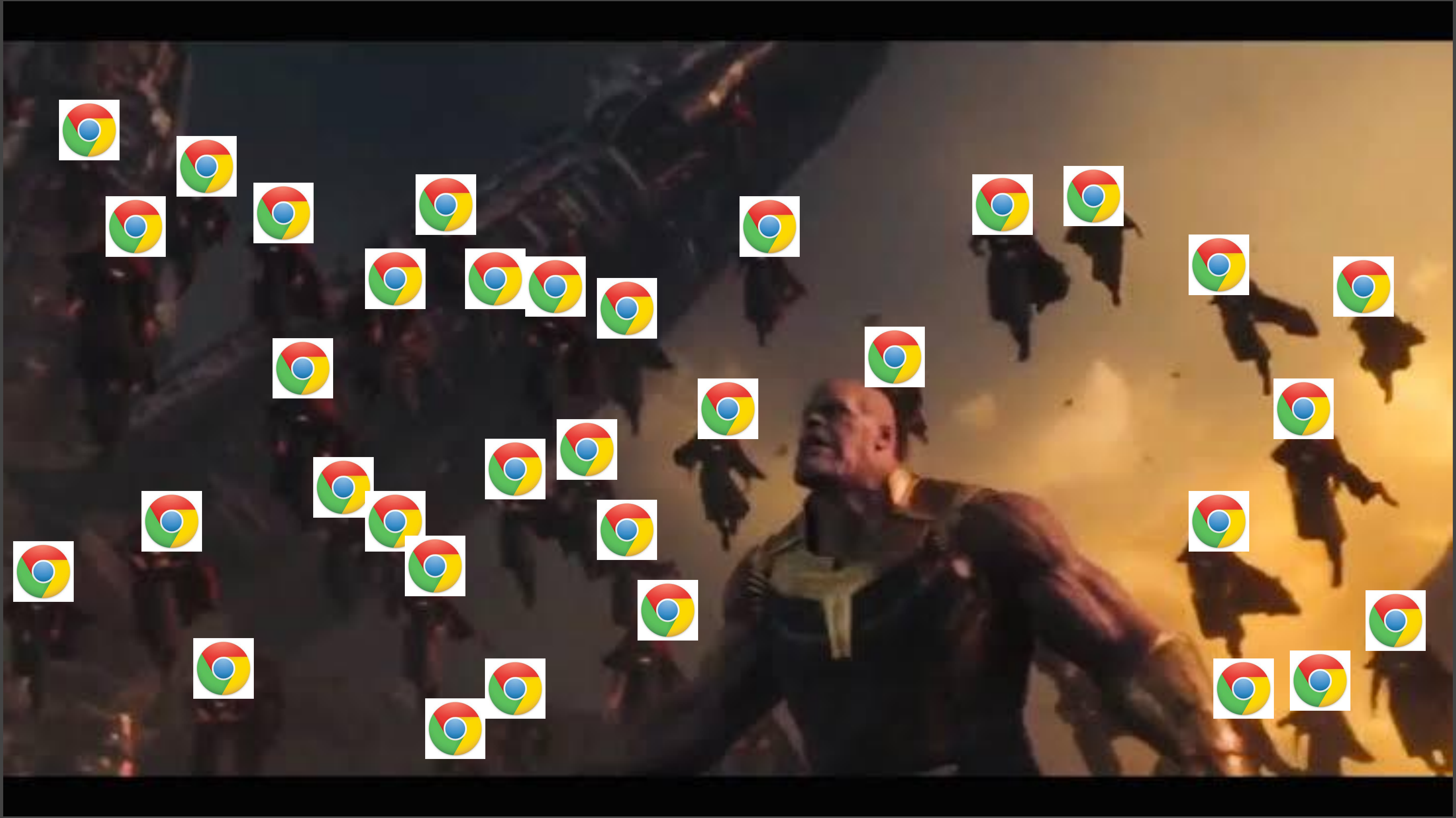
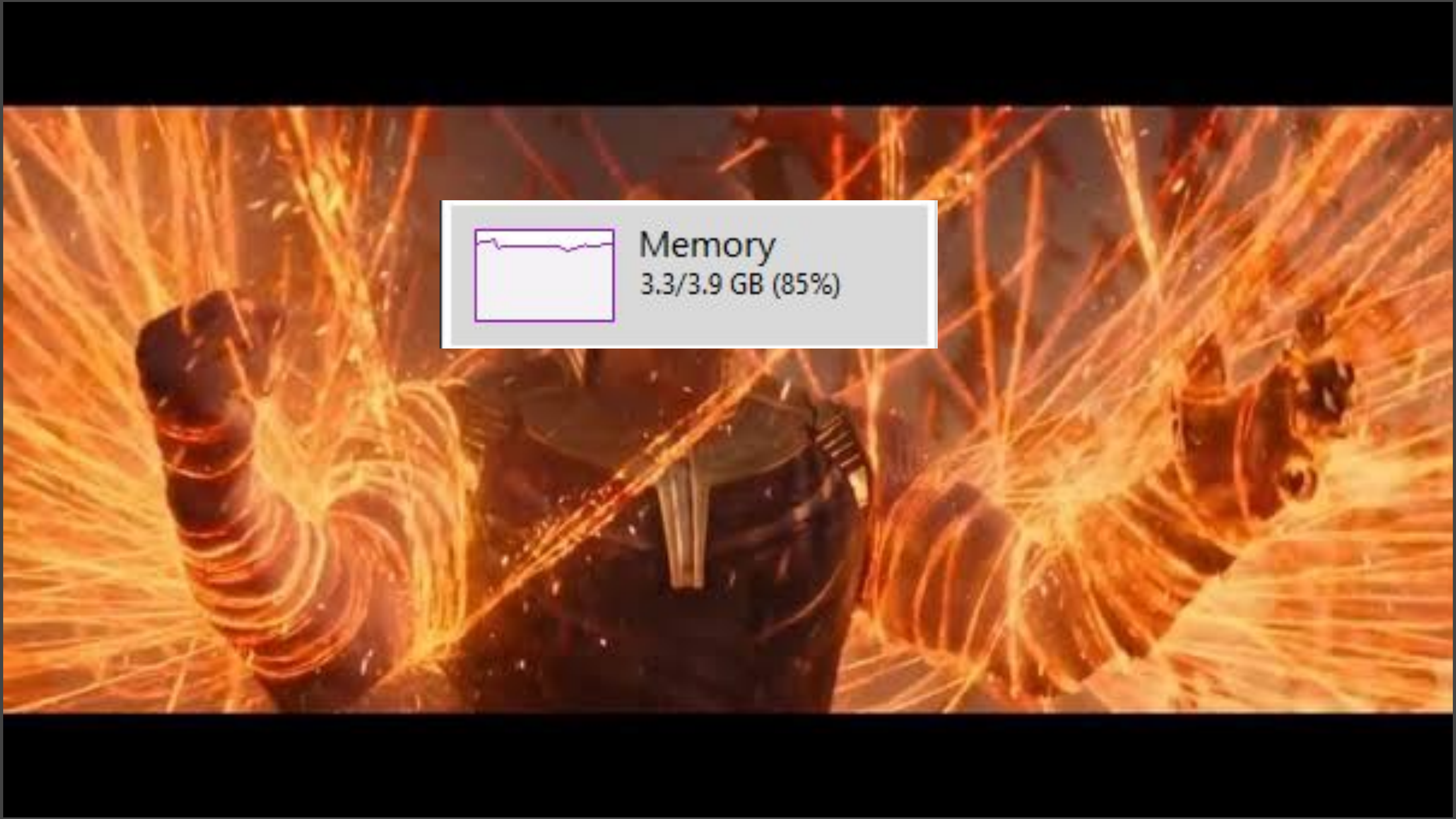

MY
COMPUTER




Memory
3.3/3.9 GB (85%)

End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis

Presented by Renz Rallion T. Gomez,
Christopher M. Bermudez,
Vily Kaylle G. Cachero &
Eugene G. Rabang

Abstract

The End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis functions as a method of queuing tasks that the CPU will process. It is an improved Round Robin that uses of Shortest Job First to compare tasks and the end to end method to execute tasks. It aims to reduce these three metrics: *(1)* the time it takes to complete tasks, *(2)* the time it takes for the ready-for-processing tasks to be executed, and *(3)* the number of times the CPU switches between tasks.

Abstract

To verify these aims, test cases were conducted that showed the comparative results between E-EDRR and the original algorithms, (Round Robin and Shortest Job First). In the findings, it satisfied the expectations by getting lower scores than both of the original algorithms from all the metrics in all test cases except in one where the Round Robin's variables favor the conditions of the case. It was concluded that E-EDRR has achieved its goal and proved its theoretical acquisitions.

Introduction

The function of Scheduling Algorithms in Operating Systems is to provide an established method of queueing tasks/instructions for the Central Processing Unit (CPU) to process. Some basic scheduling algorithms:

Scheduling Algorithms are First Come First Serve (FCFS), Shortest-Job-First (SJF), Priority Scheduling, Round Robin (RR), Multilevel Queue Scheduling [1]. These algorithms are globally used in a wide variety of ways.

End to End Dynamic Round Robin

The End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest-Job-First Analysis aims a better time interval in producing results. It is an improved Round Robin scheduling that is redesigned by using the Shortest Job First analysis to queue tasks. The algorithm identifies the tasks by shortest to longest burst time.

Review of Related Literature

Most of the improved CPU scheduling algorithms are focused on the improvement of the implementation of the available resources. These resources are always of the consideration of the user and is capable of determining the criteria in measuring the various algorithms' performances. These criteria include:

Turnaround Time: The time required to complete a process (wall clock time).

Waiting Time: The time that a process spends in the queue before being executed.

Context Switch: The process of switching tasks/thread, given that the current process is saved so it can be continued later on

Review of Related Literature

These are the some most popular CPU scheduling Algorithms existing:

1. The First Come First Serve (FCFS) or also known as First In First Out
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Best Job First (BJF)

Pseudo Code vs Java Code

Pseudo Code

Let **TQ** be the ***time quantum***.

Let **NA** be the ***newly arrived processes***.

Let **Q1** be the ***ready queue***

Java Code

```
int tq = 0;

int[] NA = { };

int[] q1 = { };
```

Pseudo Code

1. if(NA == true) {

    enqueue NA to Q1,

    repeat step 1

   } else

    proceed to step 2;

Java Code

```
static int[] getTasks() {
    length = s.nextInt();
    int[] a = new int[length];
    for (int i = 0; i < length; i++) {
   [i] = s.nextInt(); }
    return a; }
```

Pseudo Code

```
2. if(Q1 != empty) {
        sort tasks according to BT,
        proceed to step 3
  } else
      proceed to step 1;
```

Java Code

```
static int[] sortArray(int[] NA) {
        int sorting = 0;
        while (sorting < NA.length) {
            if (sorting > 0) {
                quickSort(NA, 0, sorting);
                turnaroundtime++;}
            sorting++;}
        return NA;}
```

Pseudo Code

3. Determine the TQ

by using [*equation* 2]

equation 2: **TQ** = ***current shortest task's burst time***

Java Code

```
static void executeTasks(int[] q1) {
    ...
    tq = q1[shortest];
    ...
}
```

Pseudo Code

5. if(longest task != complete) {

Longest task's progress

is saved and its burst time

Is reduced by TQ

} else

proceed to step 6;

Java Code

```
static void executeTasks(int[] q1) {
    while (shortest <= longest) {
    ...
     lt = q1[longest];
     bt = lt - tq;
     q1[longest] = bt;
    ...
}
```

Pseudo Code

4. if(Q1.length != 1) {

    execute shortest task,

    execute longest task

} else

    execute shortest task;

Java Code

```
static void executeTasks(int[] q1) {
int shortest = 0, longest = length-1;
while (shortest <= longest) {
    ...
     q1[shortest] = 0;
    shortest++;
    ...
     q1[longest] = bt; ...
     if (q1[longest] == 0) longest--;
    ... } }
```

Pseudo Code

6. Dequeue completed tasks from Q1 and proceed to step 1

Java Code

```
static void main(String[] args) {
    int[] NA;
    ...
    NA = getTasks();
    ...
    int[] a = sortArray(NA);
    executeTasks(a); }
```

Test Cases

Test Cases

All test cases were performed with the consideration of the following assumptions:

1. Processes are executed in a single processor.

2. Processes are CPU bound.

3. Number of processes and BTs are initially known.

4. SJF and RR are used as benchmarking algorithms.

5. RR will have a TQ of 25 in respect to the test cases’ average BT.

Test Cases

Test Case 1: We assumed five (5) processes wherein they have equal BTs (as shown in Table 2 below)

Table 3 shows the comparative results of E-EDRR against the benchmarking algorithms.

Task	Burst Time
T0	25
T1	25
T2	25
T3	25
T4	35

[Table 2: Test Case 1]

Algorithm	TTAT	ATAT	AWT	CS
E-EDRR	137	27.4	20	5
SJF	149	29.8	50	5
RR	125	25	50	5

[Table 3 Test Case 1 result]

Test Cases

Test Case 2: We assumed five (5) processes wherein their BTs in increasing order (as shown in Table 4 below).

Table 5 shows the comparative results of E-EDRR against the benchmarking algorithms.

Task	Burst Time
T0	19
T1	22
T2	25
T3	28
T4	31

[Table 4: Test Case 2]

Algorithm	TTAT	ATAT	AWT	CS
E-EDRR	145	29	30.6	5
SJF	149	29.8	44	5
RR	175	35	90	7

[Table 5: Test Cae 2 result]

Test Cases

Here's the Graph of the Results:

[Figure 1: Test Case 1 Results Chart]

[Figure 2: Test Case 2 Results Chart]

Test Cases

Test Case 3: We assumed five (5) processes wherein their BTs in decreasing order (as shown in Table 6).

Table 7 shows the comparative results of E-EDRR against the benchmarking algorithms.

Task	Burst Time
T0	31
T1	28
T2	25
T3	22
T4	19

[Table 6: Test Case 3]

Algorithm	TTAT	ATAT	AWT	CS
E-EDRR	145	29	30.6	5
SJF	149	29.8	44	5
RR	175	35	90	7

[Table 7: Test Case 3 result]

Test Cases

Test Case 4: We assumed five (5) processes wherein their BTs in random order (as shown in Table 8).

Table 9 shows the comparative results of E-EDRR against the benchmarking algorithms.

Task	Burst Time
T0	27
T1	21
T2	29
T3	34
T4	24

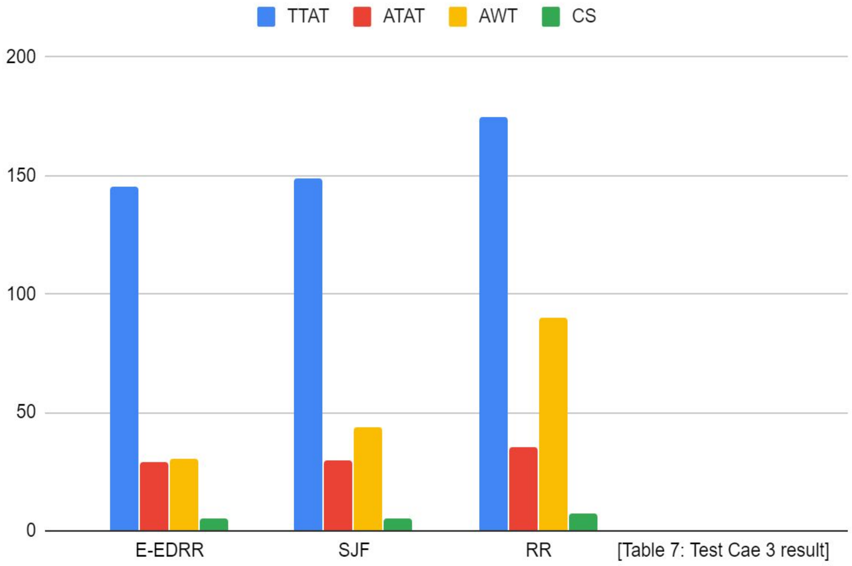
[Table 8: Test Case 4]

Algorithm	TTAT	ATAT	AWT	CS
E-EDRR	155	31	33.2	5
SJF	159	29.8	47.8	5
RR	200	40	100	8

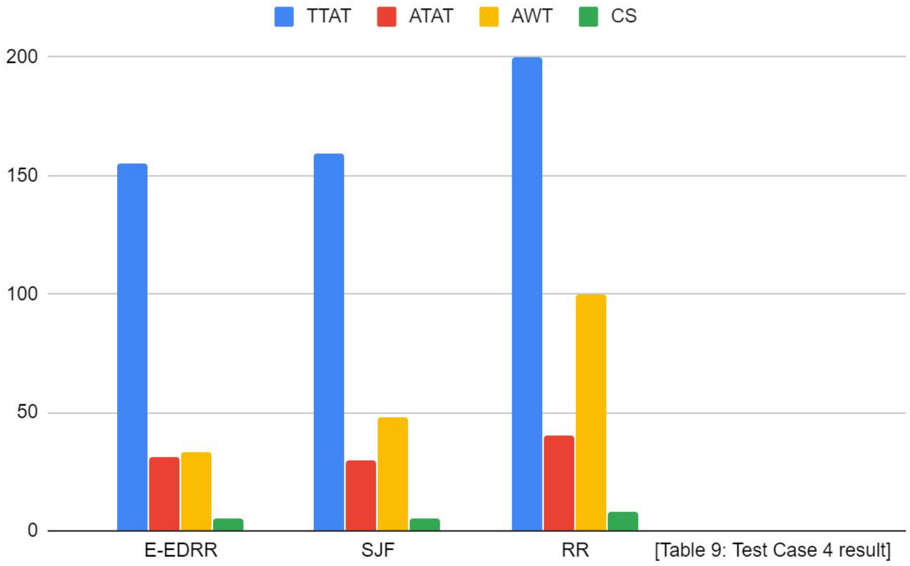
[Table 9: Test Case 4 result]

Test Cases

Here's the Graph of the Results:



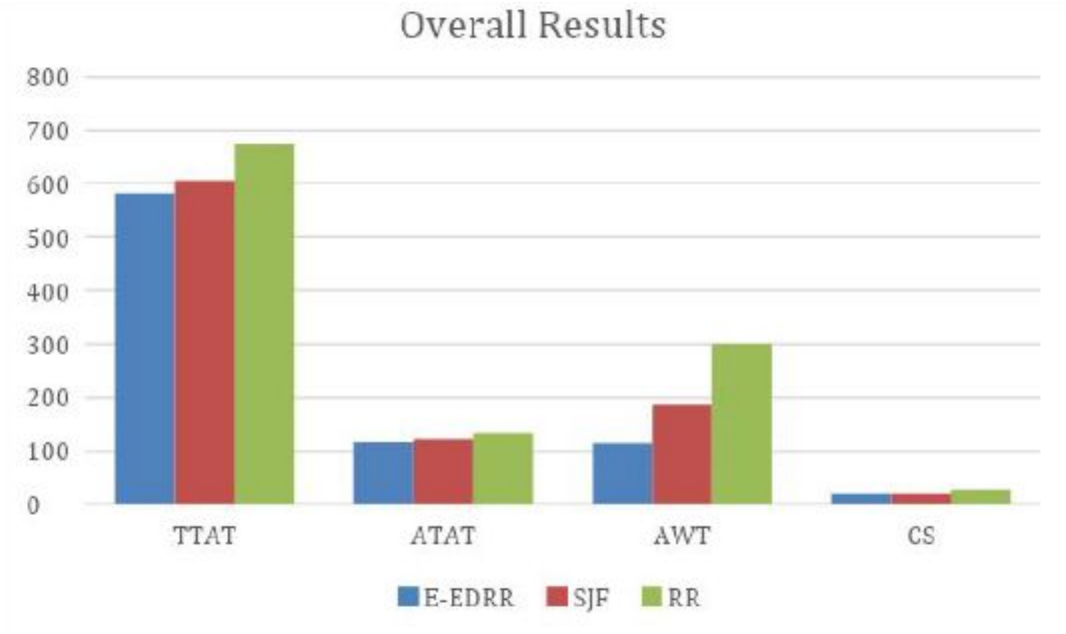
[Table 7: Test Cae 3 result]



[Table 9: Test Case 4 result]

Test Cases

Here is the Graph of the Overall Results:



DISCUSSIONS

Discussions

- E-EDRR has better performance with various Burst Times.
- RR has better TTAT in Test Case 1. Why?
- E-EDRR had consistent number of CS as the same as the SJF, but ironically, not RR as its parent concept.

Conclusions & Recommendations

Conclusions

- E-EDRR improves the CPU scheduling by reducing turnaround and waiting time without compromising in context switching.
- Results are align in conceptual analysis of the algorithm before actual testing
- We had confirmed that the algorithm’s procedural execution is better.

Recommendations

- E-EDRR can show better performance by threading implementation

- Test cases includes = Arrival Time (AT)
- Other Algorithms and stuffs can be compared to original algorithm

End to End (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis

2020 3rd International Conference on Computers in Management and Business (ICCMB2020)

Renz Rallion T. Gomez
Author/Student
University of the Cordilleras
Baguio City, Philippines
decemberavis19@gmail.com

Christopher M. Bermudez
Author/Student
University of the Cordilleras
Baguio City, Philippines
tupz0799@gmail.com

Vily Kaylle G. Cachero
Author/Student
University of the Cordilleras
Baguio City, Philippines
cacherokaylle@gmail.com

Eugene G. Rabang
Author/Student
University of the Cordilleras
Baguio City, Philippines
raterkracks@gmail.com

Rey Benjamin M. Baquirin
Co-Author/Professor
University of the Cordilleras
Baguio City, Philippines
reybenbaq@gmail.com