Constant Bandwidth Servers with Constrained Deadlines

Daniel Casini, Luca Abeni, Alessandro Biondi, Tommaso Cucinotta, and Giorgio Buttazzo

Scuola Superiore Sant’Anna – ReTiS Laboratory

Pisa, Italy

This talk in a nutshell

1

Challanges in designing a reservation servers with constrained deadlines

2

Three new different algorithms

3

Simulation study to assess their performance

Why using constrained-deadlines?

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

- "*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 (ECRTS 2017)

Why using constrained-deadlines?

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

- "*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 (ECRTS 2017)

Both requires constrained-deadline (C=D) reservations!

Why using constrained-deadlines?

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

- Currently discussed also in the Linux kernel mailing list

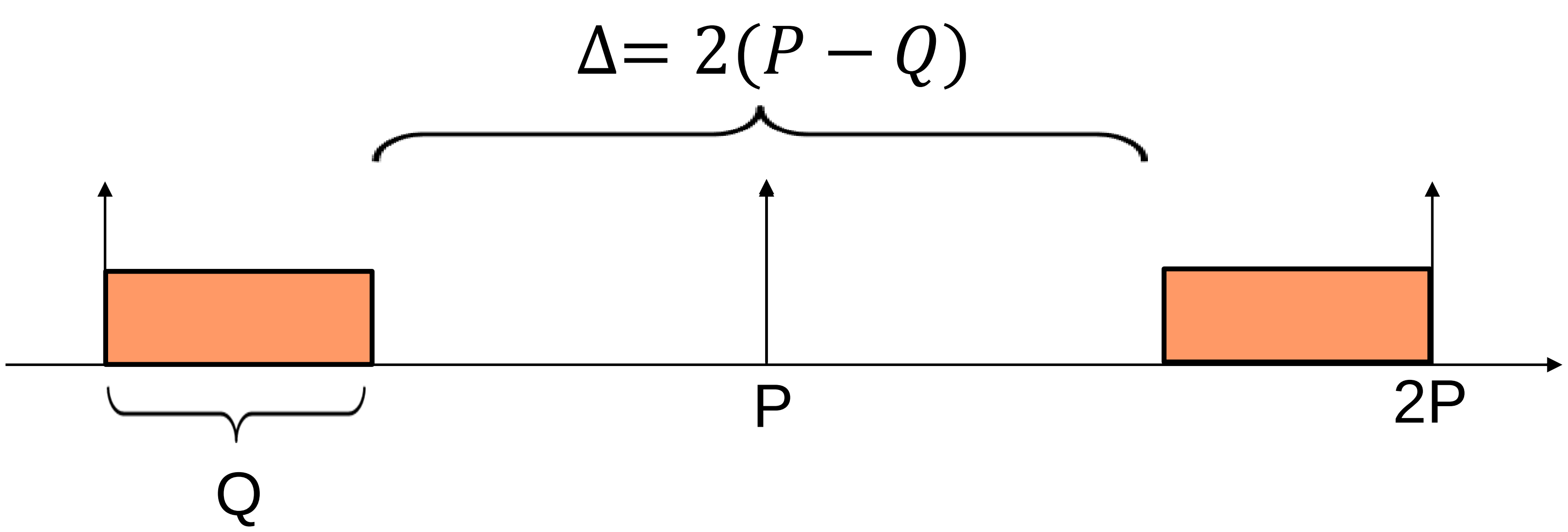
Linux

android

Hard Constant Bandwidth Server

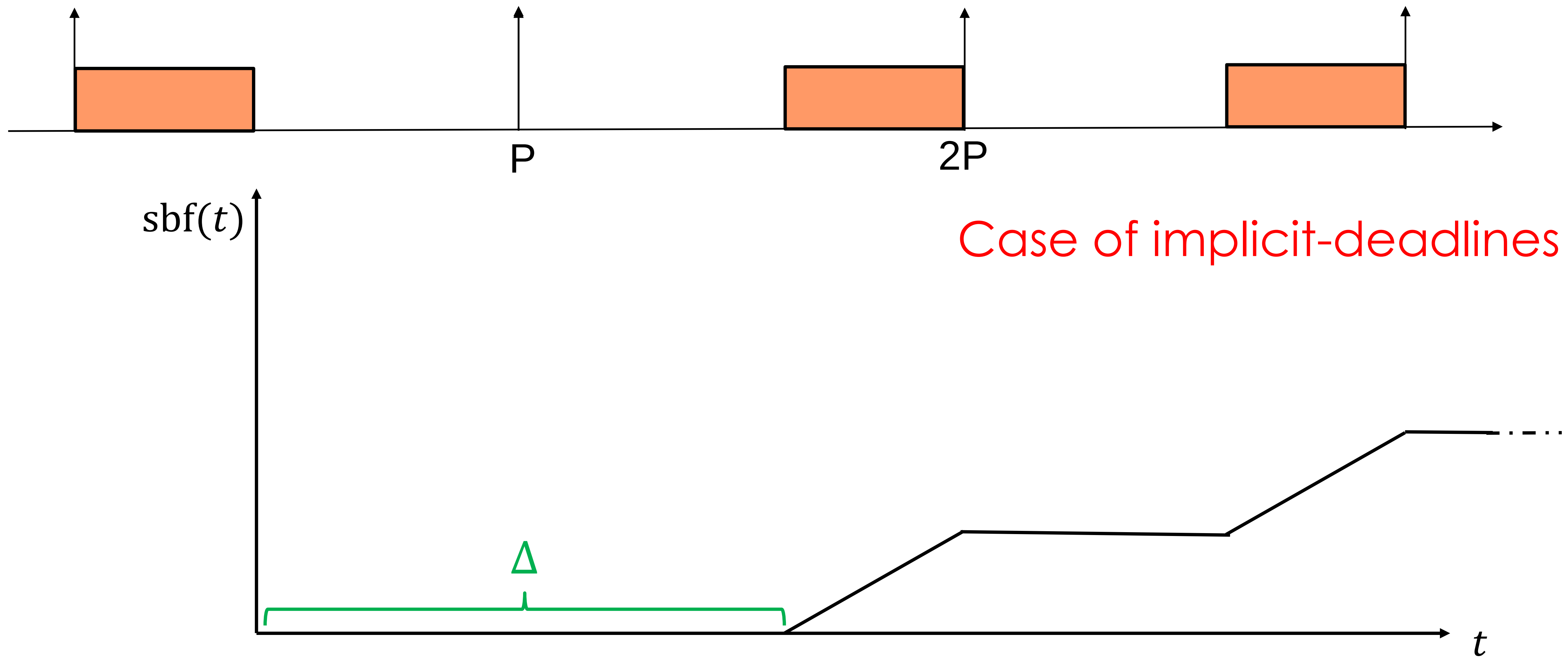
- H-CBS is a reservation algorithm allowing to guarantee:
  - A bandwidth $\alpha = \frac{Q}{P}$
  - A bounded maximum service-delay $\Delta = 2(P - Q)$

Worst-case scenario for the service delay



Importance of a bounded delay

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



H-CBS key rule

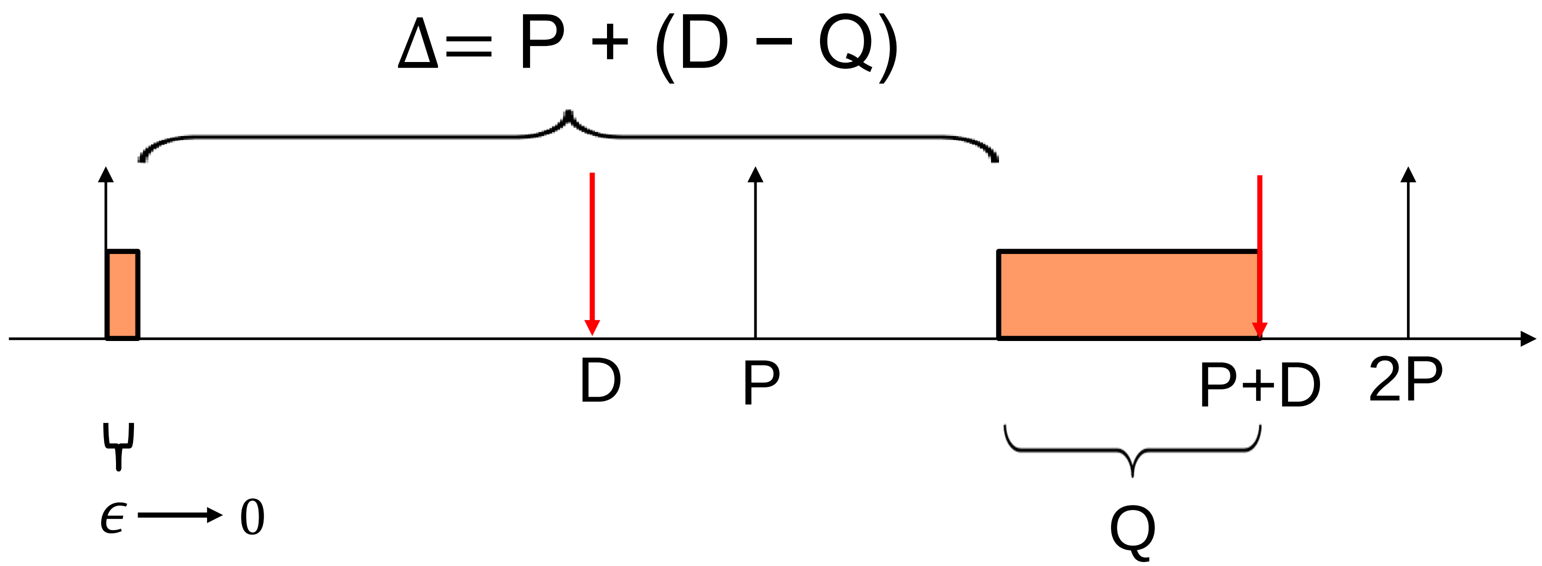
- H-CBS has a specific rule to generate a new budget and scheduling deadline when the server wakes up from the idle state:

$$(\mathrm{q}, \mathrm{d}) = \begin{cases} (q, d) & if\ t < d - q/\alpha \\ (Q, t + P) & otherwise" \end{cases}$$

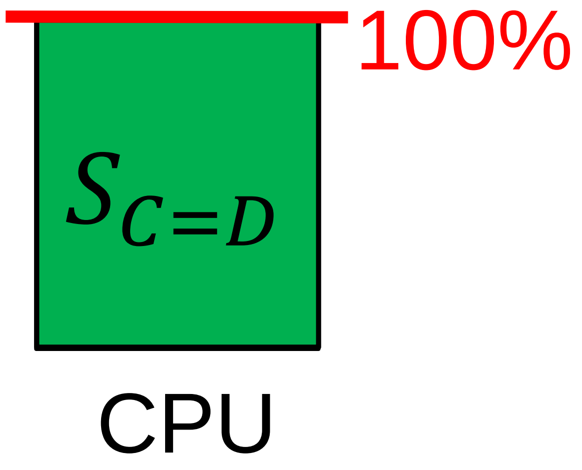
This rule has been derived by **EDF** schedulability theory for **implicit-deadline tasks** (*utilization-based*), which indeed **cannot be re-used** to ensure schedulability with constrained deadlines!

Possible simple solutions

- Mimic the polling server
  - Higher worst-case delay!



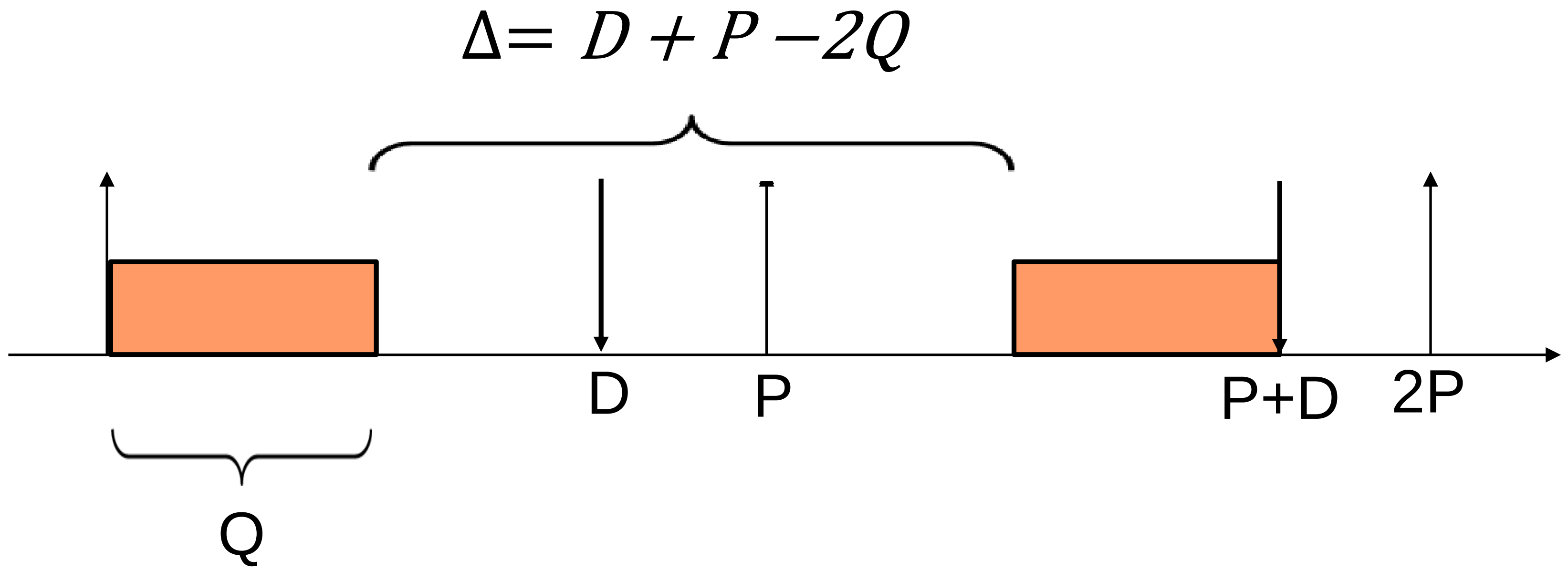
- Configure H-CBS to use D in place of P
  - High pessimism!



Design Issues

How to modify the server rules for providing a given bandwidth and a better maximum-service delay, without sacrificing schedulability?

How to achieve a maximum service delay equal to $\Delta = D + P - 2Q$?



Our Solutions

- $\boldsymbol{H\text{-}CBS^D\text{-}W}$ (H-CBS Deadline – Worst Case)
  - Our solution for hard real-time systems
- $\boldsymbol{H\text{-}CBS^D}$ (H-CBS Deadline)
  - Our solution to improve average-case performance for soft real-time systems
- $\boldsymbol{H\text{-}CBS^D\text{-}R}$ (H-CBS Deadline - Reclaiming)
  - Extends $\mathrm{H\text{-}CBS^D}$ with reclaiming

H-CBS^D-W: Idea

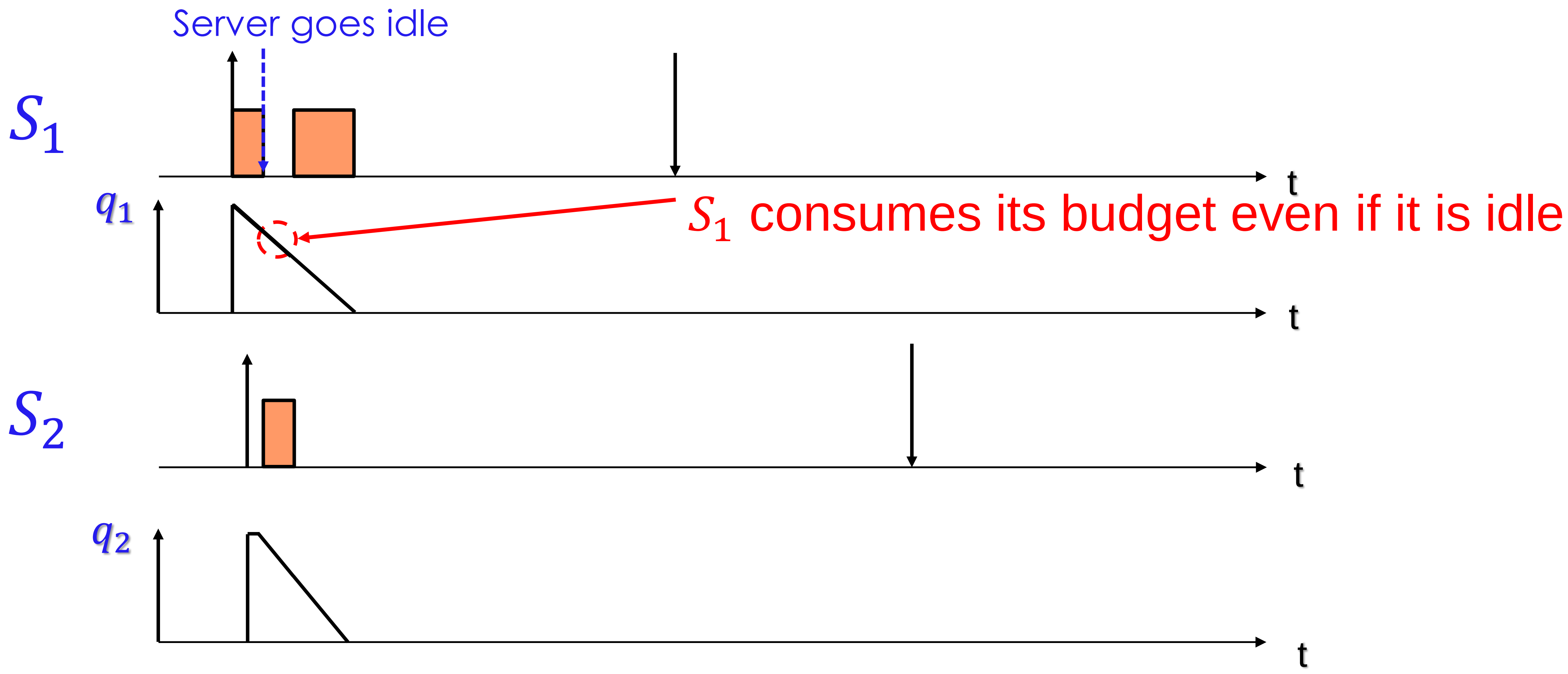
- H-CBS^D-W leverages the results proposed by Biondi et al. for real-time self-suspending tasks

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 self-suspended or not**

$H\text{-}CBS^D\text{-}W$: Idea

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



$H\text{-}CBS^{D}\text{-}W$: Evaluation

- **PROS:**
  - It guarantees a bandwidth α and a bounded delay $\Delta = D + \mathrm{P} - 2Q$
  - $\text{H-CBS}^{\text{D}}\text{-W}$ is independent from the adopted schedulability test!
- **CONS:**
  - It requires the implementation of an additional server queue to keep track of suspended servers

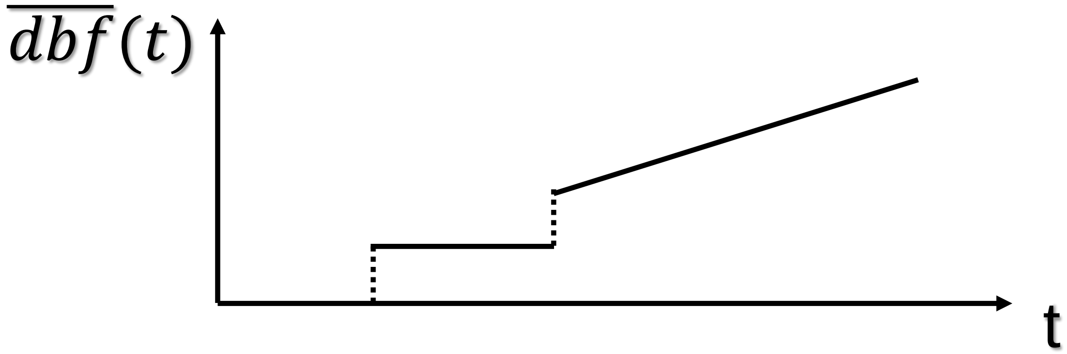
Room for improvement in terms of average-case performance and soft real-time metrics

CAN WE DO BETTER (IN TERMS OF AVERAGE-CASE PERFORMANCE AND SOFT REAL-TIME METRICS) BY LEVERAGING A SPECIFIC SCHEDULABILITY ANALYSIS?

THE $H-CBS^{D}$ ALGORITHM

$H\text{-}CBS^{D}$ Idea

- Assign current budget and deadline by means of an online schedulability test based on approximated demand bound functions, but…



Leveraging also the knowledge of online parameters!

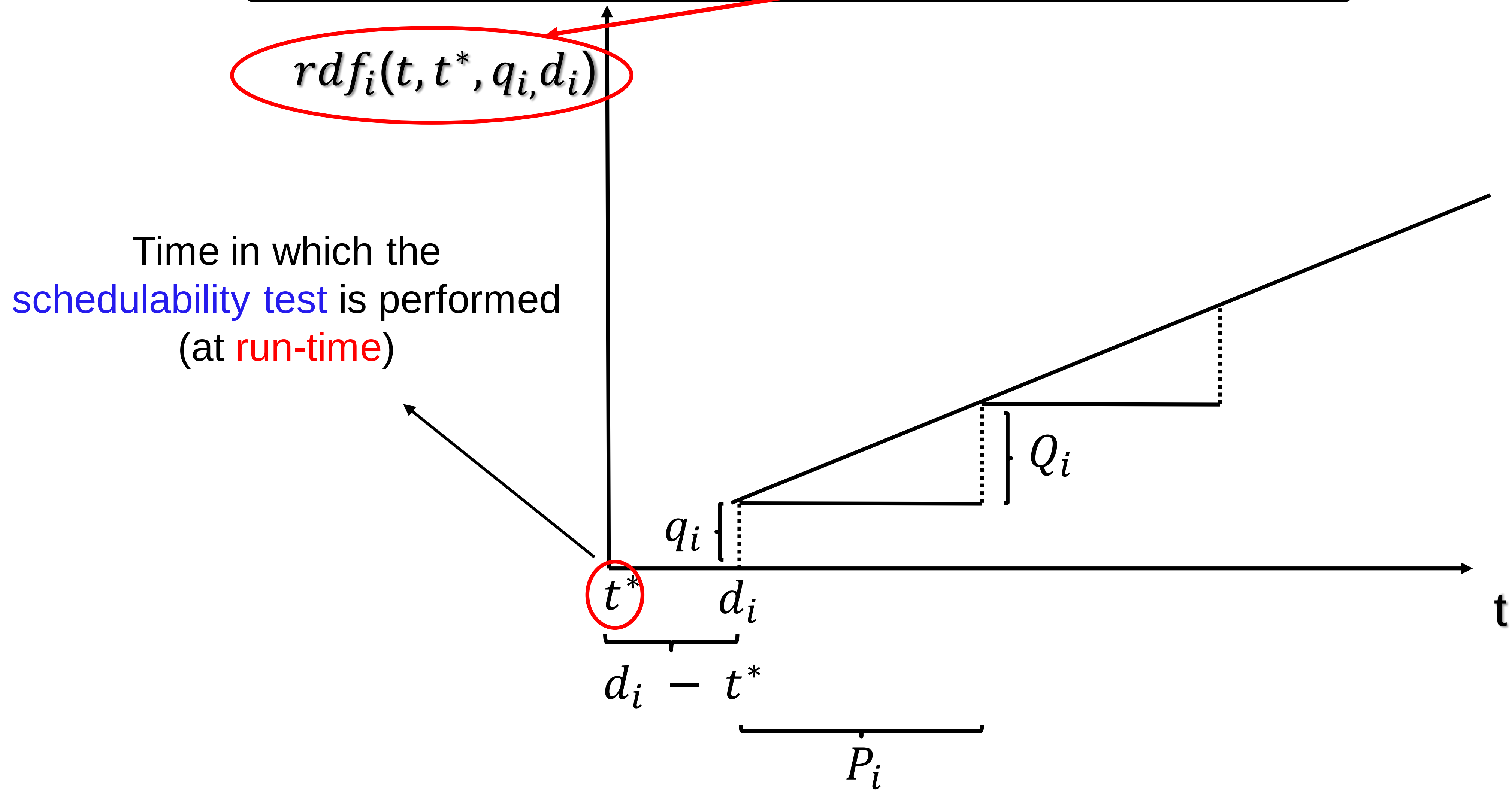
The current budget q_i

The scheduling deadline d_i

$H\text{-}CBS^D$ Idea

- From the knowledge of such online variables we derived a schedulability test based on the following abstraction of the workload:

Run-time demand function



$H\text{-}CBS^D$ key rule

- $H\text{-}CBS^D$ assigns a new budget and scheduling deadline when the server wakes up from the idle state based on the following idea:

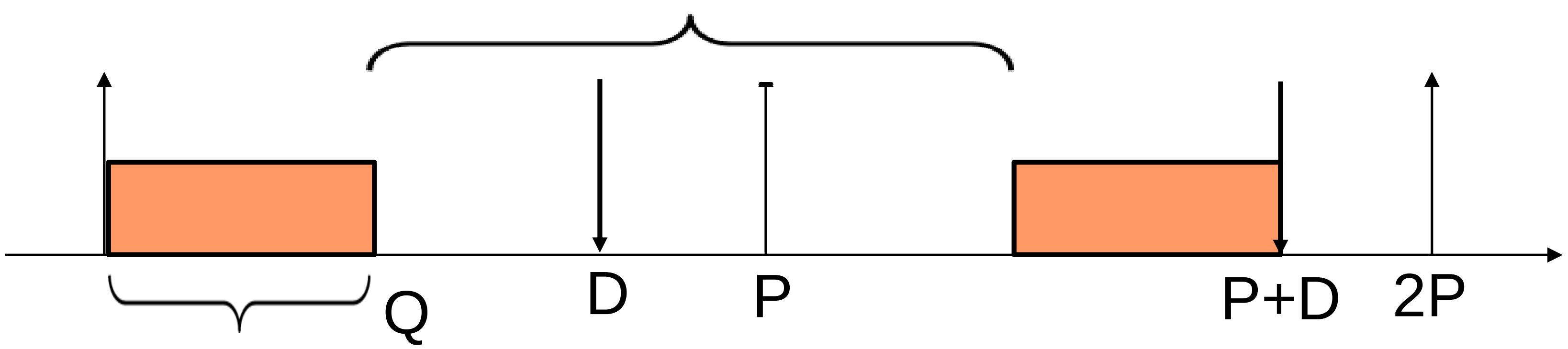
$$\begin{cases} \text{re-use } (q, d) & \text{if rdf-based test holds} \\ q=Q \text{ with } d=t+D & \text{otherwise} \end{cases}$$

Similar to H-CBS but using a schedulability test for valid constrained-deadline servers

$H\text{−}CBS^D$: Main Properties

- **Bounded-delay:** the algorithm guarantees a bounded worst-case service delay

$$\Delta = D + P - 2Q$$



- Reserves a portion of the processor capacity for idle reservations
  - ✓ A full budget replenishment is guaranteed to each reservation after at least P_i units of time from the last replenishment

A different approach: $H\text{-}CBS^D\text{-}R$

Idea: Let's allow a reservation to take the maximum possible budget which does not break schedulability!

- A reservation adopting $H\text{-}CBS^D\text{-}R$ implicitly implements a budget reclaiming
- Average-case performance and probabilistic metrics could benefit of this approach

(see simulations results…following slides!)

The H-CBS^D-R algorithm

How to do this?

- We leveraged the rdf-based schedulability analysis developed for $\boldsymbol{H-CBS^D}$ to derive a sensitivity analysis:

Find: max q such that schedulability is not violated

SIMULATION RESULTS

http://retis.santannapisa.it/~luca/RTNS17

Reservation generation

- First, we generated reservation bandwidths with the Emberson et al. Task-set generator, with:
  - Periods distributed in [5000; 500000] us
  - Budget obtained as $Q_i = \alpha_i P_i$
  - Relative deadline generated with uniform distribution in $[Q_i + \beta(P_i - Q_i), P_i]$
- The workload running into each reservation consist of a single sporadic task
  - Reservation-set with results to be unschedulable according to the approximated schedulability test have been discarded
  - 100 different reservation-set have been tested

Workload generation

- Each job running into a reservation (i.e. its computation and inter-arrival time) is controlled by:
  - $C_r = \frac{\bar{c}_i}{Q_i}$
  - $a = \frac{\overline{u_i}}{\alpha_i}$, with $\alpha_i = \frac{Q_i}{P_i}$, and $\bar{u}_i = \frac{\bar{c}_i}{\bar{p}_i}$
- Variance of execution (sc) and inter-arrival times (sp)
  - The execution time of each job is uniformely distributed in $[\bar{c}_i - \frac{sc(sz)}{2}, \bar{c}_i + \frac{sc(sz)}{2}]$
  - The inter-arrival time of each job is uniformely distributed in $[\bar{p}_i - \frac{sp(sz)}{2}, \bar{p}_i + \frac{sp(sz)}{2}]$

Workload generation

- Each job running into a reservation (i.e. its computation and inter-arrival time) is controlled by:
  - $C_r = \frac{\bar{c_i}}{Q_i}$

    Ratio between the average ET and the server budget

  - $a = \frac{\bar{u_i}}{\alpha_i}$, with $\alpha_i = \frac{Q_i}{P_i}$, and $\bar{u_i} = \frac{\bar{c_i}}{\bar{p_i}}$

- Variance of execution (sc) and inter-arrival times (sp)
  - The execution time of each job is uniformely distributed in $[\bar{c_i} - \frac{sc(sz)}{2}, \bar{c_i} + \frac{sc(sz)}{2}]$
  - The inter-arrival time of each job is uniformely distributed in $[\bar{p_i} - \frac{sp(sz)}{2}, \bar{p_i} + \frac{sp(sz)}{2}]$

Retis

Workload generation

- Each job running into a reservation (i.e. its computation and inter-arrival time) is controlled by:
  - $C_r = \frac{\bar{c_i}}{Q_i}$
  - $a = \frac{\bar{u_i}}{\alpha_i}$, with $\alpha_i = \frac{Q_i}{P_i}$, and $\bar{u_i} = \frac{\bar{c_i}}{\bar{p_i}}$

Ratio between the average task utilization and the server bandwidth

- Variance of execution (sc) and inter-arrival times (sp)
  - The execution time of each job is uniformely distributed in $[\bar{c_i} - \frac{sc(sz)}{2}, \bar{c_i} + \frac{sc(sz)}{2}]$
  - The inter-arrival time of each job is uniformely distributed in $[\bar{p_i} - \frac{sp(sz)}{2}, \bar{p_i} + \frac{sp(sz)}{2}]$

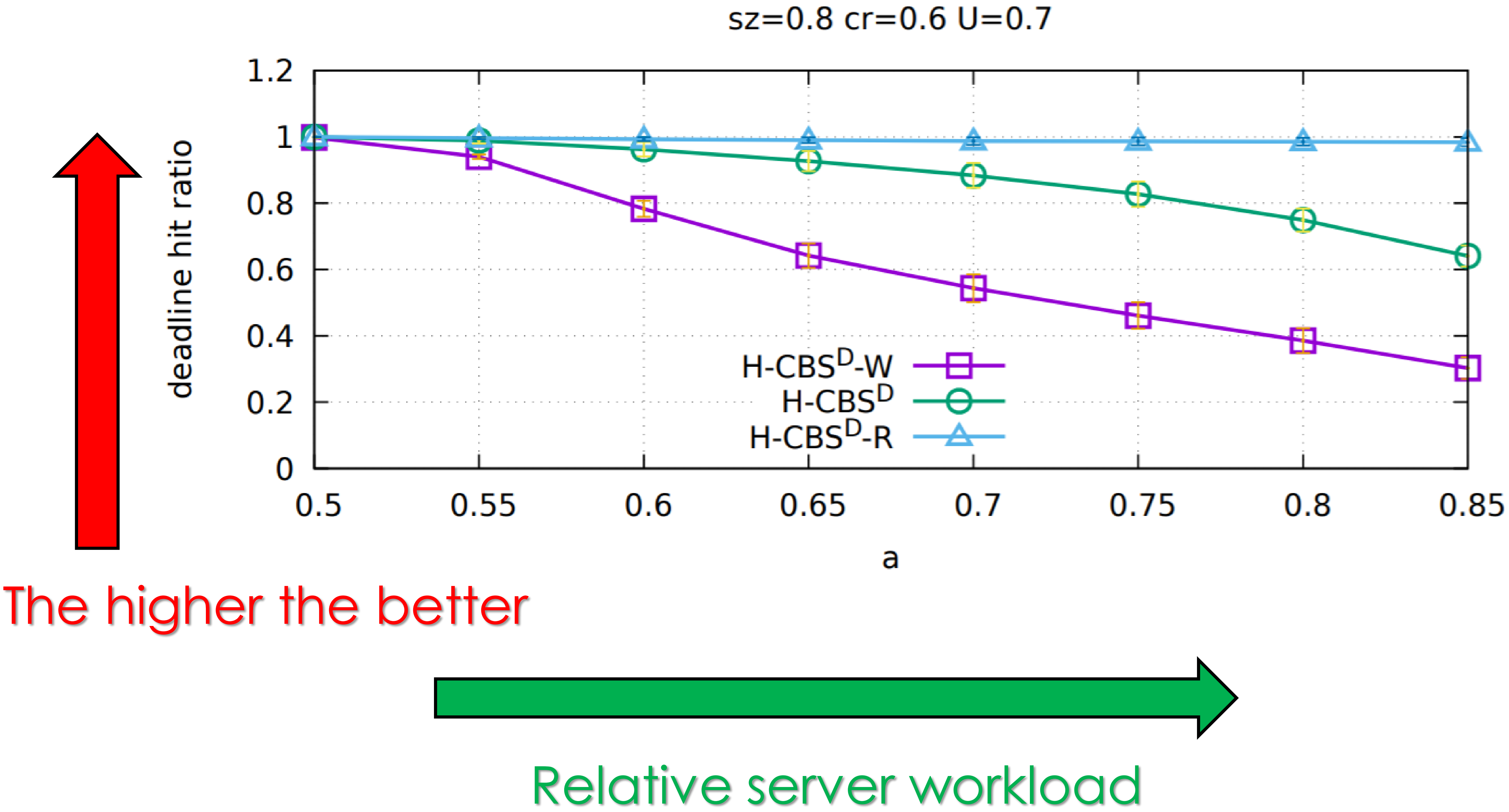
Retis

Workload generation

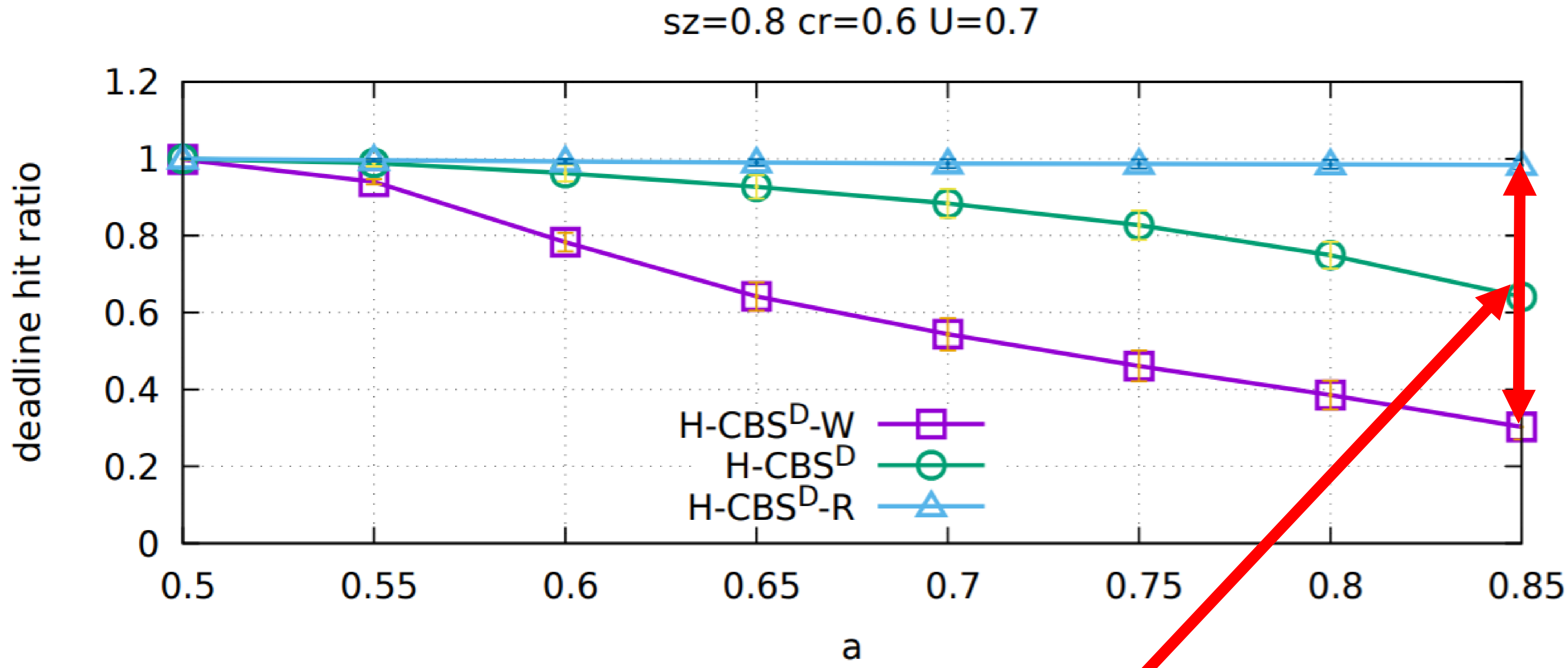
- Each job running into a reservation (i.e. its computation and inter-arrival time) is controlled by:
  - $C_r = \frac{\bar{c_i}}{Q_i}$
  - $a = \frac{\overline{u_i}}{\alpha_i}$, with $\alpha_i = \frac{Q_i}{P_i}$, and $\bar{u_i} = \frac{\bar{c_i}}{\overline{p_i}}$
- Variance of execution (sc) and inter-arrival times (sp)
  - The execution time of each job is uniformely distributed in $[\bar{c_i} - \frac{sc(sz)}{2}, \bar{c_i} + \frac{sc(sz)}{2}]$
  - The inter-arrival time of each job is uniformely distributed in $[\bar{p_i} - \frac{sp(sz)}{2}, \bar{p_i} + \frac{sp(sz)}{2}]$

Variances are function of the sz generation parameter

Simulation Results



Simulation Results

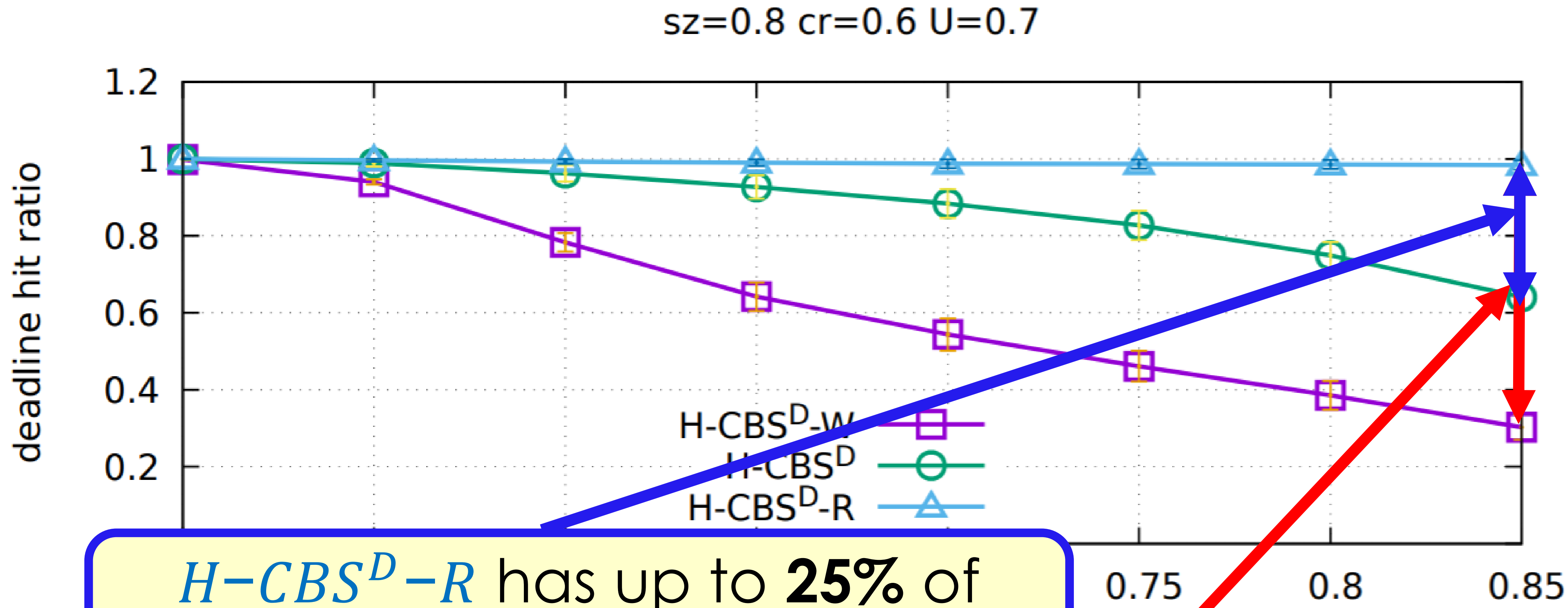


The higher the better

$H{-}CBS^D{-}R$ has up to **60%** of improvement over $H{-}CBS^D{-}W$

Relative server workload

Simulation Results



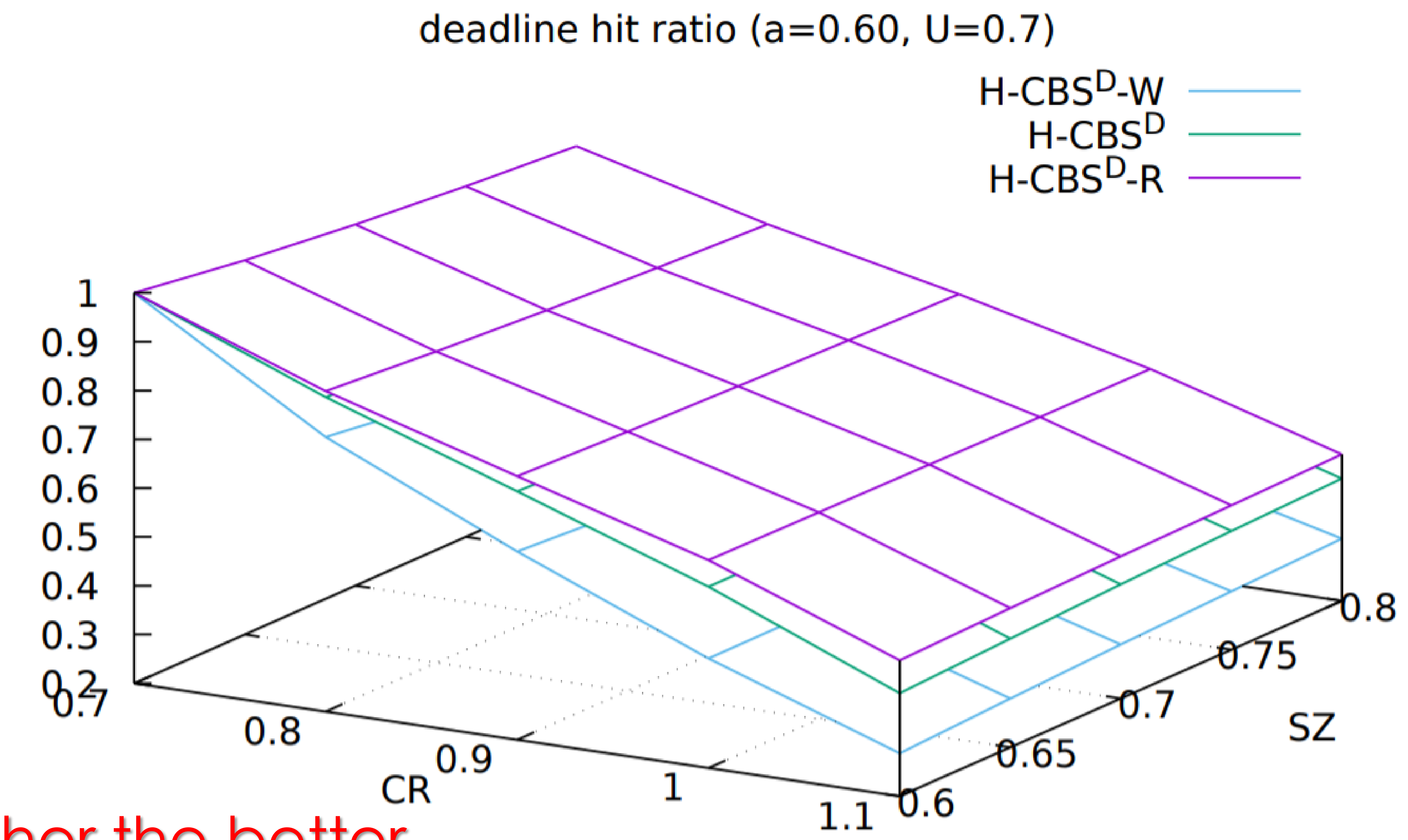
$H{-}CBS^D{-}R$ has up to **25%** of improvement over $H - CBS^D$

The higher the better

$H{-}CBS^D{-}R$ has up to **60%** of improvement over $H{-}CBS^D{-}W$

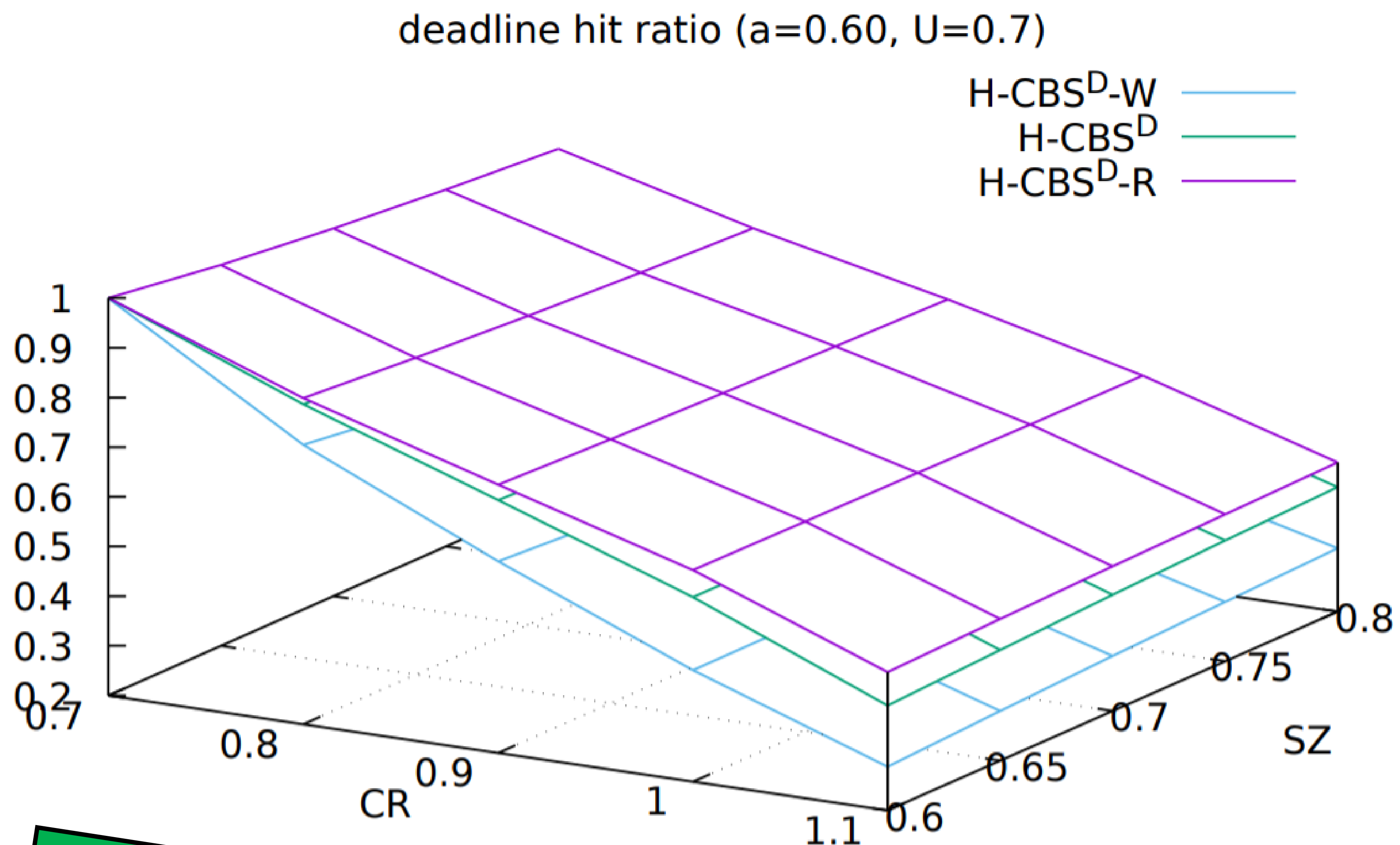
Relative server workload

Simulation Results



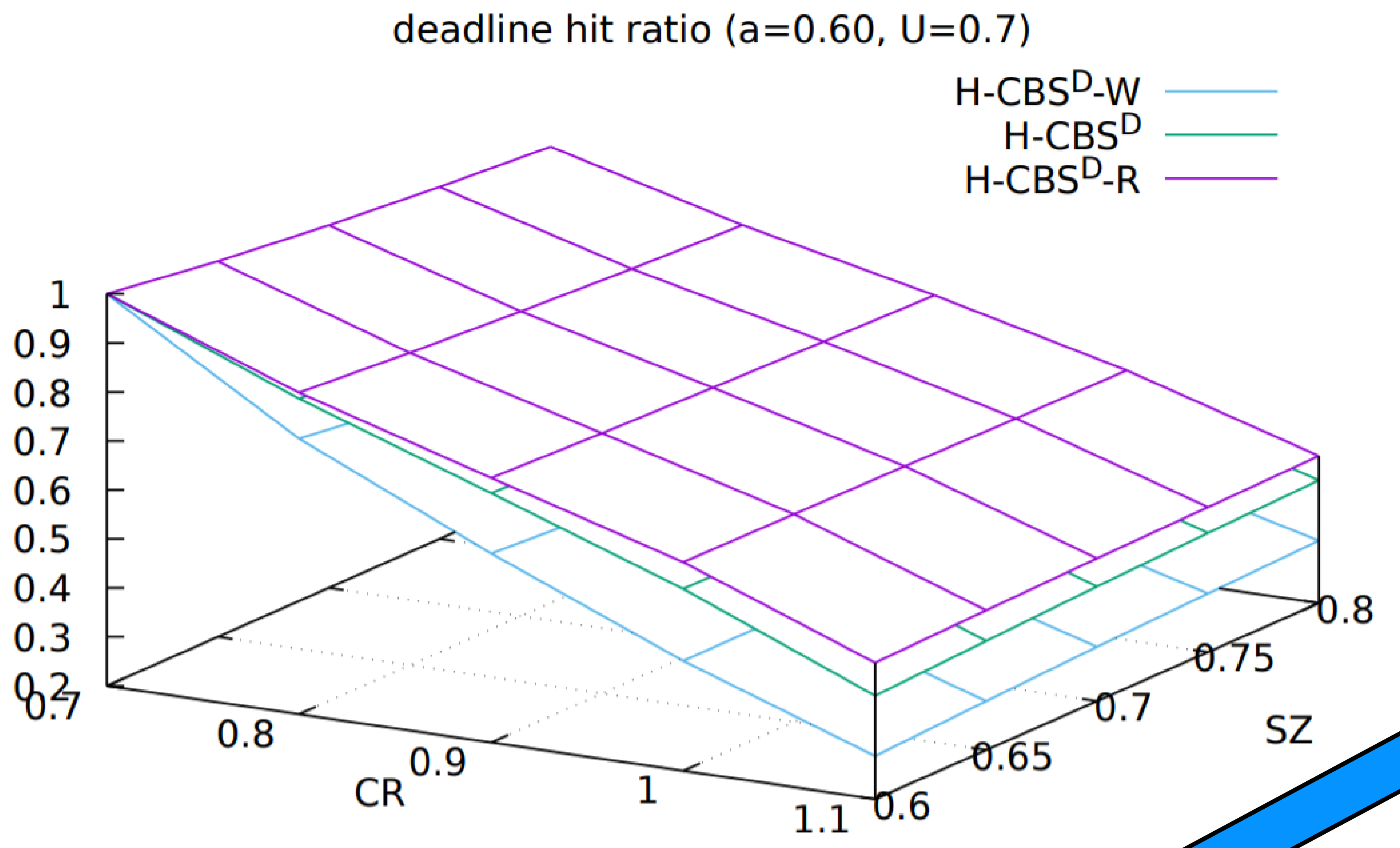
The higher the better

Simulation Results



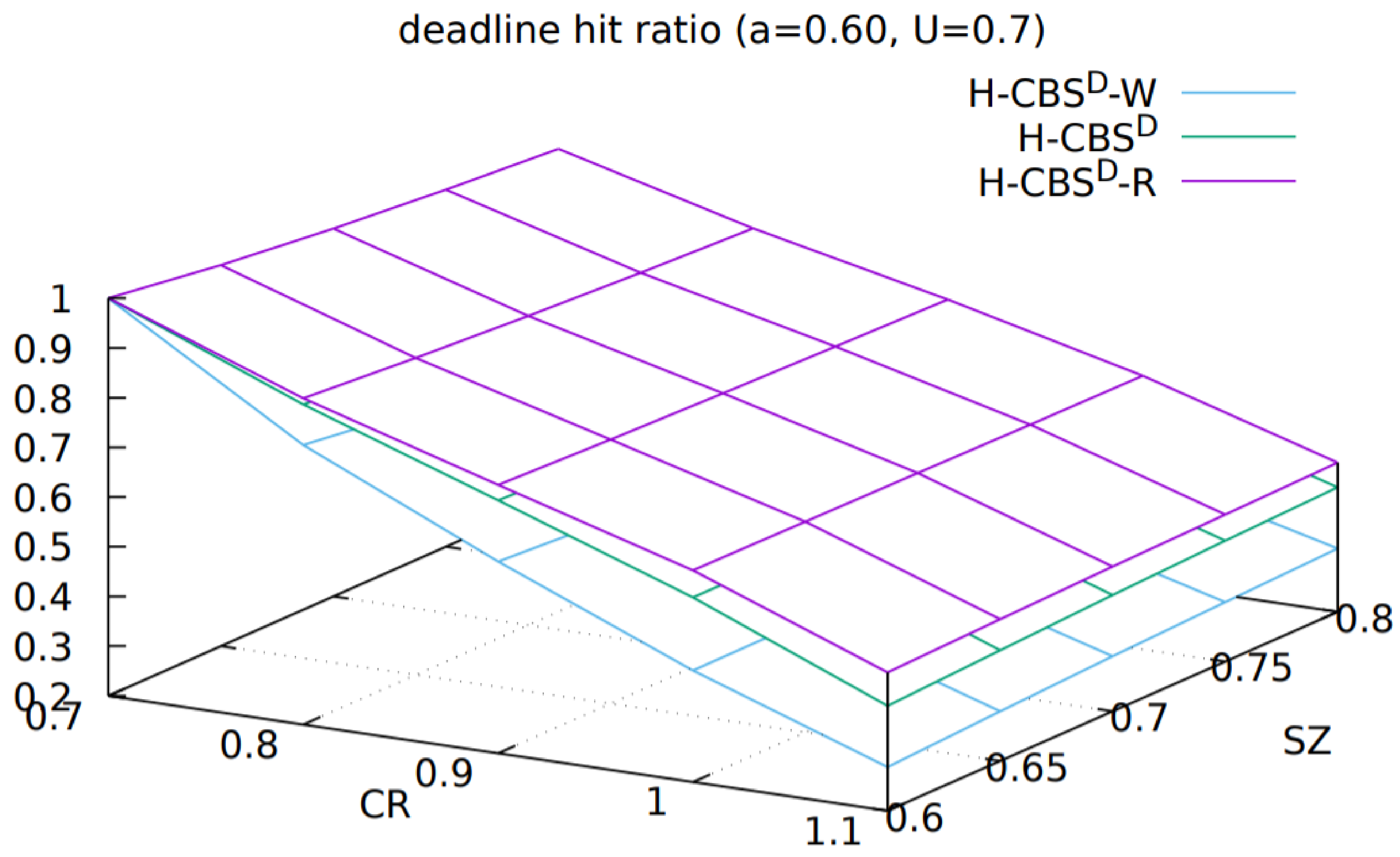
Task/Server
computation time ratio

Simulation Results



Variance of computation and inter-arrival times

Simulation Results



Similar trend

Comparison

$H\!-\!CBS^D\!-\!W$

- ✓ Guarantees a given bandwidth
- ✓ Bounded delay $\Delta = D + T - 2Q$
- ✓ Wake-up budget can be computed in constant time
- ✓ Independent from the schedulability test
- ✓ Compatible also with global scheduling
- ✗ Requires an additional queue for suspended servers
- ✗ Consumes budget also without actually executing

$H\!-\!CBS^D$ & $H\!-\!CBS^D\!-\!R$

- ✓ Guarantees a given bandwidth
- ✓ Bounded delay $\Delta = D + T - 2Q$
- ✓ Improves soft-real time metrics
- ✗ Wake-up budget can be computed in linear time
- ✗ Not compatible with global scheduling
- ✗ Penalties in terms of schedulability (sufficient test)

Conclusions

- We proposed three different reservations servers supporting constrained-deadlines
  - All of them allow to provide a given bandwidth and a bounded delay
- $H\text{-}CBS^D\text{-}W$ is independent from the schedulability test
  - Suitable for hard real-time systems
- $H\text{-}CBS^D$ and $H\text{-}CBS^D\text{-}\mathrm{R}$ leverages a specific schedulability test to improve soft real-time metrics
- The performance of the proposed algorithms has been experimented by means of simulations

Future Work

- Include a reclaiming mechanism in $H\text{-}CBS^{D}\text{-}W$ to improve average-case performance

- Extended our solutions to cope with shared resources

- Implementation in a real-time operating systems (e.g., Linux under SCHED_DEADLINE)

Thank you!

Daniel Casini
daniel.casini@sssup.it