INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

Retis

Real-Time Systems Laboratory

Need for Reservation Servers with Constrained Deadlines

Daniel Casini, Alessandro Biondi, and Giorgio Buttazzo

Scuola Superiore Sant'Anna – ReTiS Laboratory

Pisa, Italy

Why using constrained-deadlines?

Recent work showed that Semi-Partitioned scheduling can achieve high schedulability performance:

- "*Global Scheduling Not Required*" by Brandenburg and Gul for static workloads (RTSS 2016)

- "*Semi-Partitioned Scheduling of Dynamic Real-Time Workload*" by Casini et al. for dynamic workloads
  (29th June, 15:30 PM @ ECRTS 2017)

Why using constrained-deadlines?

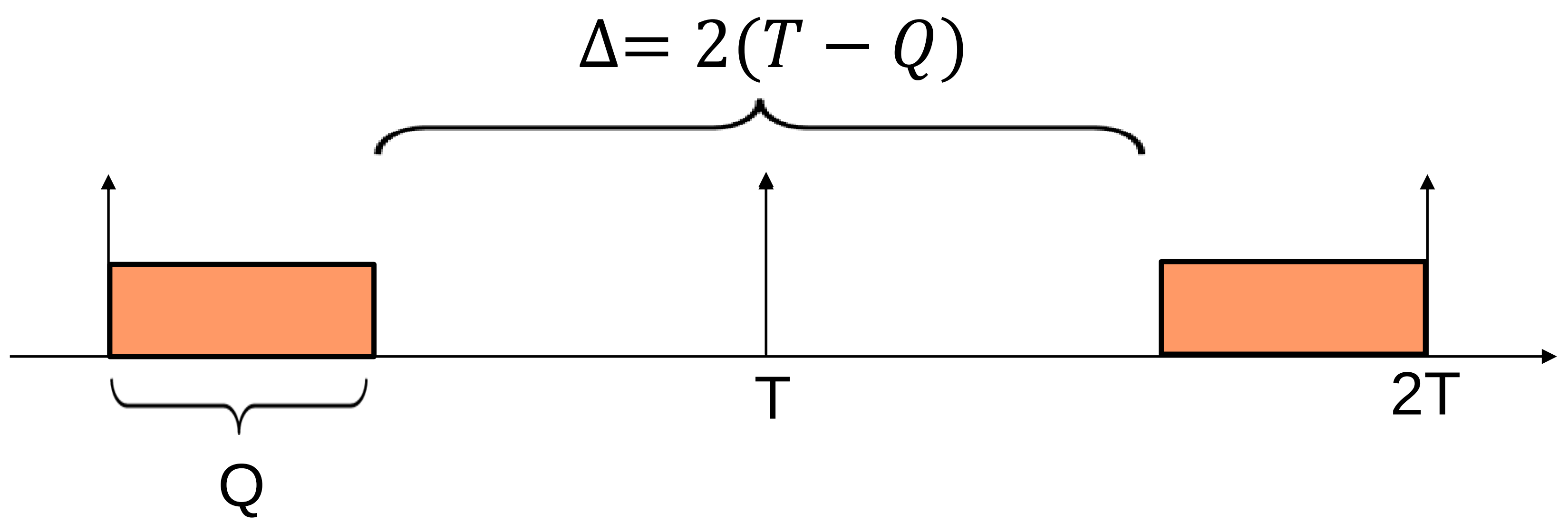
- Supporting constrained-deadlines is an open problem also for the SCHED_DEADLINE scheduling class of Linux (based on reservation with the **CBS** algorithm)

- Currently discussed also in the Linux kernel mailing list

Hard Constant Bandwidth Server

- H-CBS is a reservation algorithm allowing to guarantee:
  - A bandwidth $\alpha = \frac{Q}{T}$
  - A bounded maximum service-delay $\Delta = 2(T - Q)$

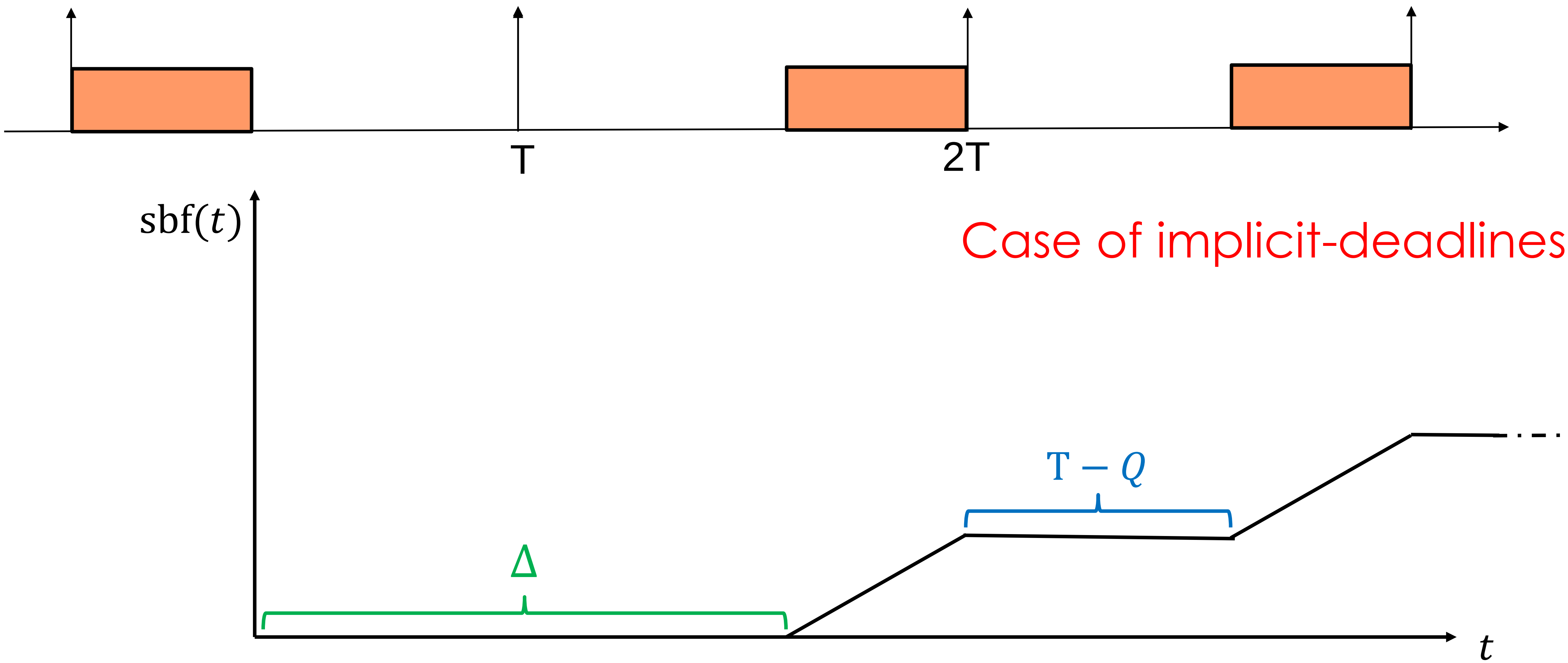
Worst-case scenario for the service delay



Used in several works and implementations

Importance of a bounded delay

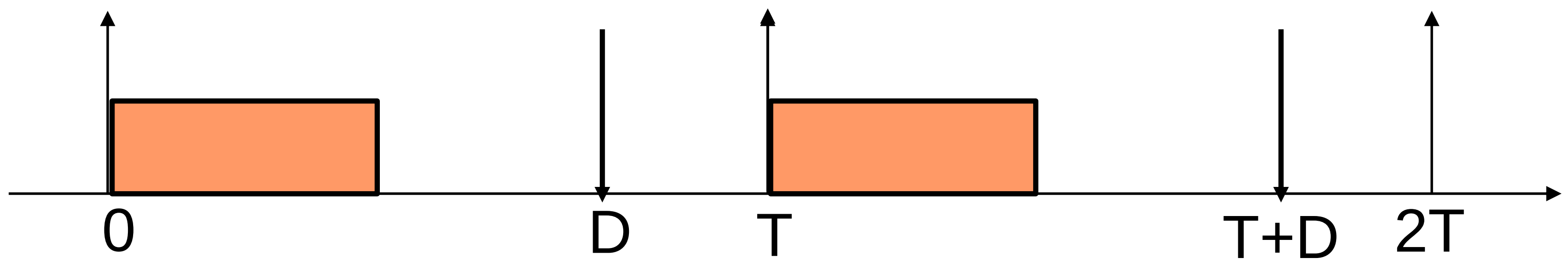
A bounded-delay allows deriving a supply function that can be used for testing the schedulability of the workload running inside the server:



H-CBS and constrained-deadlines

- As long as the server behaves (in the worst-case) as a standard periodic/sporadic task with constrained deadlines, existing EDF schedulability theory can be applied

- The core issue is how to guarantee that the demand generated by the server never exceeds the one of a corresponding sporadic task in all possible scenarios…



H-CBS key rule

- H-CBS has a specific rule when the server wakes up from the idle state:
  - **Rule 2:** "*When H-CBS is idle and a job arrives at time t, a replenishment time is computed as* $t_r = d - \frac{q}{\alpha}$"
    - Then, if $t < t_r$ the server is suspended until time t, where the budget is replenished and the absolute deadline is postponed to time $t_r + T$;
    - otherwise, the budget is immediately replenished and the absolute deadline is postponed to $\mathrm{t} + T$.

H-CBS and constrained-deadlines

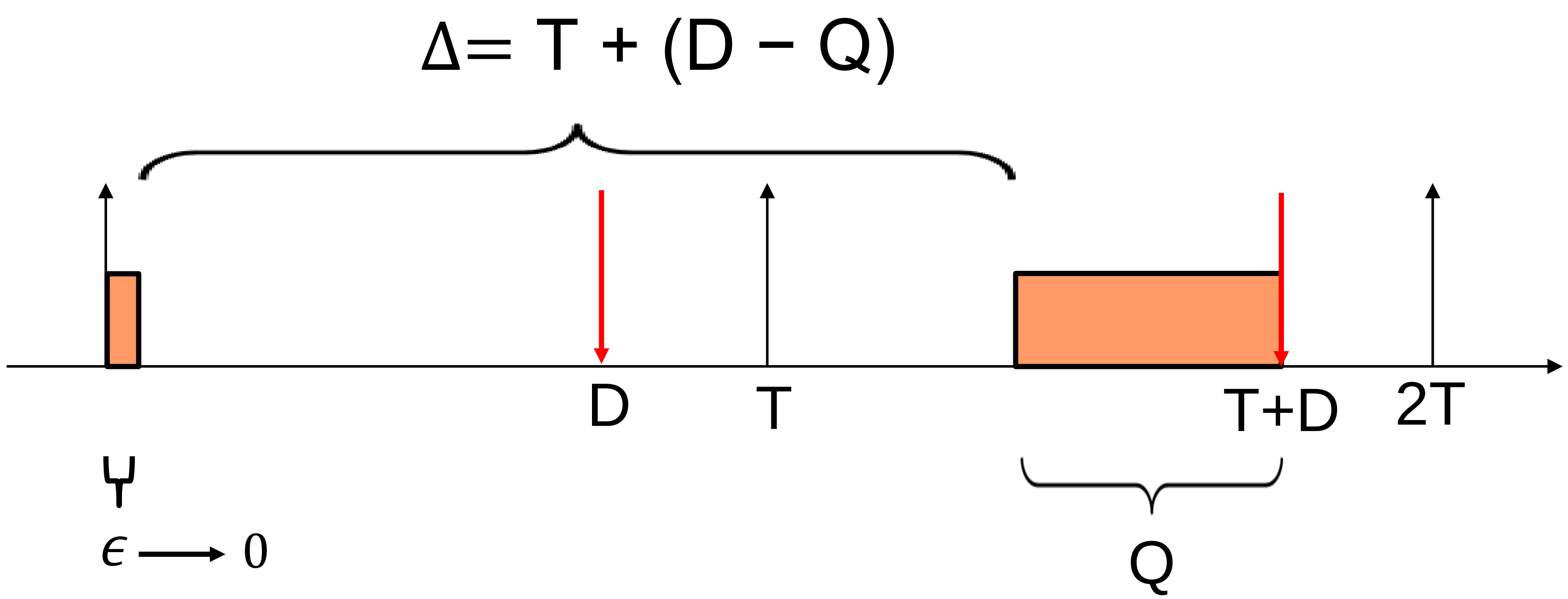
- H-CBS rules are not directly applicable in case of constrained-deadlines:
  - **Rule 2:** "*When H-CBS is idle and a job arrives at time t, a replenishment time is computed as* $t_r = d - \frac{q}{\alpha}$"

This rule has been derived by **EDF** schedulability theory for **implicit-deadline tasks** (*utilization-based*), which indeed cannot be re-used to ensure schedulability with constrained deadlines!

Naïve solution

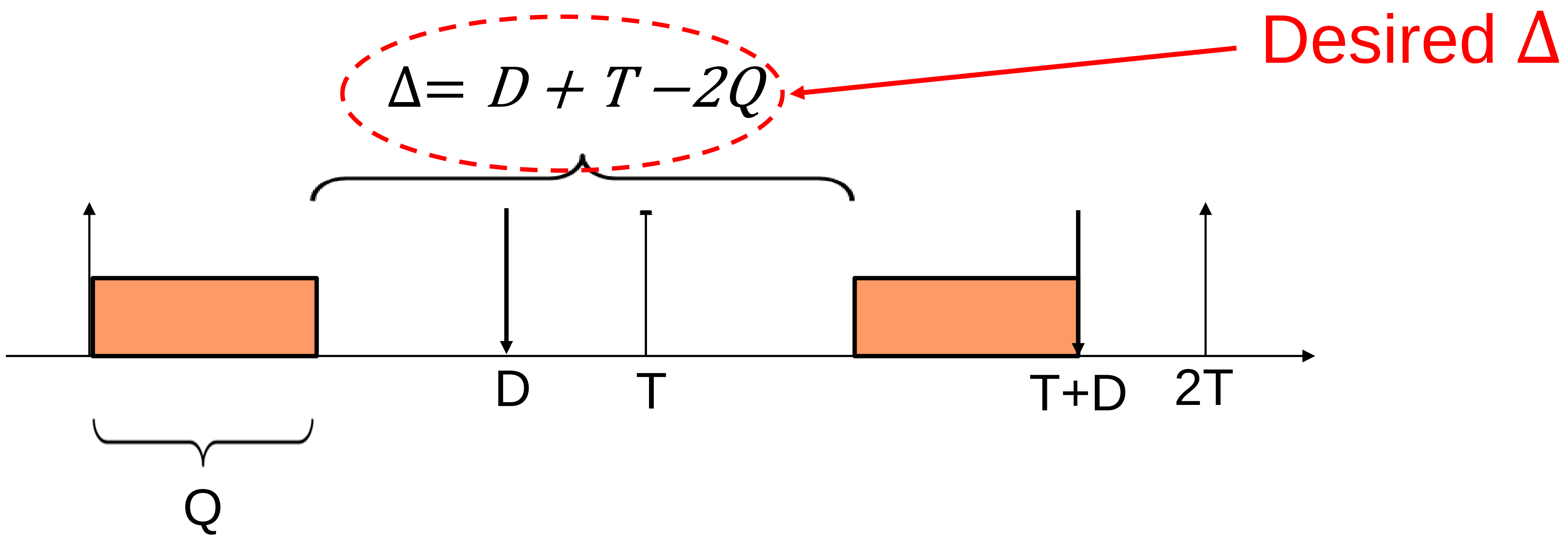
Mimic the polling server

- **New Rule:** "*When H-CBS goes IDLE, discard all the budget. The budget is replenishment only at server periods, i.e.,* $t_r = kT_i$"



The worst-case service delay is much **higher**!

Questions



How to modify the replenishment rules for obtaining a better maximum-service delay?

Is it possible to achieve a maximum service delay equal to $\Delta = D + T - 2Q$?

Issues with shared resources

BROE

- ✓ Avoids budget overruns
- ✓ Ensures bandwidth isolation
- ✓ Guarantees bounded-delay

The protocol is based on a proportional deadline-postponement rule which relies on the server bandwidth

(again, EDF schedulability theory for <u>implicit-deadlines</u>)

Issues with shared resources

BROE

- ✓ Avoids budget overruns
- ✓ Ensures bandwidth isolation
- ✓ Guarantees bounded-delay

How to guarantee a bounded-delay partition in the presence of shared-resources?

Issues with admission control

- Replenishment rules are based on the admission test, so another question arise:

Which admission control test should be used for admitting reservations?

- We expect that the adopted admission test will strongly influence the server rules
- An efficient (and hence possibly sufficient) admission test would also reduce the server run-time overhead

Issues with admission control

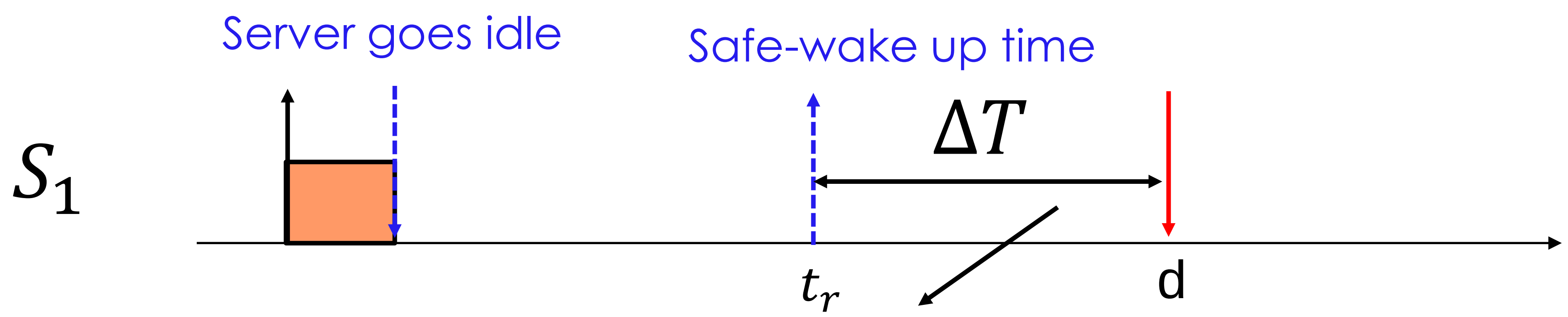
- With implicit-deadline the admission test of the H-CBS (based on EDF) is very simple:

$$\boxed{\sum \alpha_i \leq 1}$$

Exact test

Constant-time complexity

- This is relevant to our purpose because the H-CBS rule builds upon the schedulability test



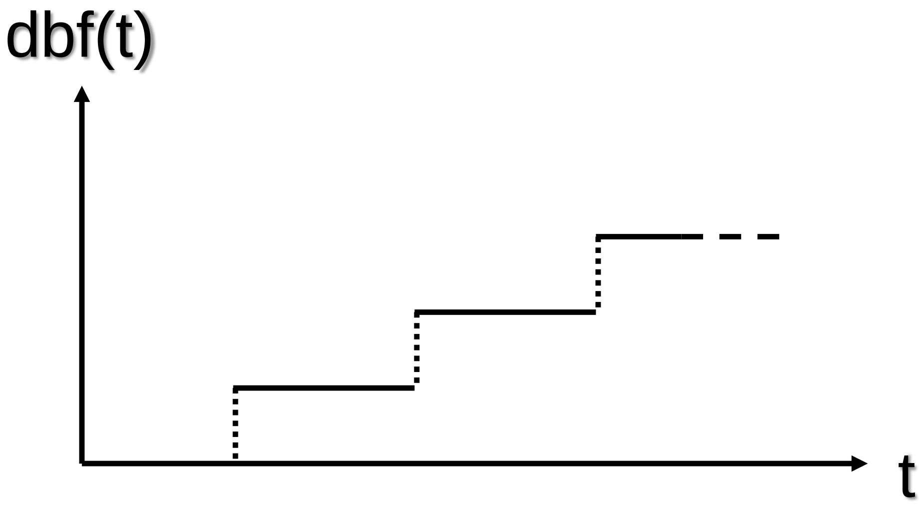
What is the t_r which guarantees a bandwidth α in ΔT?

$$\alpha \Delta T = q \rightarrow \alpha(d - t_r) = q \rightarrow t_r = d - \frac{q}{\alpha}$$

Issues with admission control

- Conversely, considering constrained-deadlines the schedulability check is based on Processor Demand Criterion (*Baruah et al. 1990*)

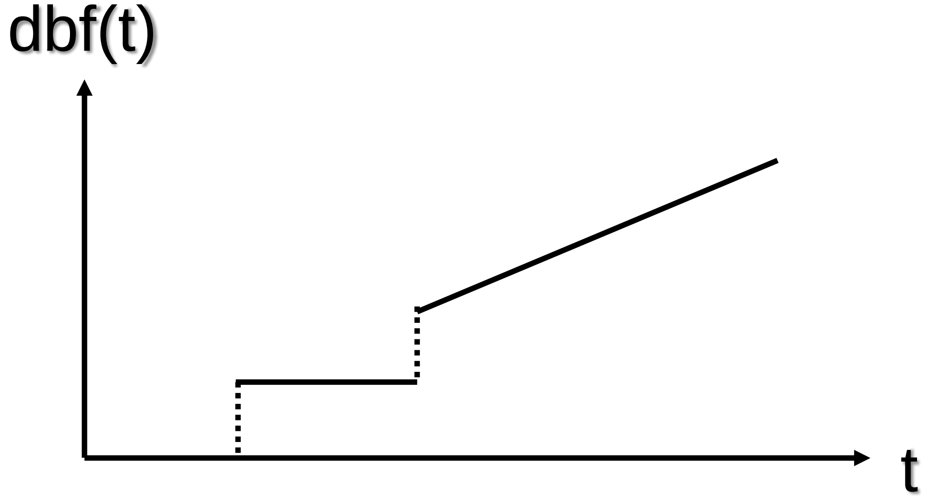
Based on demand bound functions



Exact test, Pseudo-polynomial complexity if $\sum \alpha_i < 1$

Issues with admission control

- Some approximations exist to limit the computational complexity of the admission-test
- They are based on approximating the demand-bound function with a fixed number of discontinuities (*Fisher et al., 2006*)



Polynomial-time complexity (sufficient test)

Questions

How to modify the replenishment rules for obtaining a better maximum-service delay?

Is it possible to achieve a maximum service delay equal to $\Delta = D + T - 2Q$?

How to guarantee a bounded-delay partion in presence of shared-resources?

Which admission control test should be used for admitting reservations?

THE QUESTION

How to implement a new Hard Constant Bandwidth Server supporting constrained-deadlines?

SKETCH OF SOLUTION:
SHADOW BUDGETING

Sketch of solution

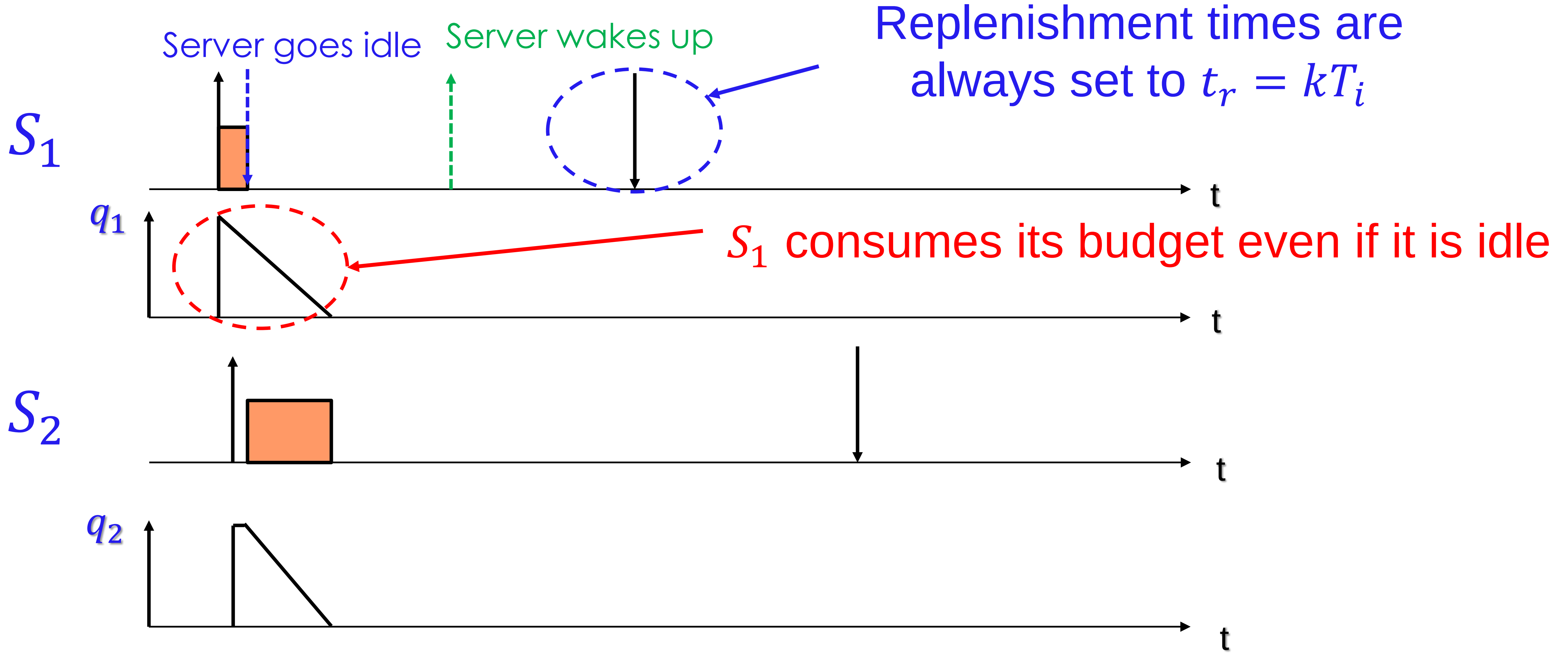
- The results proposed by Biondi et al. for real-time self-suspending tasks can be used to derive a solution

> Alessandro Biondi, Alessio Balsini, and Mauro Marinoni, "*Resource reservation for real-time self-suspending tasks: theory and practice*" (RTNS 2015)

- According to their approach, whenever a server should execute according to EDF scheduling, it consumes its budget independently whether it is suspended or not

Shadow budget

- A similar approach can be adopted when a reservation goes idle:



Pro and Cons

- Simplicity
- Worst-case service delay is smallest as possible
- Independent from the admission test
- Lower throughtput (average-case)
- Still do not consider shared resources

What are we doing?

- Evaluation of different solutions
- Simulations
- Derive methodologies to increase the throughtput
- Develop a solution to cope with shared resources
- Implement the new resource reservation server in Linux (SCHED_DEADLINE)

Thank you!

Daniel Casini
daniel.casini@sssup.it