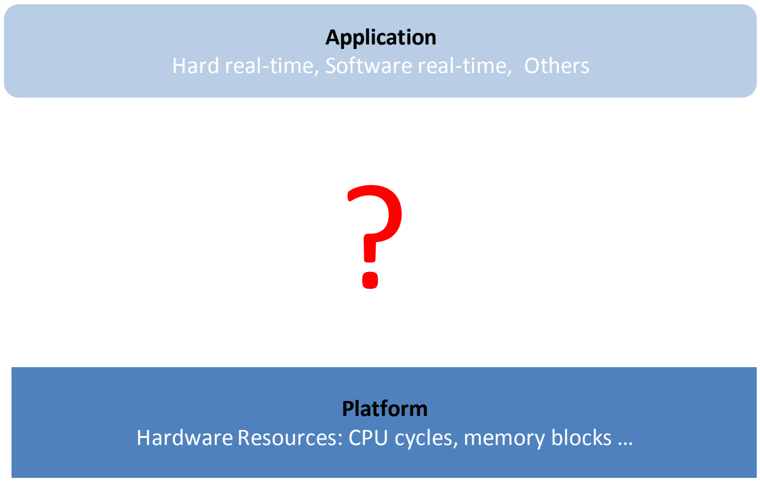
Resource Reservation & Resource Servers

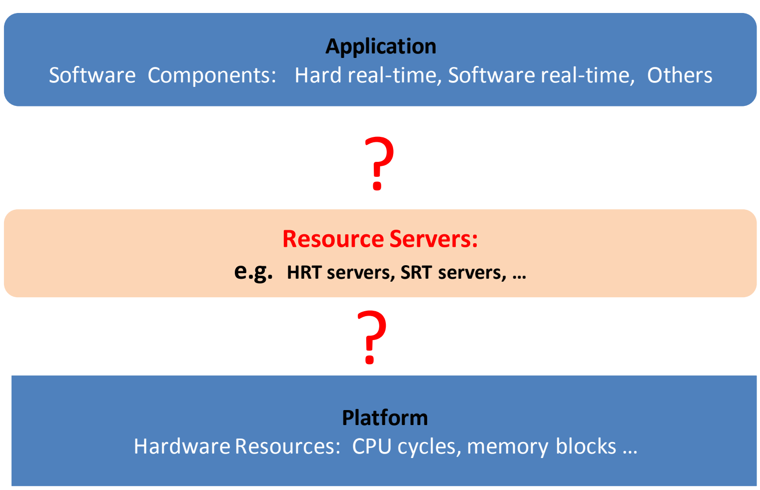
Problems to solve

- Hard-deadline tasks may be
  - Periodic or
  - Sporadic (with a known minimum arrival time) or
  - Non periodic (how to deal with this?)
- Soft-deadline tasks (and/or non RT) may be
  - Various types (periodic or non periodic etc)
- We want to shedule the mixed task set so that
  - All hard tasks meet their deadlines
  - All soft tasks get average response times as low as possible
    - Best average performance

Resource Reservation



Resource Reservation/Virtual Resources



Resource reservation by "servers"

- Creating a *periodic server* Ts=(Cs, Ps) for processing aperiodic workload and a queue to buffer tasks that arrive at different times
  - Create one or more servers for different classes of tasks
- Non-periodic tasks are scheduled in the periodic server's time slots. This policy could be based on deadline, arrival time, or computation time.
- Algorithms – all algorithms behave in the same manner when there are enough aperiodic tasks to execute
  - Polling Server (bandwidth non-preserving)
  - Deferrable Server (bandwidth preserving)
  - Priority Exchange Server (bandwidth preserving)

5

Polling Server (PS)

- Idea:
  - Consider that all hard tasks are periodic
  - Create a periodic task (a server) with period Ts and capacity Cs (the allowed computing time in each period)
  - Schedule the server as a periodic task (Cs, Ts)
- Run time behaviour:
  - Once the server is active, it serves all pending (buffered) aperiodic requests within its capacity Cs according to other algorithms e.g FCFS, SJF etc
  - If no aperiodic requests, the capacity is lost: if a request arrives after the server has been suspended, it must wait until the next polling period

6

Deferrable server (PS preserving capacity) [Lehoczky and Sha et al, 87,95]

- It is similar to Polling server
- The only difference is that the capacity of DS will be preserved if no pending requests upon the activation of the server. The capacity is maintained until the end of the server
  - within the period, an aperiodic request will be served; thus improving average response time

Priority Exchange (interesting!)

- Similar to PS and DS, PE has a periodic server (usually with high priority) for serving aperiodic tasks. The difference is in the way how the capacity of the server is preserved
- Run Time Behaviour:
  - If the PE server is currently the task with highest priority but there is no aperiodic request pending, then
    - the periodic task with next highest priority runs and
    - the server is assigned with the periodic task's lower priority
  - Thus the capacity of the server is not lost but preserved with a lower priority (the exchange continues until new aperiodic requests arrive)

So far, we should know

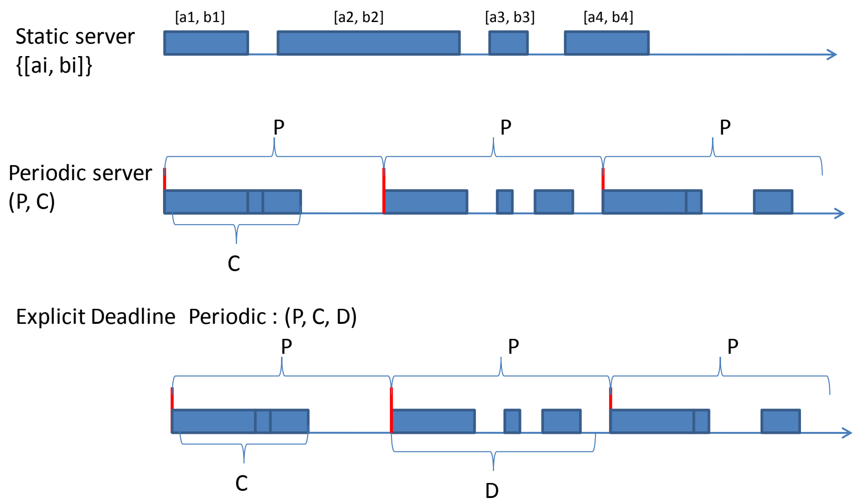
- How to schedule non-periodic task sets
  - Optimal scheduling algorithms
  - Precedence constraints
- How to schedule periodic task sets
  - Schedulability tests
  - Calculation of response time
- How to schedule mixed task sets
  - Ensure the timing constraints of non-periodic RT tasks
  - Improve response times for soft tasks
- How to avoid un-bounded priority inversion
  - Resource access protocols
  - Calculation of blocking time

9

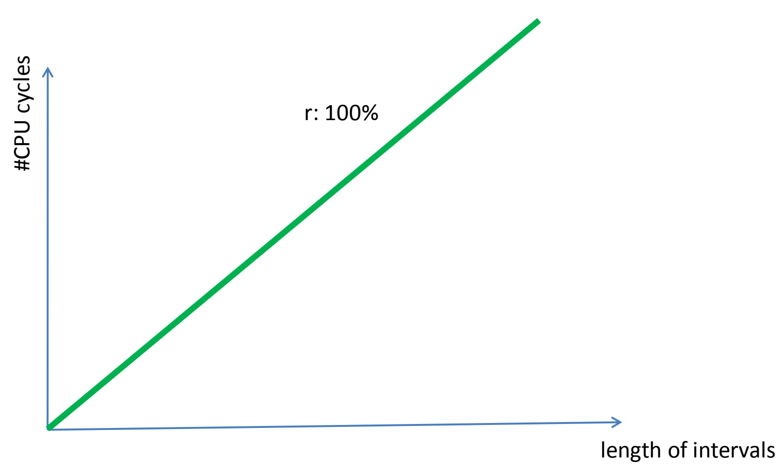
Implementation of Resource Servers

- Periodic Servers
  - Servers as periodic tasks:
    - Hart Real-Time: constrained deadline
    - Soft Real-Time: arbitrary deadline/non-deadline
- Other servers
  - E.g. Bounded Delay Server
    - Round Robin + Worst Case Delay

Resource server: Example



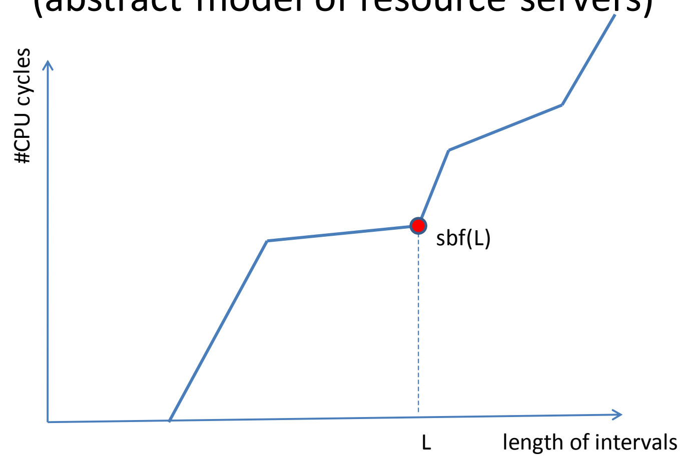
The SBF of a ”perfect” processor



Over L ”cycles”, you are guaranteed at least L ”cycles”

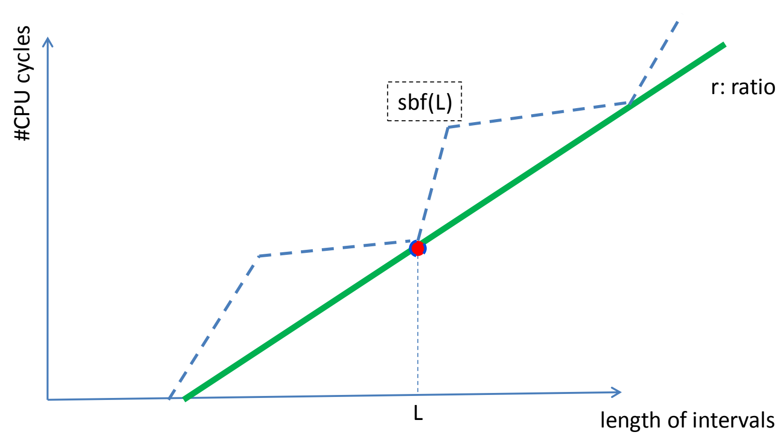
Suply Bound Function (sbf)

(abstract model of resource servers)



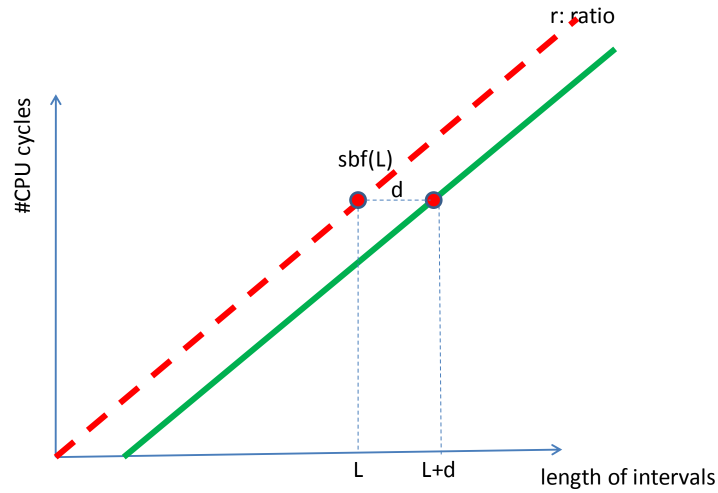
Over L, you are guaranteed at least sbf(L) cycles

Linear Lower Bound of SBF



Over L, you are guaranteed at least sbf(L) cycles

Bounded Delay Server: (r,d)



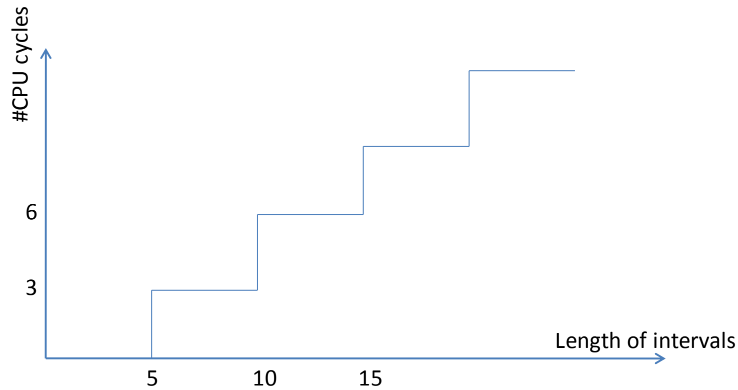
Over L, you are guaranteed at least sbf(L) cycles

Workload models

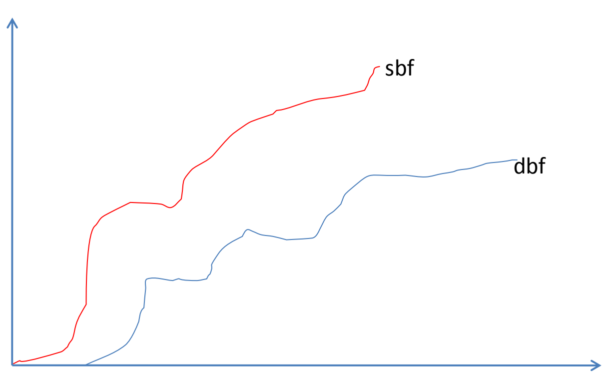
- Non-periodic tasks
- Periodic tasks
- Sporadic tasks

- Demand bound function: dbf(t)
  - over time interval t, the accumulated amount of work to be computed

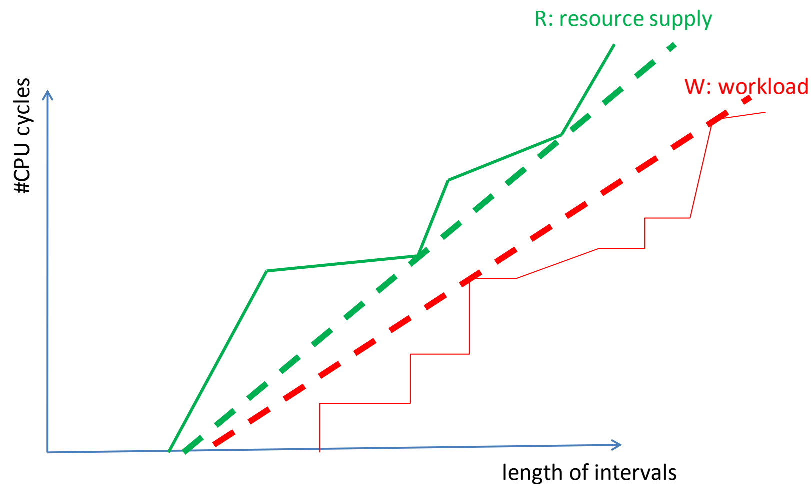
Example: Periodic task



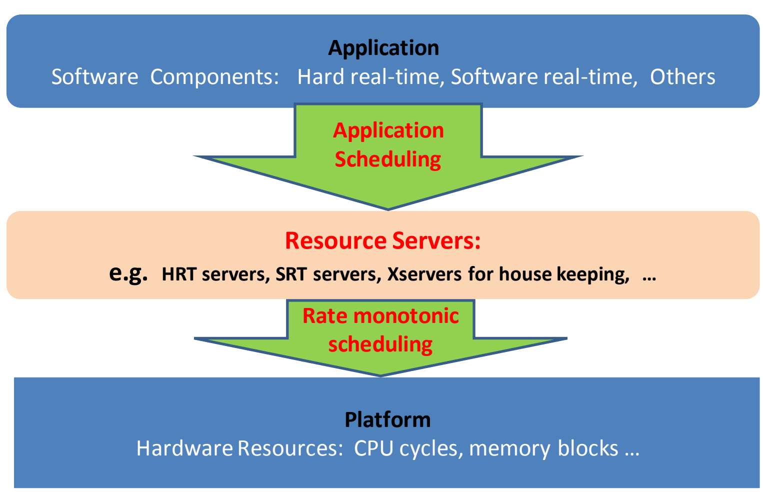
Schedulability analysis: for all t, dbf(t) <= sbf(t)



Schedulability



Resource Reservation/Virtual Resources



Application Scheduling

- Given
  - a group of applications e.g. HRT apps,
  - a Resource Server
- Design a Scheduler s.t.
  - no deadline miss

Application Schedulers

- Non Real-Time Applications
  - Schedule to run according to the budgets allocated by the servers e.g "60% resource rate"
- Real-Time Application: (W, R, A)
  - W: Workload model e.g. dbf
  - R: Resource server e.g. sbf
  - A: Scheduling algorithm e,g, EDF
- Research Questions
  - Schedulability analysis
    - dbf(t)<= sbf(t) for all t (Necessary condition for all A's)
  - Synthesis problems e.g.
    - Given W, R, find A s.t. W is schedulable
    - Given W, A, find the "minimal" R s.t. W is schedulable.
    - Given W, R, A: calculate R*: the remaining resource after W

Hierarchical Application Scheduling

- Given
  - a collection of applications: workloads
  - a set of Servers
- Decompose
  - the apps into "sub-apps"
  - Servers into "sub-servers"

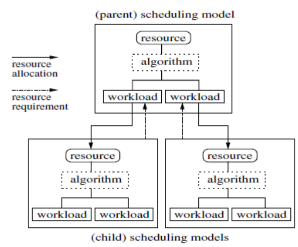


Figure 1. Hierarchcial scheduling framework: parent and children scheduling models.

Resource Virtualization (Buttazo et al)

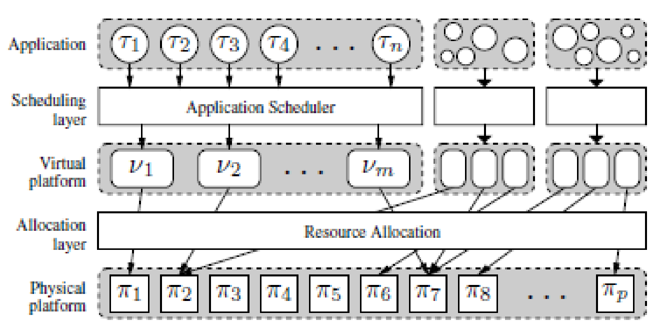


Fig. 1. Architecture overview.