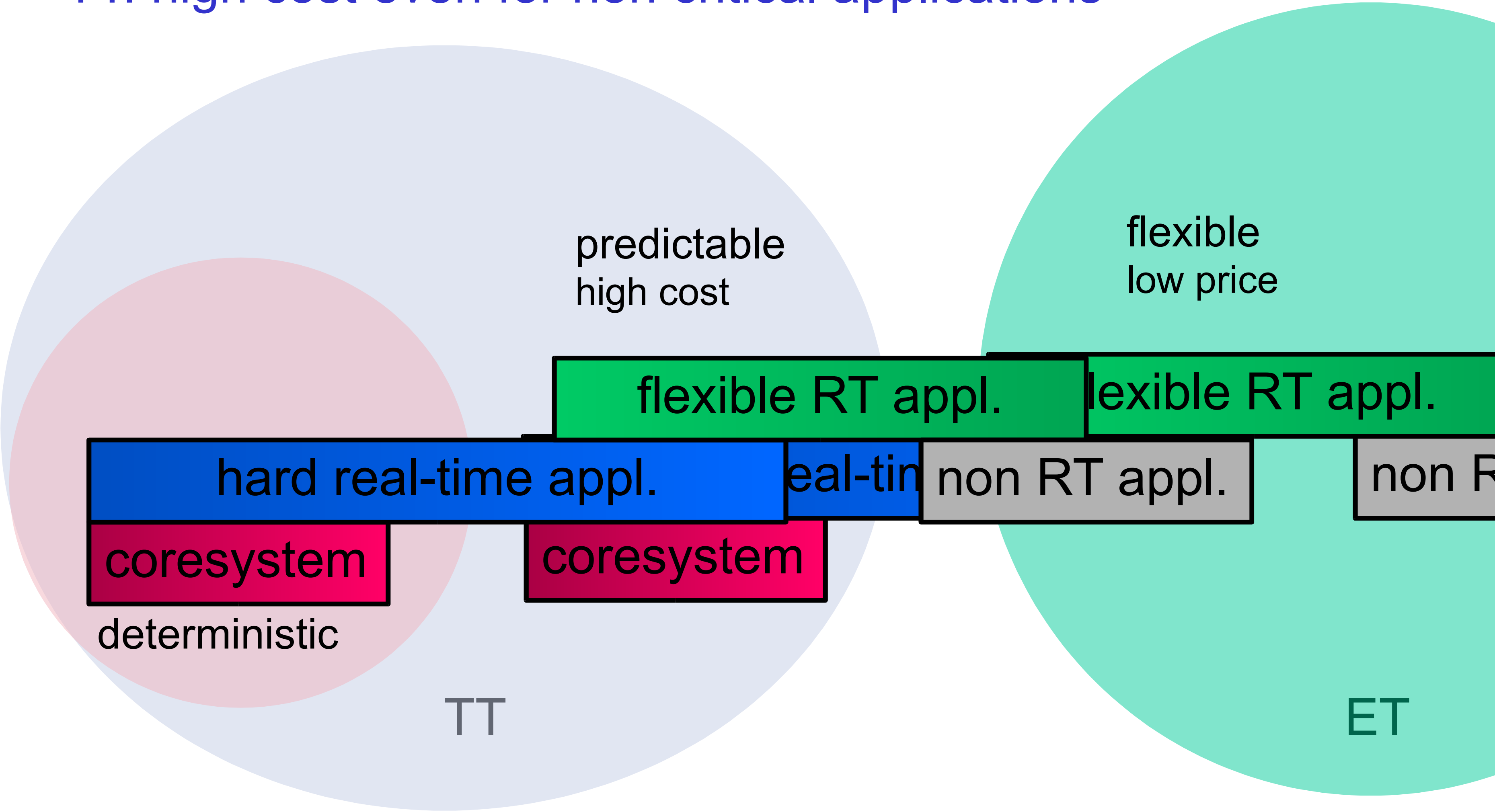
- Gerhard Fohler
  Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems
  Proc. of the 16th IEEE Real-Time Systems Symposium, Pisa, Italy, December 1995.
- Damir Isovic, Gerhard Fohler
  Efficient Scheduling of Sporadic, Aperiodic, and Periodic Tasks with Complex Constraints
  Proc. of the 21st IEEE Real-Time Systems Symposium, Orlando, Florida, USA , November 2000

Novel Applications

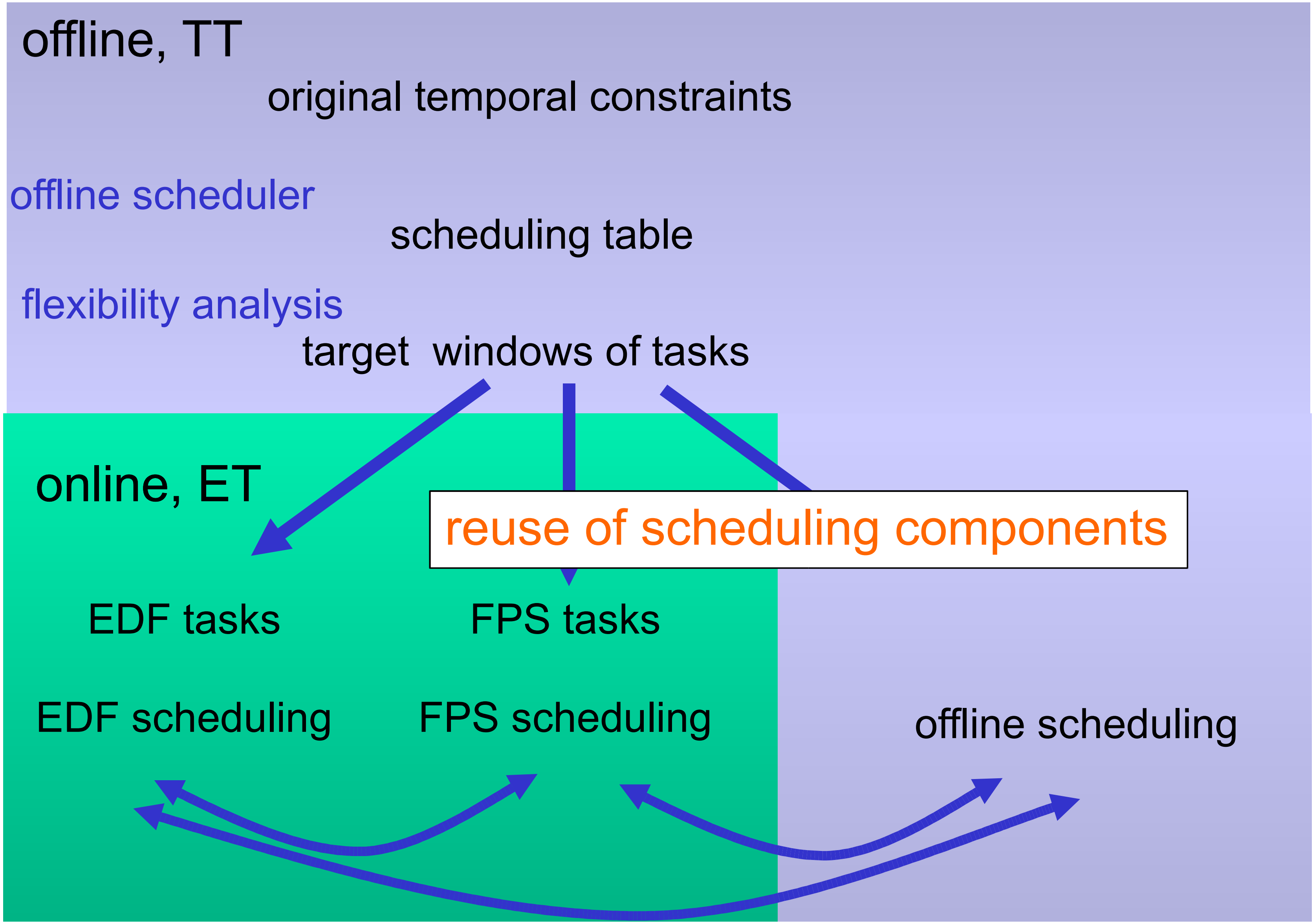
mix of activities and demands

- core system with high demands
  - strict timing behavior
  - safety critical, fault tolerant
  - proven and tested for worst case
- hard real-time applications
  - temporal correctness, etc.
- flexible real-time applications
  - not completely known
  - some deadlines can be missed
- non real-time activities
  - must not disturb real-time activities

TT: high cost even for non critical applications



ET: not deterministic behavior of critical activities



offline, TT

original temporal constraints

offline scheduler

scheduling table

flexibility analysis

target windows of tasks

online, ET

reuse of scheduling components

EDF tasks

FPS tasks

EDF scheduling

FPS scheduling

offline scheduling

Predictable Flexibility

target windows control flexibility of task execution

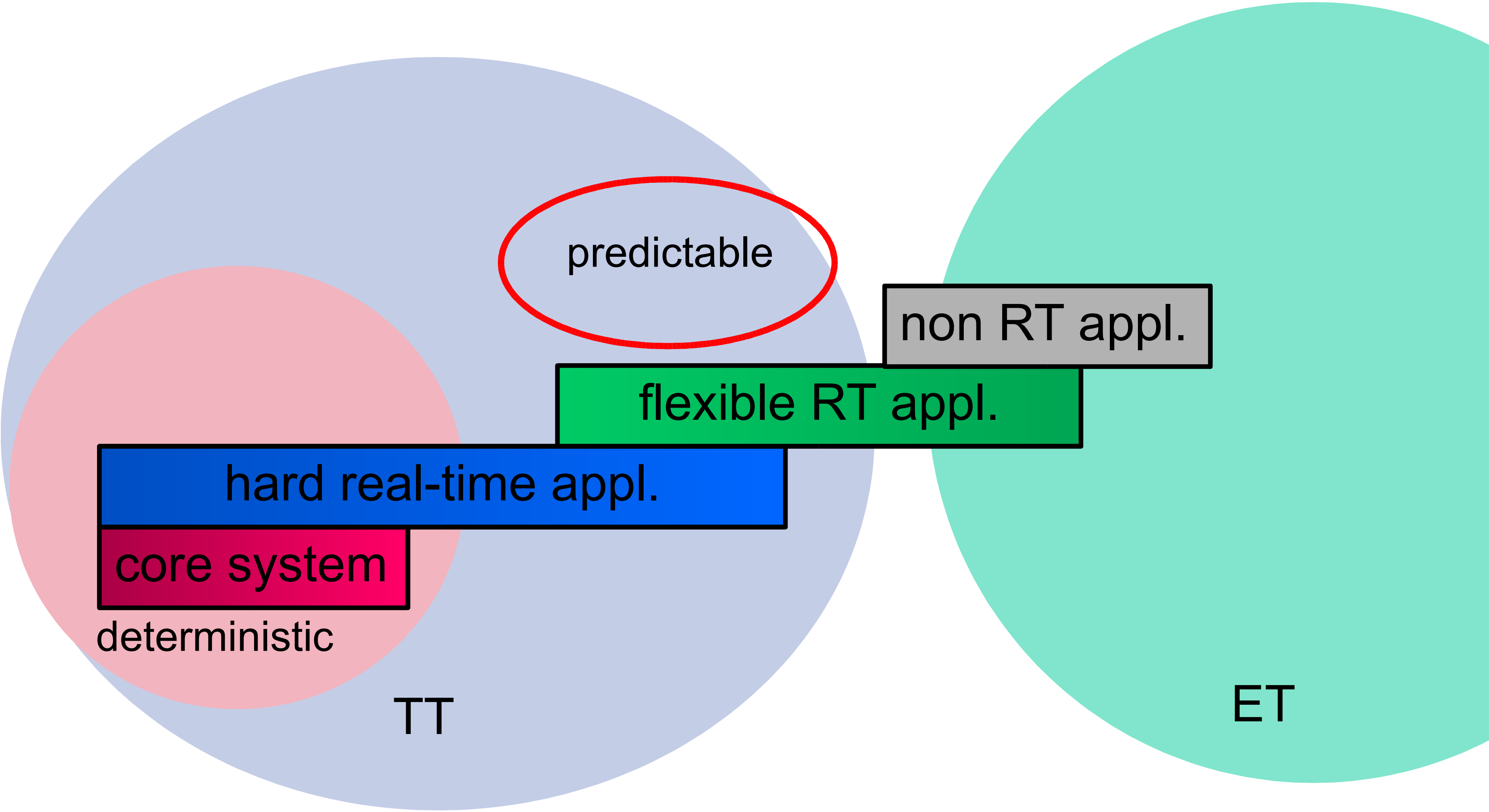
- target window = original task execution
  no flexibility, original schedule
- target window after flexibility analysis
  flexibility of execution while meeting demands
- reduced target windows
  reduced flexibility, e.g., for jitter control
- modifying target windows selects flexibility of tasks individually

Meeting Novel Application Demands

- core system
  offline scheduling
- hard real-time applications
  offline scheduling or online scheduling
- flexible real-time applications
  combined offline/online approach
- non real-time activities
  together with combined offline/online
- flexibility individually configured

- guaranteed tasks protected

Predictably flexible real-time systems


predictable
non RT appl.
flexible RT appl.
hard real-time appl.
core system
deterministic
TT
ET