Improving Soft Real-Time Performance Through Better Slack Reclaiming

Authors:

Caixue Lin and Scott A.Brandt

Presenter:

Muhammad Ali Awan

PhD Student, Cister-ISEP, Portugal.

OUTLINE

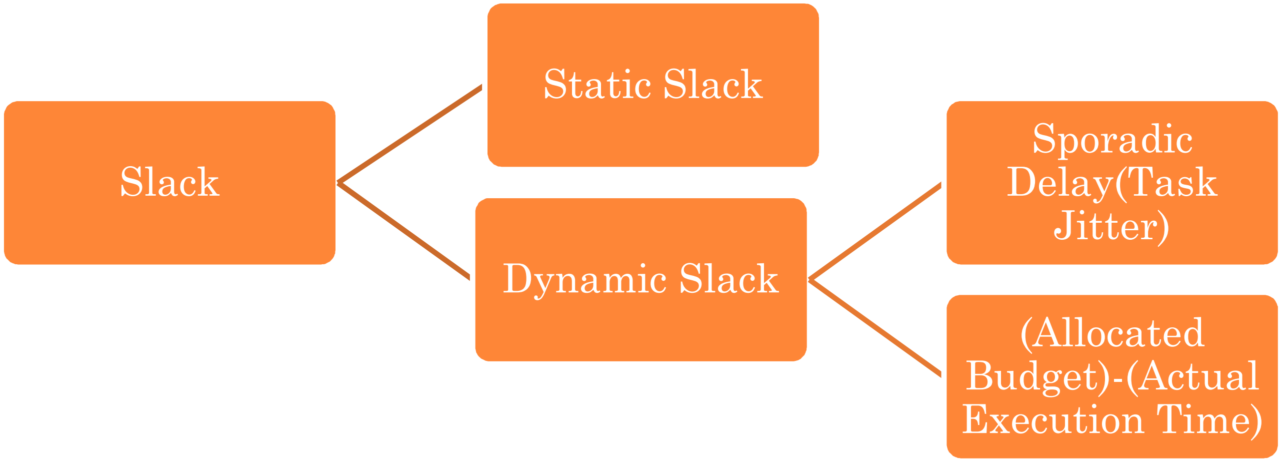
- Motivation
- Slack
- When to allocate slack
- Task selection to allocate slack
- Borrowing from the future
- Donating to the past
- Results
- Conclusion

Motivation

- Application with variety of timing constraints
  - Hard real time
  - Soft real time
  - Best Effort
- Performance Guarantee
  - Worst case resource reservation
  - Average case resource reservation
- Effective Distribution of slack

WHAT IS SLACK

- The execution time not used in system is called slack.



Slack

Static Slack

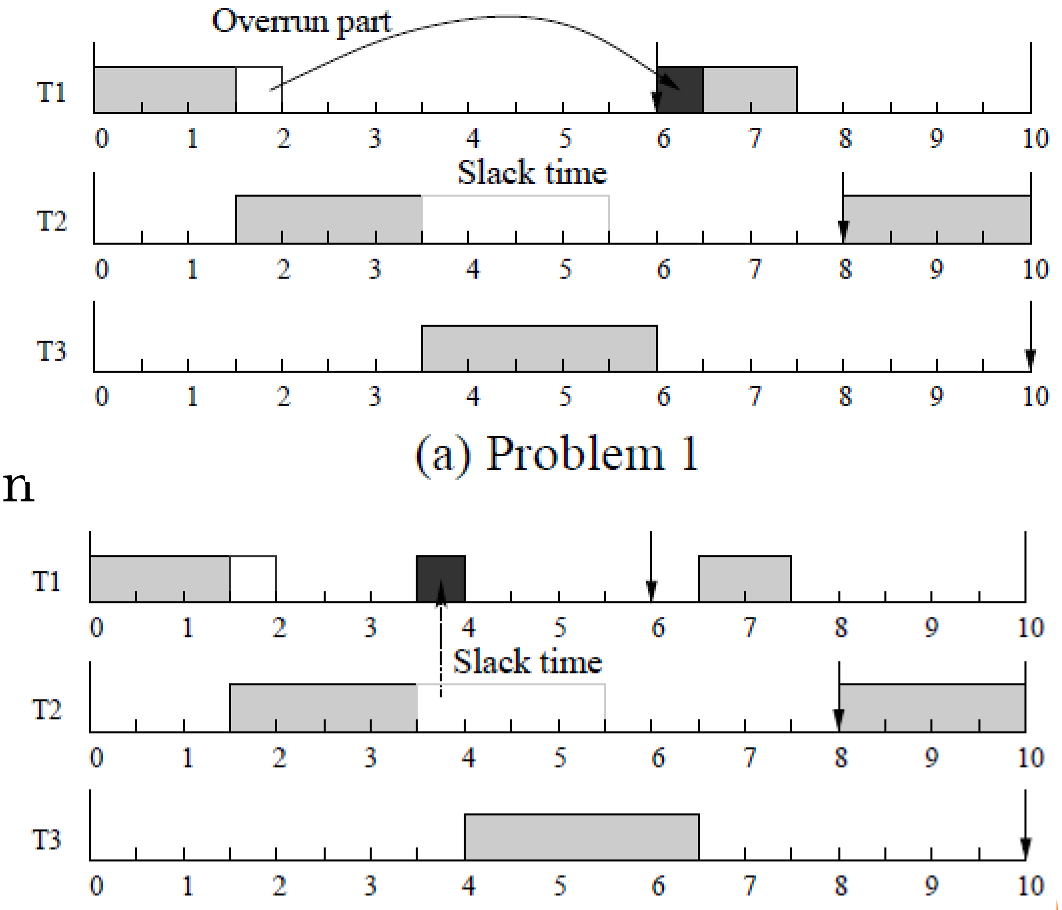
Dynamic Slack

Sporadic Delay(Task Jitter)

(Allocated Budget)-(Actual Execution Time)

WHEN TO ALLOCATE SLACK

- Allocate When Real Time(RT) task are idle
  - Isolates RT tasks
  - Delays slack use
- Task Set, T(E,T)
  - T_1(1.5,6), .5 unit overrun
  - T_2(4,8), needs 2 unit
  - T_3(2.5,10)



(a) Problem 1

(b) Solution 1

When to Allocate slack

Principle 1:

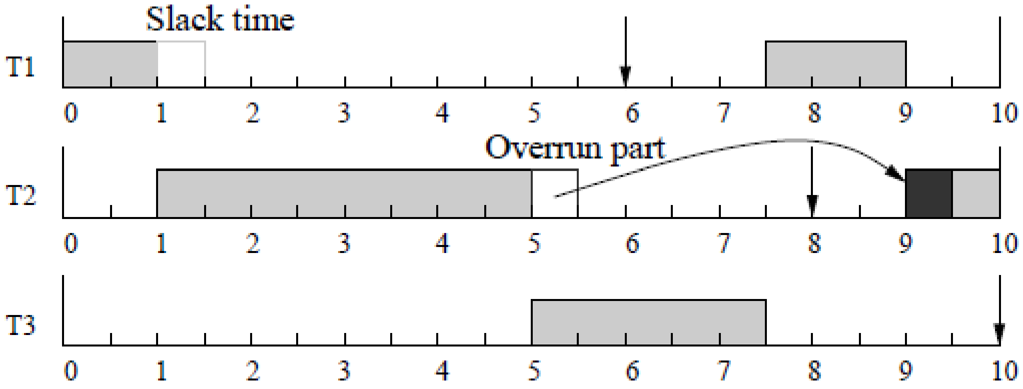
Allocate slack as early as possible, with the priority of the donating task.

- SRAND implements principle 1
  - On task completion, remaining budget is allocated to randomly selected task
  - Fewer deadline misses
  - Random selection is not optimal

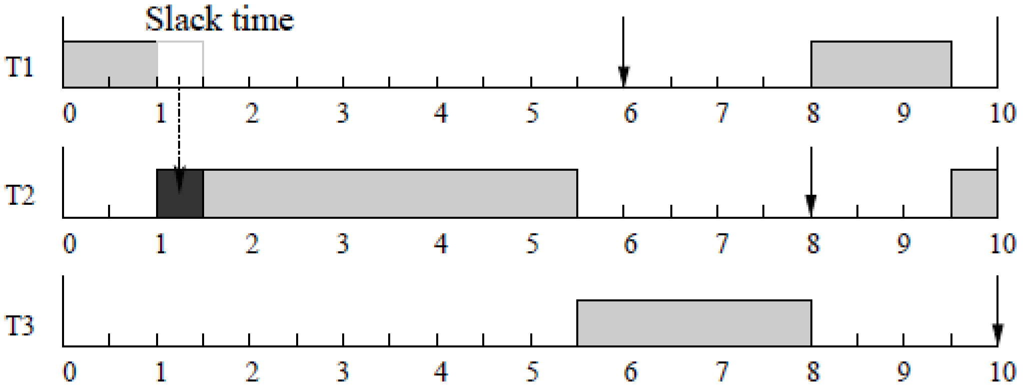
TASK SELECTION TO ALLOCATE SLACK

- Allocate to only overrun task.
- No overrun task at the time of slack generation

- Task Set
  - T1(1.5,6), needs 1 unit
  - T2(4,8), .5 unit overrun
  - T3(2.5,10)



(a) Problem 2



(b) Solution 2

TASK SELECTION TO ALLOCATE SLACK

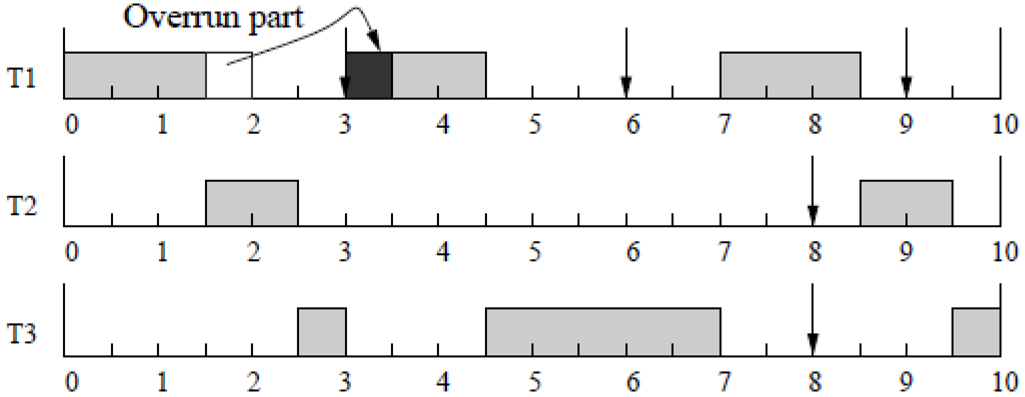
Principle 2:

Allocate slack to the task with the highest priority (earliest deadline(ED)).

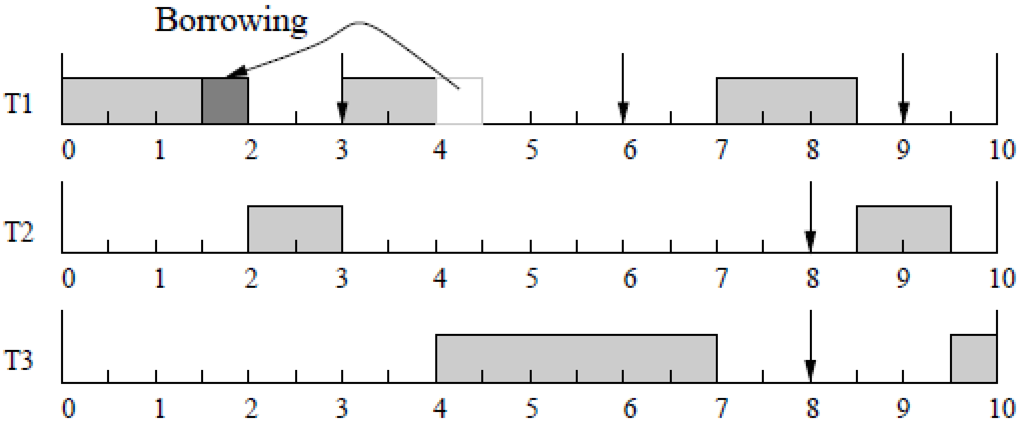
- SLAD Implements principle 1 and 2.
  - Make available as soon as possible(principle 1)
  - Give it pre-emptively to Earliest Deadline task(principle 2)
  - Task consumes slack first, before its reservation
  - Interrupting higher priority task consumes leftover slack
- Reasons
  - ED task is the Most critical task
  - Least likely to receive slack
  - Mostly likely to overrun
- SLAD outperforms SRAND and CBS

Borrowing from future

- CBS, RBED, IRIS and BEBS
  - allocate from future if overrun
  - Extend the deadline
- SLAD
  - Allows future borrowing (no slack available)
- Taskset
  - $T_1(1.5,3)$,
    - J_1 needs 2 units and J_2 needs 1 unit
  - $T_2(1,8)$,
  - $T_3(3,8)$



(a) Problem 3



(b) Solution 3

Borrowing from future

Principle 3

Allow tasks to borrow against their own future resource reservations (with the priority of the job from which the resources are borrowed) to complete their current job.

BORROWING FROM FUTURE

- SLASH implements principle 1, 2 and 3
  - Allows donation as soon as possible, to earliest deadline task (principle 1 and 2)
  - Allows borrowing from future job releases
    - Similar to (CBS, RBED, IRIS and BEBS)
  - In this way it serve the most critical jobs first
  - Assumes borrowed resources will turn out to be slack

- Issue with principle 3
  - Overrun task misses opportunity to get slack donation(priority lowered)

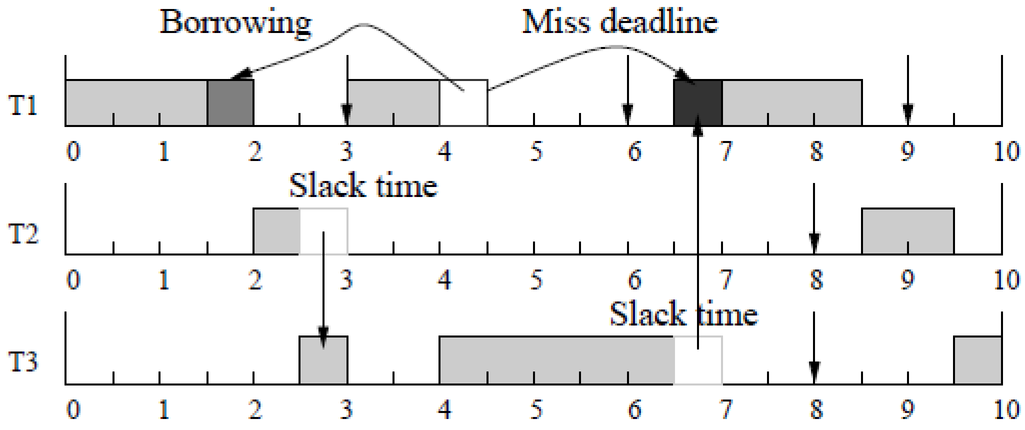
BORROWING FROM FUTURE

Revised Principle 2

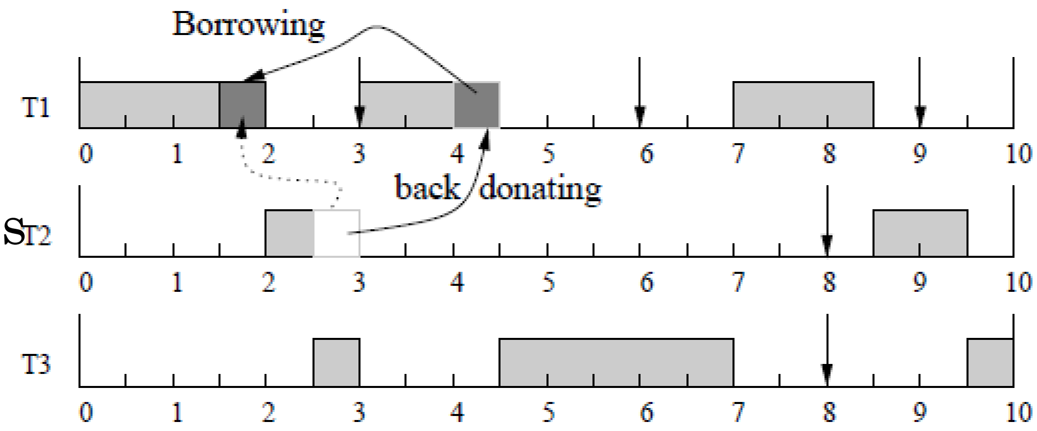
Allocate slack to the task with the highest priority (earliest original deadline)

DONATING TO THE PAST

- Issue
  - Finished job with borrowed budget from future
  - No longer in the ready queue
- Taskset
  - T_1(1.5,3),
    - J_1 and J_2needs 2 units
  - T_2(1,8), needs 0.5 units
  - T_3(3,8)



(a) Problem 4



(b) Solution 4

DONATING TO THE PAST

Principle 4:

Retroactively allocate slack to tasks that have borrowed from their current budget to complete a previous job.

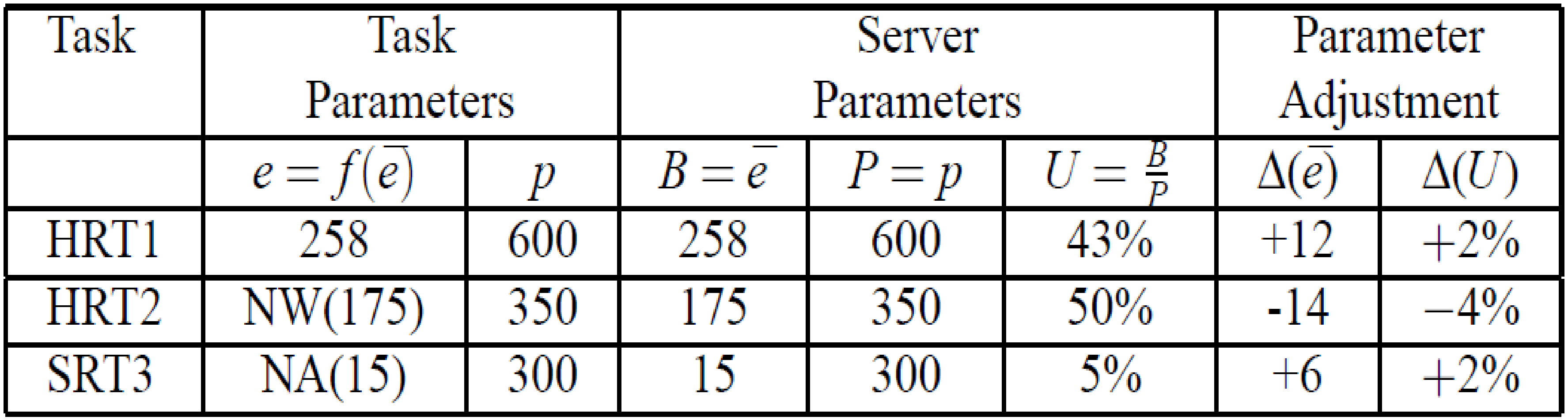
- BACKSLASH Implements principle 1,2,3 and 4
  - Similar to HistroyReWriting paper for fixed priorities(Static Rate monotonic)
  - Task that previously consumed slack are eligible to receive future slack donations
  - Need to store
    - information of the completed jobs that borrowed
    - Depleting jobs
  - Outperforms over all other algorithms (SRAND, SLAD and SLASH)

RESULTS

- Metrics
  - Deadline Miss Ratio
    - (deadline misses /Number of jobs)
  - Tardiness
    - (Total accumulated lateness/Total length of All Periods)
- Fixed task sets
- Random Task set

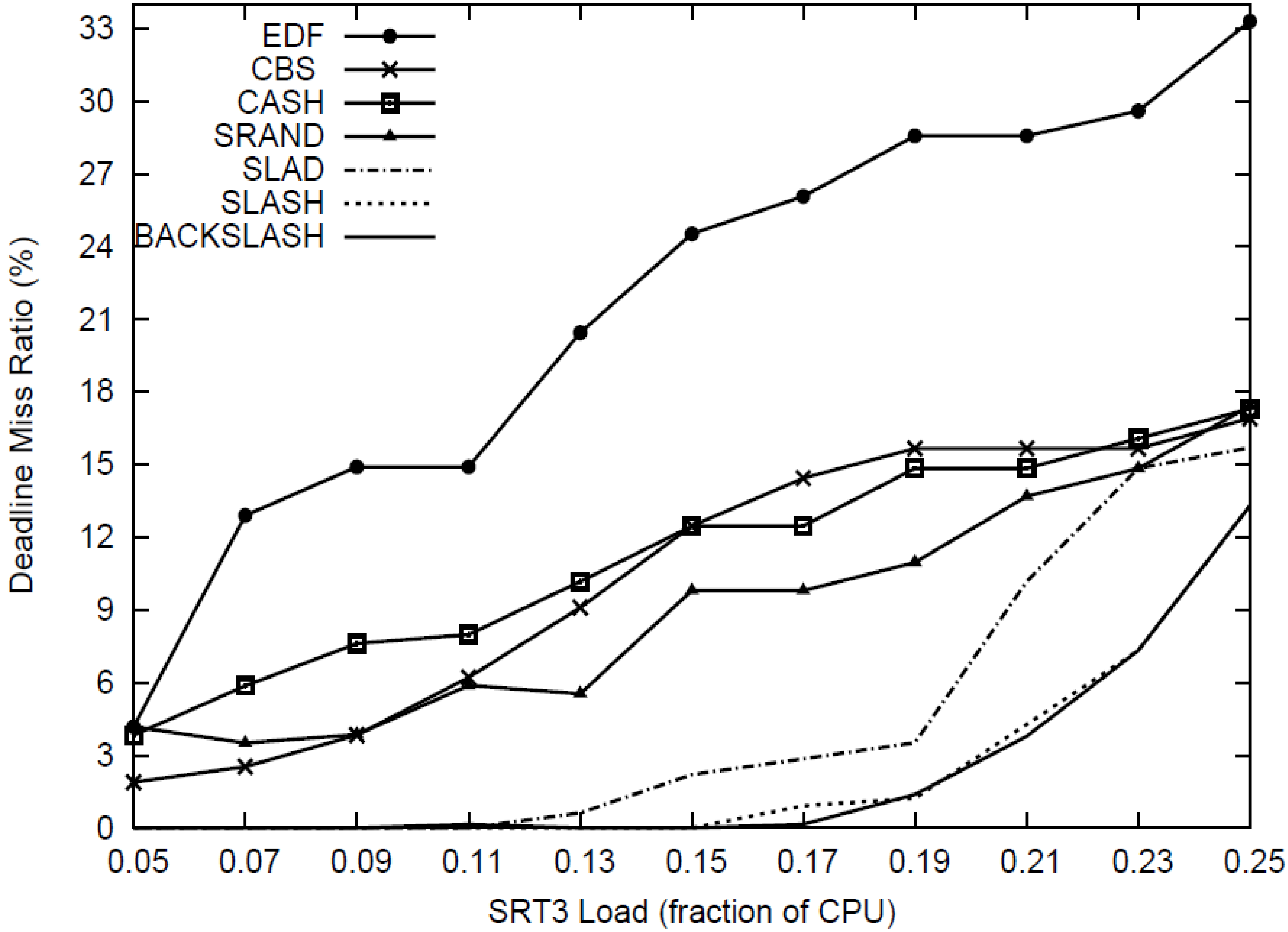
FIXED TASK SET (PERFORMANCE AS FUNCTION OF SYSTEM LOAD)

Table 1. Workload 1

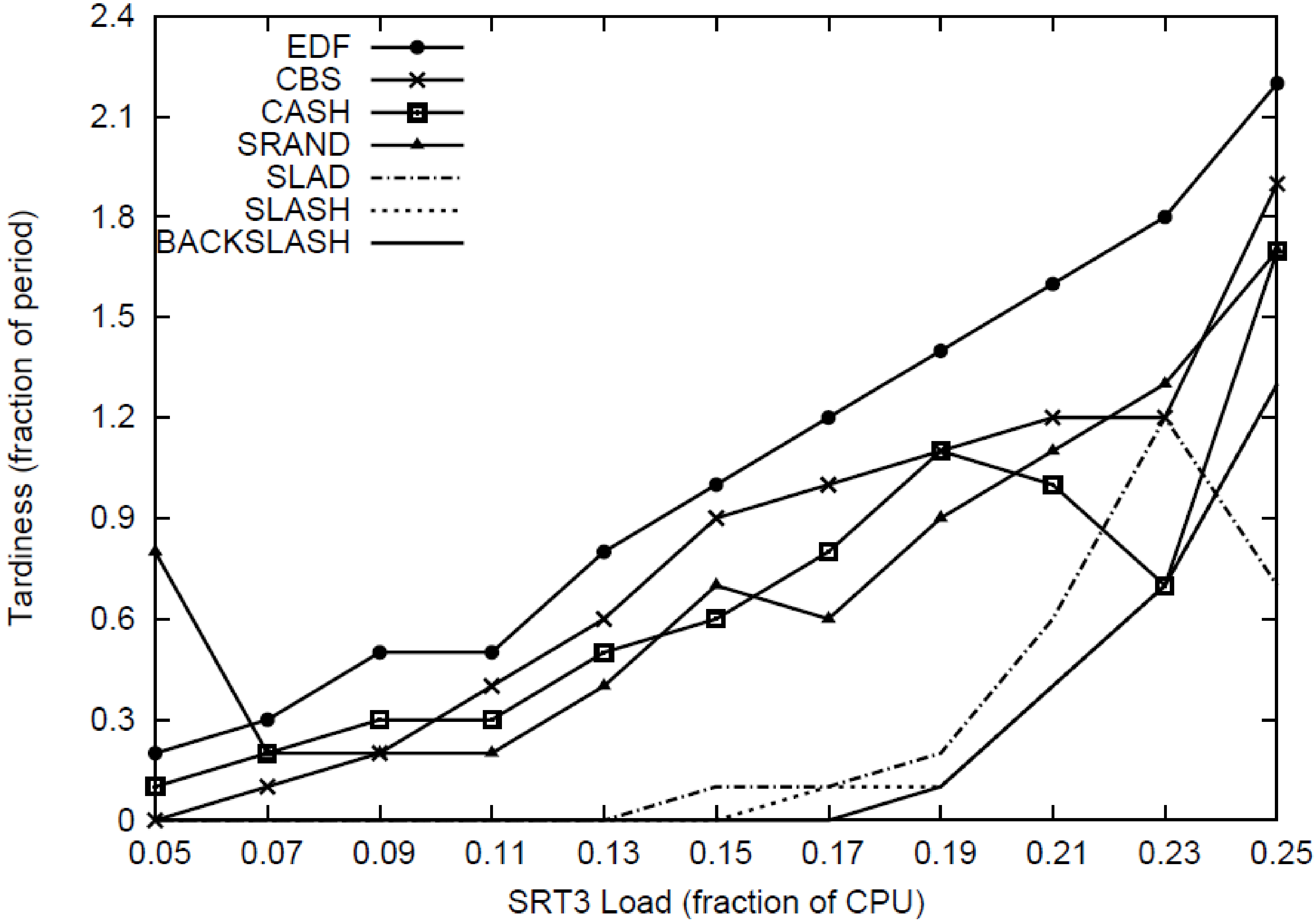


Task	Task Parameters		Server Parameters			Parameter Adjustment	
	$e = f(\overline{e})$	p	$B = \overline{e}$	$P = p$	$U = \frac{B}{P}$	$\Delta(\overline{e})$	$\Delta(U)$
HRT1	258	600	258	600	43%	+12	+2%
HRT2	NW(175)	350	175	350	50%	-14	−4%
SRT3	NA(15)	300	15	300	5%	+6	+2%

DEADLINE MISS RATIO AS FUNCTION OF LOAD



TARDINESS AS FUNCTION OF LOAD



Fixed Task set (Performance as a function of Period)

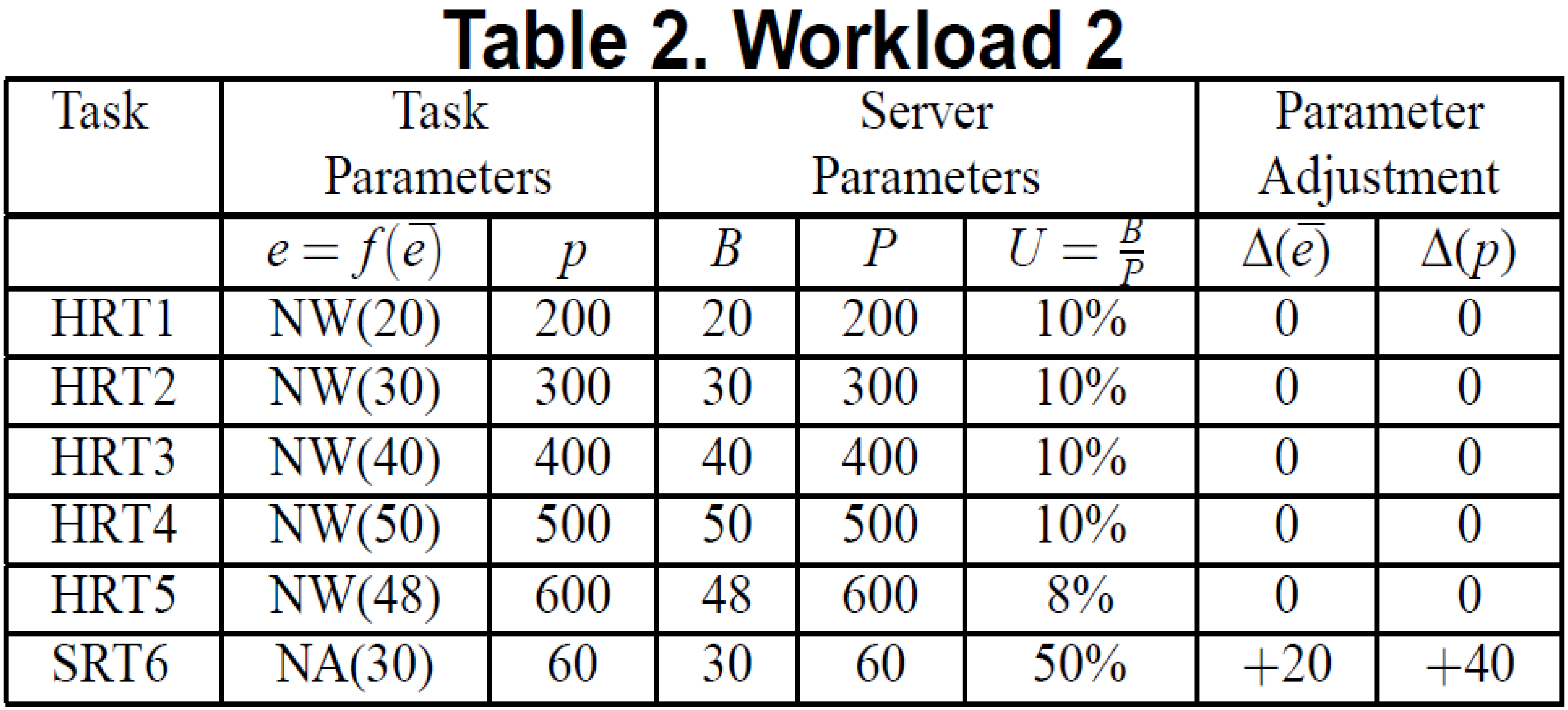
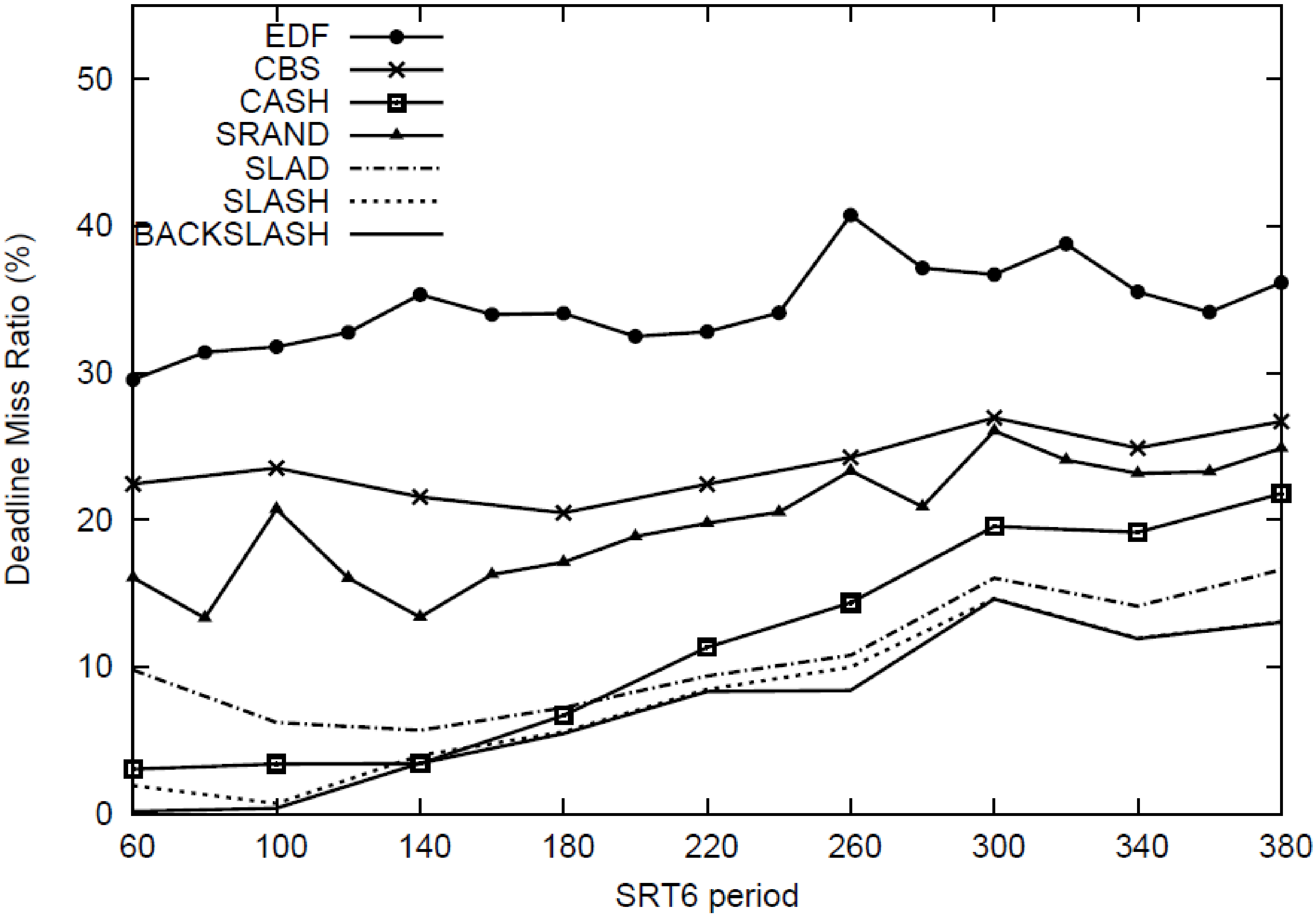


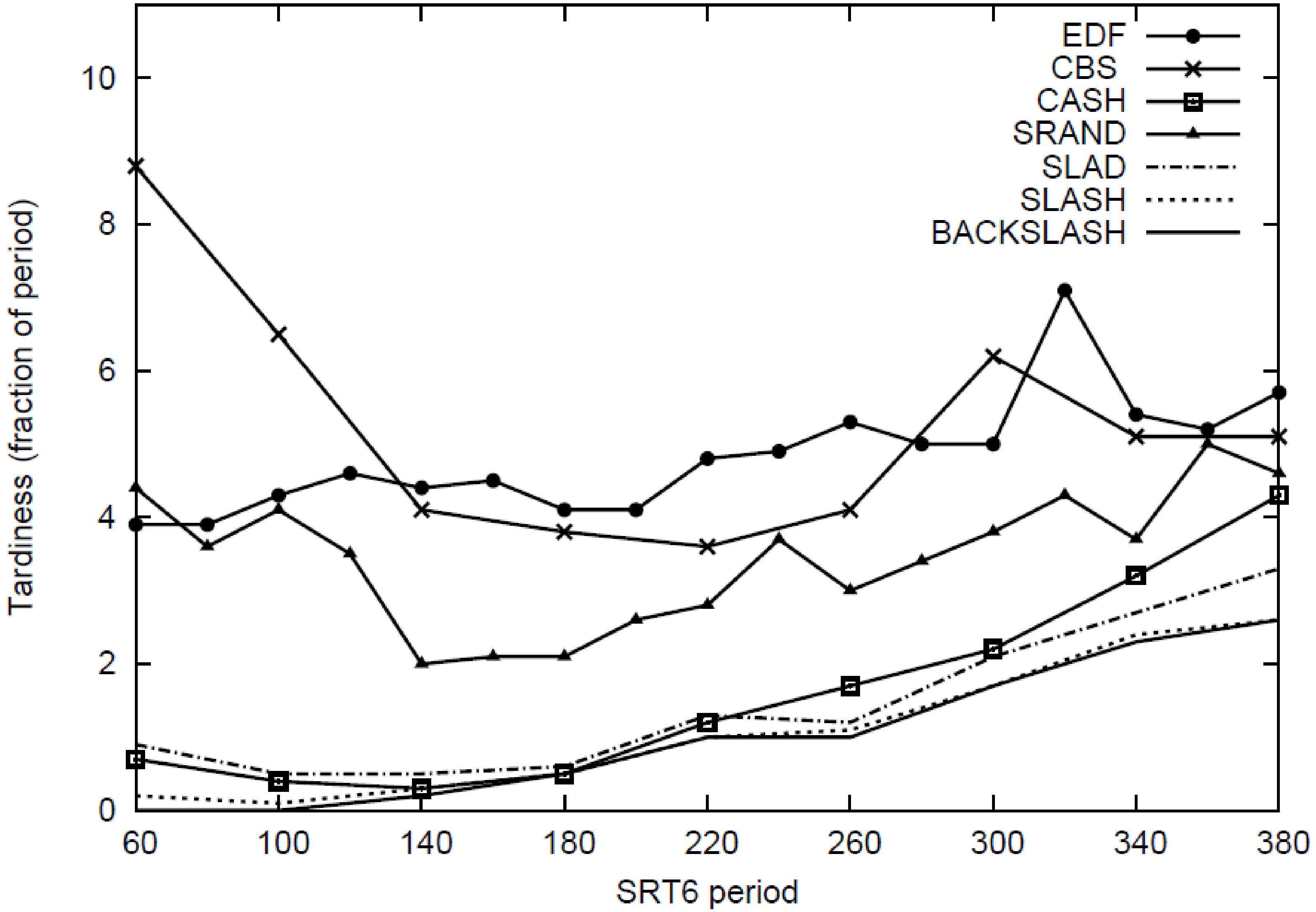
Table 2. Workload 2

Task	Task Parameters		Server Parameters			Parameter Adjustment	
	$e = f(\overline{e})$	p	B	P	$U = \frac{B}{P}$	$\Delta(\overline{e})$	$\Delta(p)$
HRT1	NW(20)	200	20	200	10%	0	0
HRT2	NW(30)	300	30	300	10%	0	0
HRT3	NW(40)	400	40	400	10%	0	0
HRT4	NW(50)	500	50	500	10%	0	0
HRT5	NW(48)	600	48	600	8%	0	0
SRT6	NA(30)	60	30	60	50%	+20	+40

DEADLINE MISS RATIO AS A FUNCTION OF PERIOD



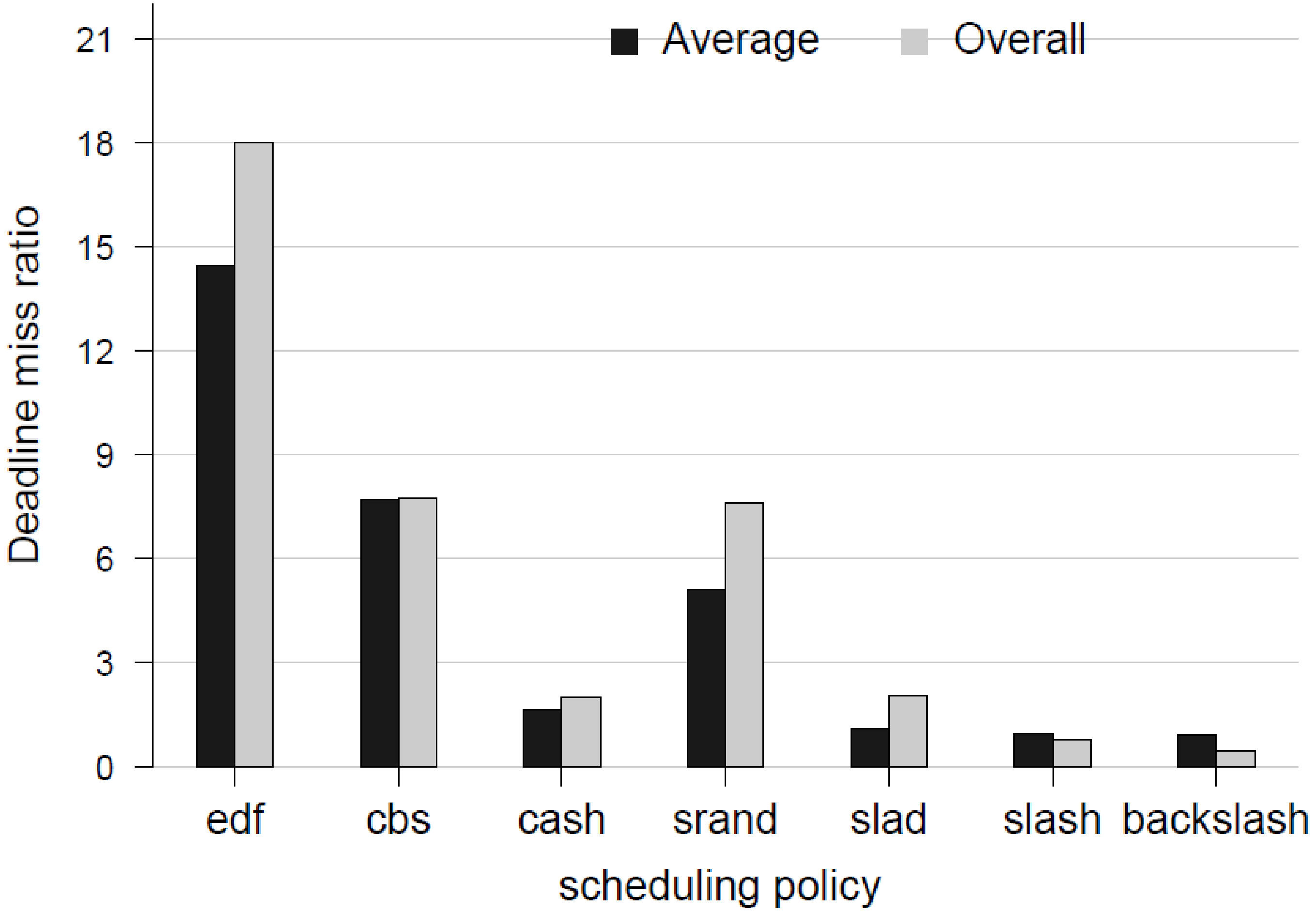
Tardiness as a function of period



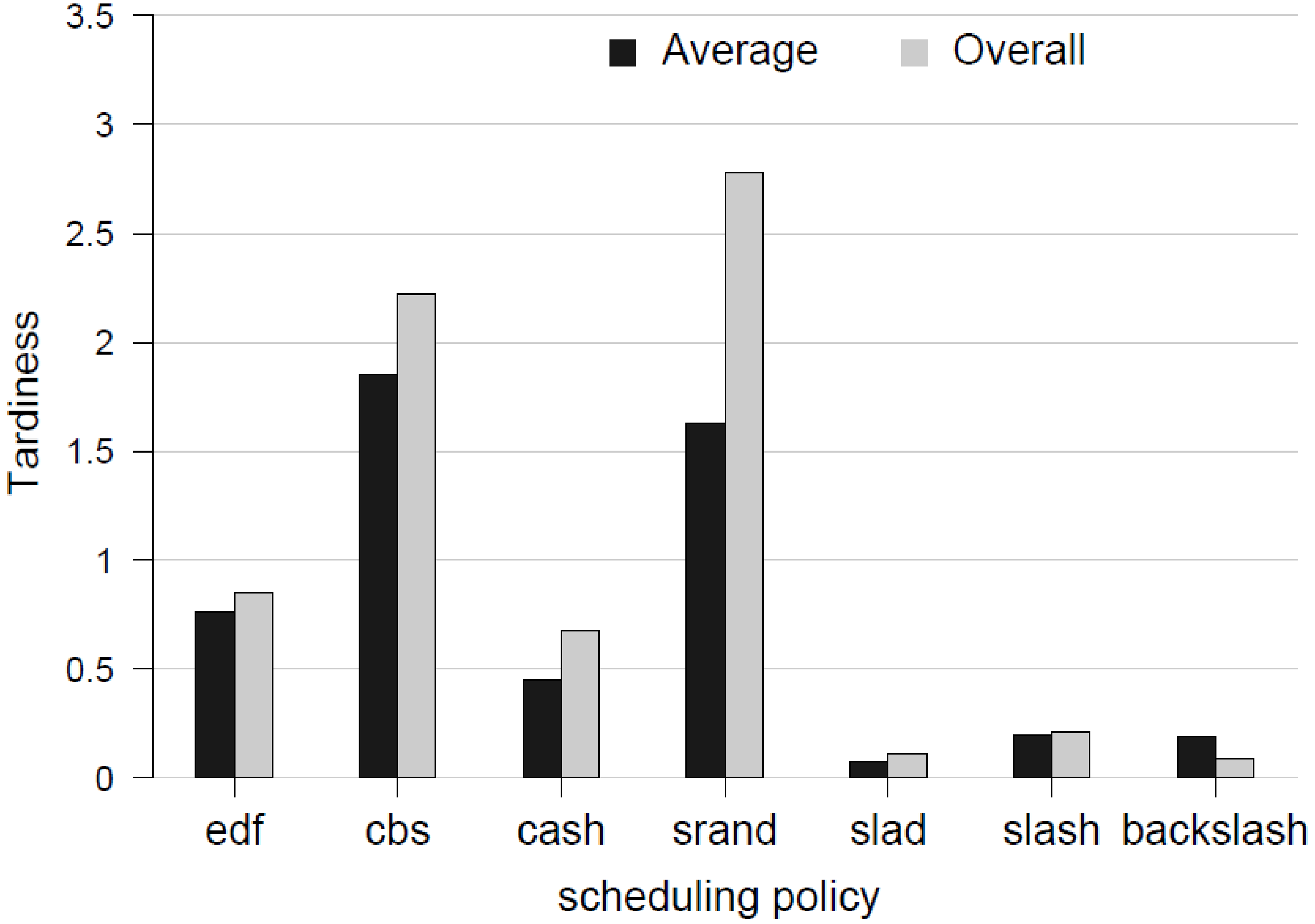
Random Task set

- Variation
  - Number of Hard real time task
  - Number of soft real time task
  - Task model (periodic, aperiodic)
  - Task Parameters (1ms to 1000ms)
    - Periods
    - Execution time

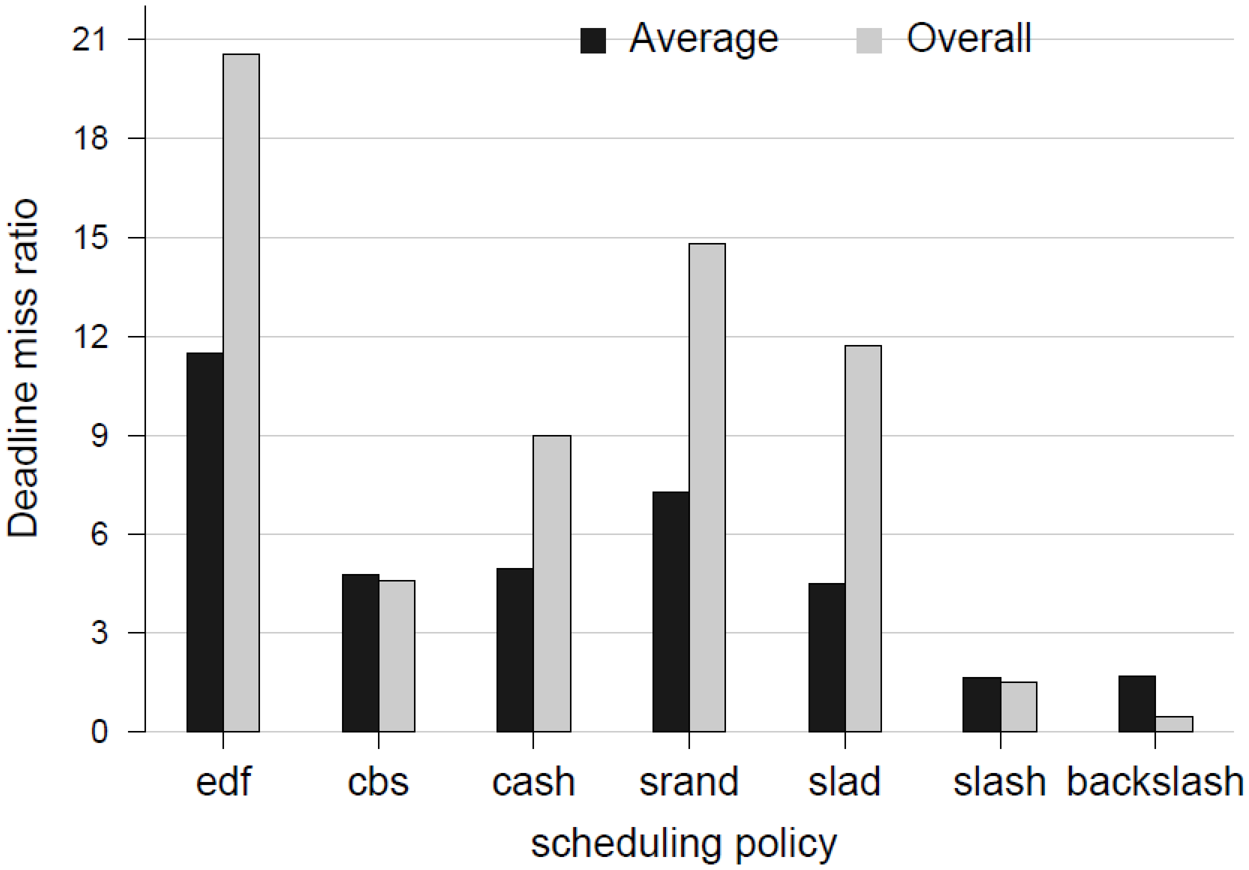
- Selected random workload
  - 12 task sets
    - Each with 8 periodic/aperiodic (random distribution among soft and hard RT tasks)



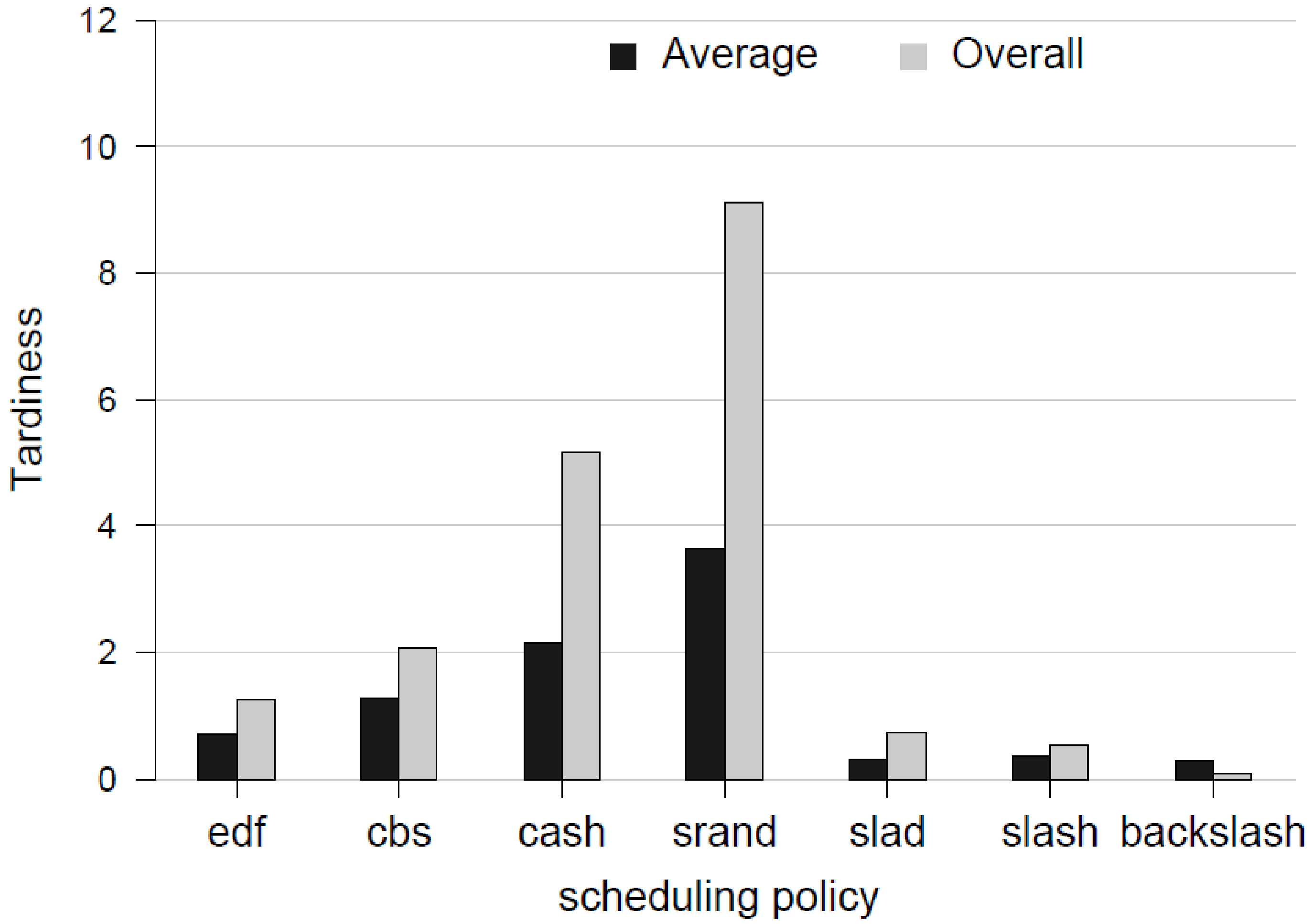
(a) Deadline miss ratio: periodic task sets



(b) Tardiness: periodic task sets



(c) Deadline miss ratio: aperiodic task sets



(d) Tardiness: aperiodic task sets

CONCLUSION

- 4 principle
  - As early as possible
  - Allocate to earliest deadline first
  - Borrow from future
  - Retroactively allocate slack
- Implemented the principles in four algorithms
  - SRAND, SLAD, SLASH and BACKSLASH
- BACKSLASH outperform all other algorithms, including CBS, CASH, RBED and IRIS

REFERENCES

- All material and figures are taken from the original paper(Improving Soft Real-Time Performance Through Better Slack Reclaiming).
- Slack slide is extracted from the SMARTS(Slack MAnagement for hierarchical Real-Time Systems) project proposal.

QUESTIONS