Analysis of CPU-Scheduling Algorithms and Real time Scheduling

Operating System

Maunil Vyas Deep Patel Shreyas Patel Karan Patel

School of Engineering and Applied Science
Ahmedabad University

Group - 22

Guidance By:Dr Sanjay Chaudhary,Shashvat Sanghavi

8 December, 2016

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Introduction

Processor Aim

The aim of processor scheduling is to assign processes to be executed by the processor or processors over time, in a way that meets system objectives, such as response time, throughput, and processor efficiency. In many systems, this scheduling activity is broken down into three separate functions: long-, medium-, and short-term scheduling, and those functions are handled by schedulers.

Introduction

Processor Aim

The aim of processor scheduling is to assign processes to be executed by the processor or processors over time, in a way that meets system objectives, such as response time, throughput, and processor efficiency. In many systems, this scheduling activity is broken down into three separate functions: long-, medium-, and short-term scheduling, and those functions are handled by schedulers.

Scheduling Mechanism in Support

As per the above claim scheduling is a crucial task to manage the efficiency of system. so here we have incorporated two kind of CPU scheduling schemes and analyze how they match up with certain scheduling criteria.

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Scheduling Criteria

User Oriented Criteria

- Turnaround time
- Waiting time
- Deadline

Scheduling Criteria

User Oriented Criteria

- Turnaround time
- Waiting time
- Deadline

System Oriented Criteria

- Throughput
- CPU Utilization
- Fairness

Scheduling Criteria

User Oriented Criteria

- Turnaround time
- Waiting time
- Deadline

System Oriented Criteria

- Throughput
- CPU Utilization
- Fairness

- High quality CPU scheduling algorithms generally focus on criterias such as throughput, CPU utilization rate, response time, turnaround time, and waiting time.

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Scheduling Algorithms

Generalize Scheduling Schemes

- Preemptive Scheduling.
- Non Preemptive Scheduling.
- Priority based Scheduling.

Scheduling Algorithms

Generalize Scheduling Schemes

- Preemptive Scheduling.
- Non Preemptive Scheduling.
- Priority based Scheduling.

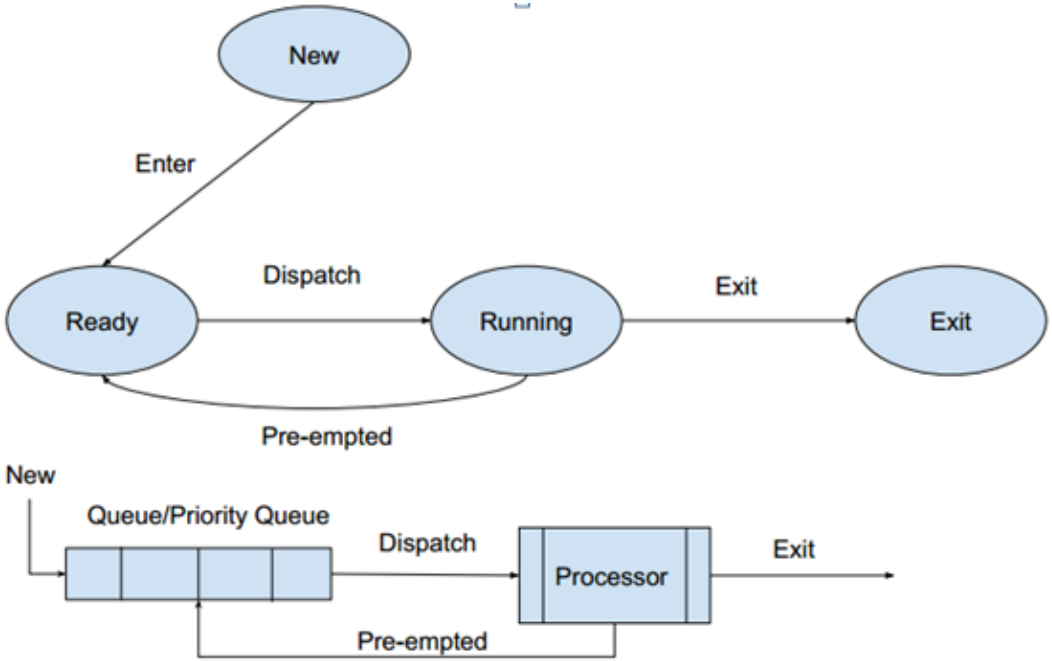


Figure : Scheduling State transactions diagram

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Simulation

Simulated Scheduling Algorithms

- FCFS - First Come First Serve
- SRN - Shortest Remaining Next
- RR - Round Robin(Quantum-6)
- PS - Priority Based Scheduling

Simulation

Simulated Scheduling Algorithms

- FCFS - First Come First Serve
- SRN - Shortest Remaining Next
- RR - Round Robin(Quantum-6)
- PS - Priority Based Scheduling

Analysis Criteria

- Average Waiting Time
- Average Turn Around Time
- Deadline Miss
- Average Throughput

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Average Waiting Time

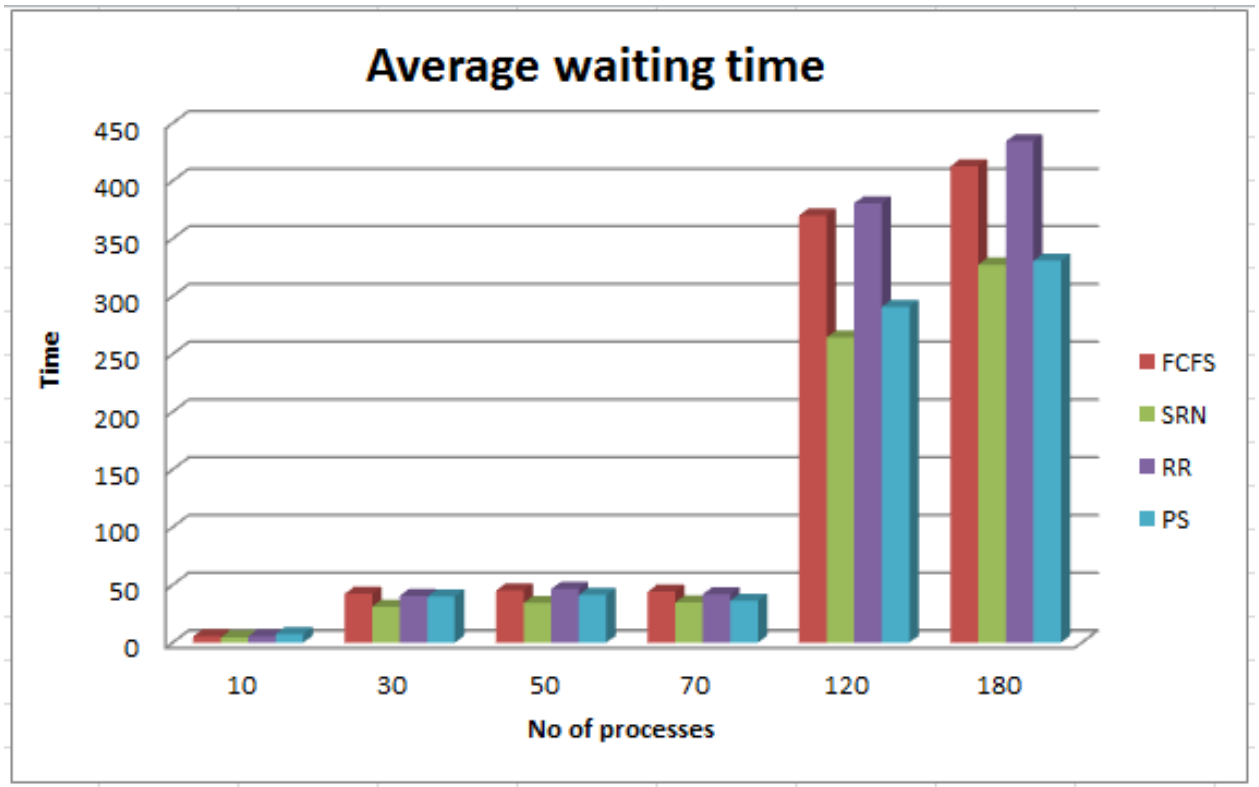
$$Average\ waiting\ time = \frac{\sum_{i=1}^{n} w_i}{n} \tag{1}$$

- n = number of processes, w_i = waiting time of i^{th} process

Average Waiting Time

$$Average\ waiting\ time = \frac{\sum_{i=1}^{n} w_i}{n} \quad (1)$$

- n = number of processes, w_i = waiting time of i^{th} process



Average Turn Around Time

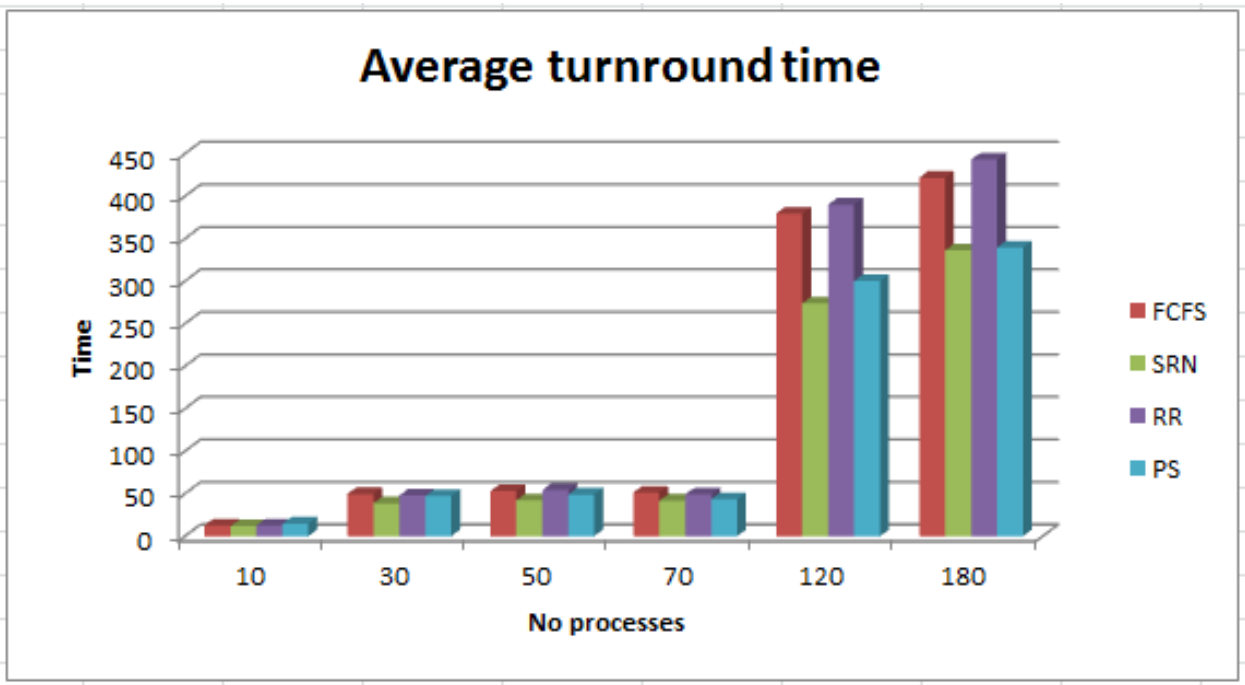
$$Average\ turnaround\ time = \frac{\sum_{i=1}^{n}(w_i + e_i)}{n} \tag{2}$$

- n = number of processes, w_i = waiting time of i^{th} process
  e_i = execution time of i^{th} process

Average Turn Around Time

$$Average\ turnaround\ time = \frac{\sum_{i=1}^{n}(w_i + e_i)}{n} \quad (2)$$

- n = number of processes, w_i = waiting time of i^{th} process
  e_i = execution time of i^{th} process



Average Throughput

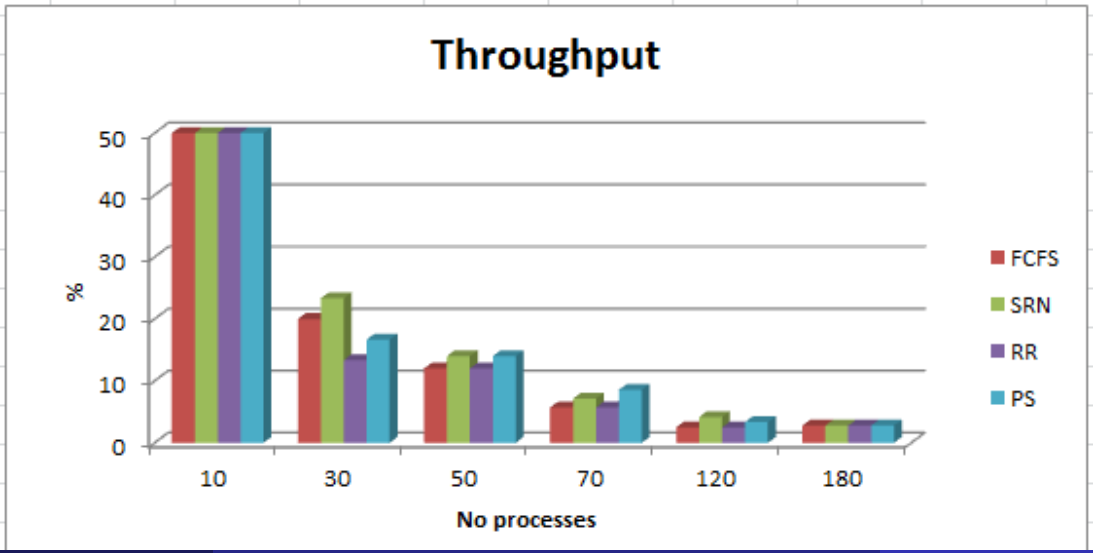
$$Average\ Throughput = \frac{n^{'}}{T} \tag{3}$$

- The number of processes completed successfully per time unit.
- n = number of processes , T = Time taken for completion of n processes , t = time value where t $<$ T $n^{'}$ = number of processes with finish time $<$ t
- Here, average through put means number of processes completed successfully before given time.

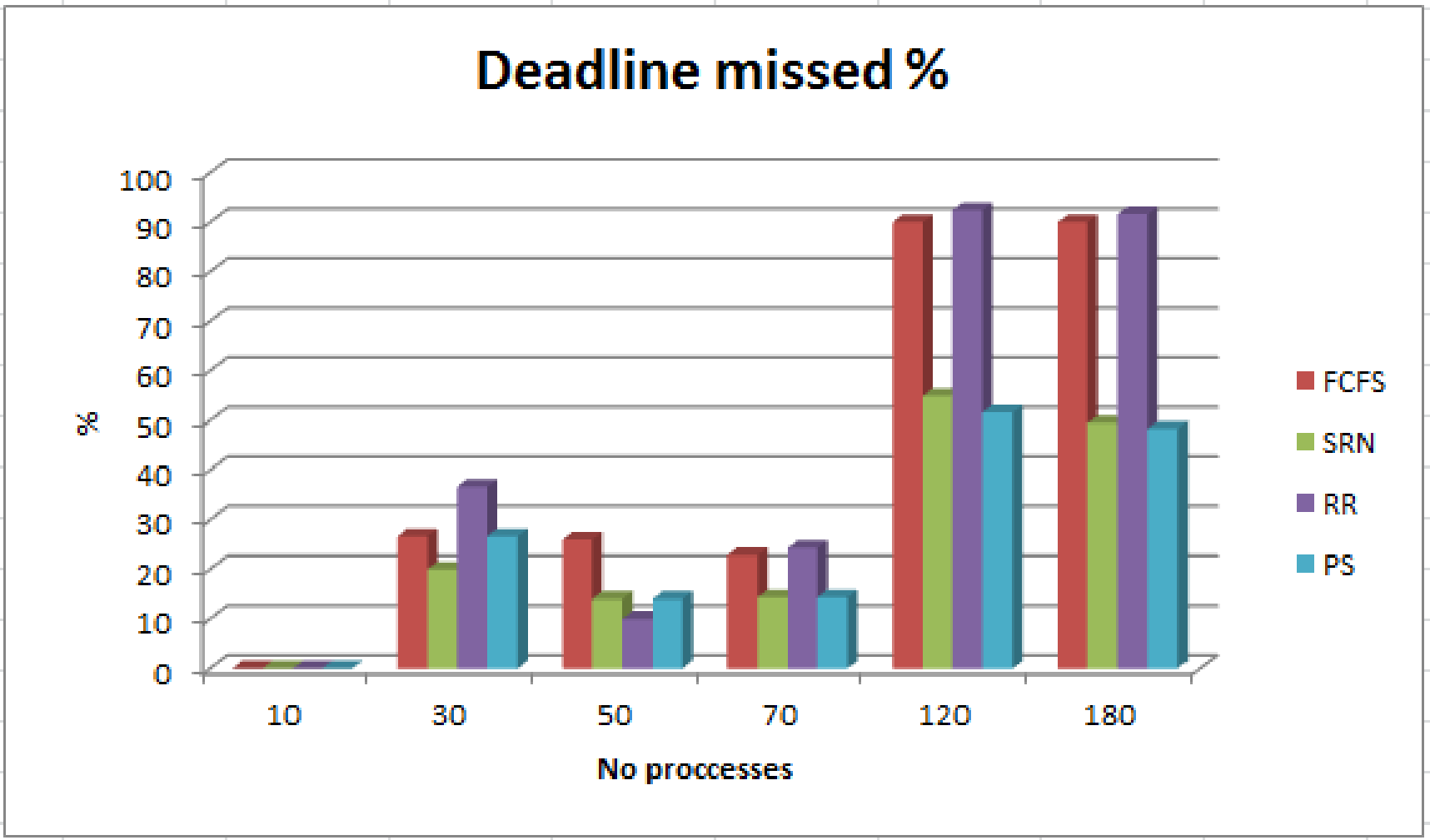
Average Throughput

$$Average\ Throughput = \frac{n'}{T} \tag{3}$$

- The number of processes completed successfully per time unit.
- n = number of processes , T = Time taken for completion of n processes , t = time value where t < T n' = number of processes with finish time < t
- Here, average through put means number of processes completed successfully before given time.



Deadline Missed



Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Why Real time Scheduling and It's required parameters

- From the previous deadline miss result we can say that this type of scheduling algorithms, may break down performance of Real time systems very badly because those systems are depended on fixed deadlines.

Why Real time Scheduling and It's required parameters

- From the previous deadline miss result we can say that this type of scheduling algorithms, may break down performance of Real time systems very badly because those systems are depended on fixed deadlines.
- A common characteristic of real time system is that their requirements specification includes timing information in the form of deadlines which traditional scheduling algorithms not emphasize

Why Real time Scheduling and It's required parameters

- From the previous deadline miss result we can say that this type of scheduling algorithms, may break down performance of Real time systems very badly because those systems are depended on fixed deadlines.
- A common characteristic of real time system is that their requirements specification includes timing information in the form of deadlines which traditional scheduling algorithms not emphasize
- Based on deadlines we can classify the deadline consequences
  - Hard Deadlines
  - Soft Deadlines

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Deadline Criteria and General Classes of Real time Scheduling algorithms

Deadline Criteria

$$C \leq D$$

$C \leq D \leq T$ For Periodic Process

where C - Computation time , D - Deadline , T - Period

Deadline Criteria and General Classes of Real time Scheduling algorithms

Deadline Criteria

$$C \leq D$$

$C \leq D \leq T$ For Periodic Process

where C - Computation time , D - Deadline , T - Period

General Classes of Real time Scheduling algorithms

- Static table-driven approaches
- Static priority-driven preemptive approaches
- Dynamic planning-based approaches
- Dynamic best effort approaches:

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Assumptions and Algorithms

Assumptions

- Ci $\leq$ Di = Pi (i.e. The processes have computation time less than their deadline, and the deadline is equal to their period).
- Computation times for a given process are constant.
- No precedence relations exist between processes.
- No inter-process communication or synchronization is permitted.
- Context switches have zero cost.
- All processes are allocated to a single processor.

Assumptions and Algorithms

Assumptions

- Ci $\leq$ Di = Pi (i.e. The processes have computation time less than their deadline, and the deadline is equal to their period).
- Computation times for a given process are constant.
- No precedence relations exist between processes.
- No inter-process communication or synchronization is permitted.
- Context switches have zero cost.
- All processes are allocated to a single processor.

Implemented algorithms

- RMS - Rate Monotonic Scheduling
- EDF - Earliest Deadline First

RMS

- RMS is static priority driven pre-emptive approach.

RMS

- RMS is static priority driven pre-emptive approach.
- RMS requires preemptive scheduler; all processes are allocated a priority according to their period. Shorter the period, higher their priority.

RMS

- RMS is static priority driven pre-emptive approach.
- RMS requires preemptive scheduler; all processes are allocated a priority according to their period. Shorter the period, higher their priority.

Criteria

$$\frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + ... + \frac{C_n}{P_n} = U \leq n(2^{\frac{1}{n}} - 1)$$

- C - Computation time , P - Process Period.

RMS Implementation

```
int Period[] = {10,5,30,15};
int Execution[] = {2,1,5,2};
process *Array[N];
```

```
// Register signal and signal handler
signal(SIGINT, signal_callback_handler);
```

```
// Define the function to be called when ctrl-c (SIGINT) signal is sent to process
void signal_callback_handler(int signum)
{
   outputfile<<" Interrupt is Caught :- signal "<<signum<<endl;
   // Cleanup and close up stuff here
   outputfile<<" Interrupt is disabled. "<<endl;
   // Terminate program
   cout<<" Interrupt is Caught :- signal "<<signum<<endl;
   // Cleanup and close up stuff here
   cout<<" Interrupt is disabled. "<<endl;
}
```

```
Process 1 created successfully.
Process 2 created successfully.
Process 3 created successfully.
Process 4 created successfully.

CPU Utilization :-0.700000 Theoritical :- 0.756828
Process No. -> Waiting time -> Turn around time

ProcessID : 1 state : ready Tue Dec  6 23:38:29 2016

ProcessID : 2 state : ready Tue Dec  6 23:38:29 2016

ProcessID : 3 state : ready Tue Dec  6 23:38:29 2016

ProcessID : 4 state : ready Tue Dec  6 23:38:29 2016


ProcessID : 2 state : running Tue Dec  6 23:38:29 2016

ProcessID : 2 state : exit Tue Dec  6 23:38:30 2016

ProcessID : 2 -> 0 -> 1
ProcessID : 1 state : running Tue Dec  6 23:38:30 2016

ProcessID : 1 state : exit Tue Dec  6 23:38:32 2016

ProcessID : 1 -> 1 -> 3
ProcessID : 4 state : running Tue Dec  6 23:38:32 2016

ProcessID : 4 state : exit Tue Dec  6 23:38:34 2016
```

EDF

- The process with the(current) closest deadline is assigned the highest priority in the system and therefore it executes first.

EDF

- The process with the(current) closest deadline is assigned the highest priority in the system and therefore it executes first.

Criteria

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1 \tag{4}$$

EDF

- The process with the(current) closest deadline is assigned the highest priority in the system and therefore it executes first.

Criteria

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1 \tag{4}$$

- Hence, 100 % processor utilization is possible.

EDF Implementation

```
int deadline[] = {15,12,9,8};
int Execution[] = {4,3,5,2};
int Arrivaltime[] = {0,0,2,5};
```

```
Process 1 created successfully.
Process 2 created successfully.
Process 3 created successfully.
Process 4 created successfully.
ProcessID : 1 state : ready Tue Dec  6 23:40:14 2016

ProcessID : 2 state : ready Tue Dec  6 23:40:14 2016

ProcessID : 2 state : running Tue Dec  6 23:40:14 2016

ProcessID : 2 state : ready Tue Dec  6 23:40:16 2016

ProcessID : 3 state : ready Tue Dec  6 23:40:16 2016

ProcessID : 3 state : running Tue Dec  6 23:40:16 2016

ProcessID : 3 state : ready Tue Dec  6 23:40:19 2016

ProcessID : 4 state : ready Tue Dec  6 23:40:19 2016

ProcessID : 4 state : running Tue Dec  6 23:40:19 2016

ProcessID : 4 state : exit Tue Dec  6 23:40:21 2016

ProcessID : 4 -> 2 -> 0 -> 2
ProcessID : 3 state : running Tue Dec  6 23:40:21 2016

ProcessID : 3 state : exit Tue Dec  6 23:40:23 2016

ProcessID : 3 -> 5 -> 2 -> 7
ProcessID : 2 state : running Tue Dec  6 23:40:23 2016
```

EDF Deadline Missed

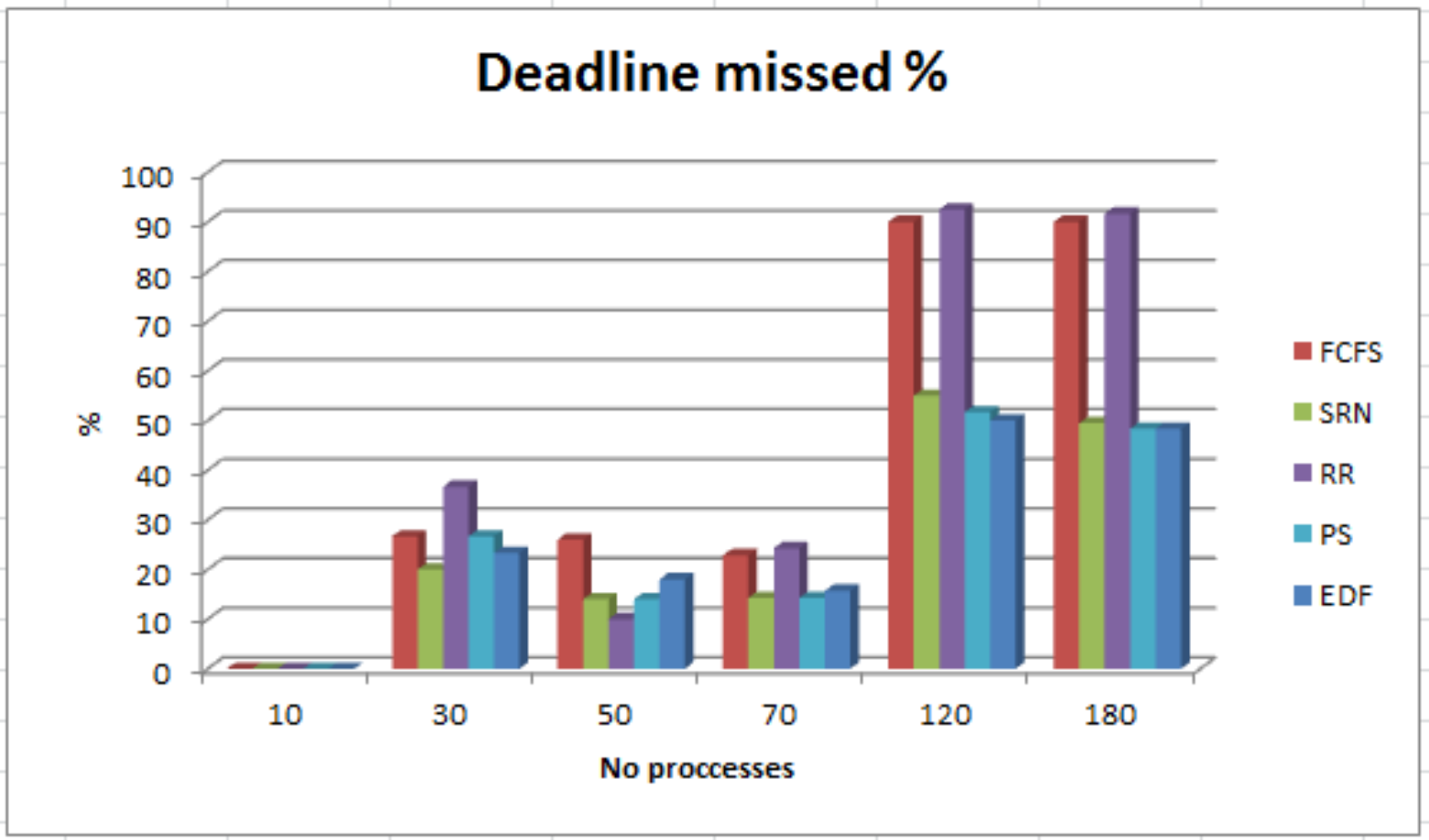


Figure : Deadline Miss with EDF

Outline

1. Project Overview
   - Introduction
2. Scheduling
   - Scheduling Criteria
3. CPU Scheduling Algorithms
   - Scheduling Algorithms
   - Simulation Considering Uniprocessor
   - Simulation Results
4. Real time Scheduling
   - Why Real time Scheduling and It's required parameters
   - Deadline Criteria and General Classes of Real time Scheduling algorithms
   - Implementation
5. Conclusion
   - Conclusion

Conclusion

- We compared various CPU Scheduling Algorithms both static and real time and did their simulation in C++, after an- alyzing their results, we concluded that traditional scheduling algorithms like FCFS, RR, SJF, PS etc. missed lots of dead- lines. Therefore we conclude that, these type of scheduling algortihms cannot be used for scheduling in Real time systems and the specialized real time algorithms like RMS and EDF are needed for process scheduling in Real time systems.

References I


William Stalling,
Operating Systems: Internals and Design Principles, 7- 6th edition.
PEARSON


Sultan Almakdi, Mohmmad Alesia, Mohmmad Alsheri Simulation and Performance Evaluation of CPU Scheduling Algorithms.


WIKIPEDIA.
For Images.