# METODE ALTERNATIF PENETAPAN TIME QUANTUM DINAMIS UNTUK PERBAIKAN KINERJA PENJADWALAN ROUND ROBIN PADA PROSESSOR TUNGGAL

**Tati Erlina**

tatierlina@fmipa.unand.ac.id
Sistem Komputer, Fakultas Teknologi Informasi Universitas Andalas

**Abstrak**

*Round Robin (RR) merupakan algoritma penjadwalan CPU yang populer digunakan dalam interactive dan multiprogramming systems. Algoritma ini banyak digunakan karena dapat meminimalkan besarnya waiting time pada proses pendek yang merupakan masalah utama pada algoritma First Come First Serve (FCFS). Perbaikan terhadap kinerja algoritma Round Robin yang diusulkan pada paper ini dilakukan dengan mengkombinasikan konsep Shortest Job First (SJF) scheduling yang dilakukan dengan mengurutkan semua proses yang sudah berada pada ready queue dan Round Robin scheduling yang penentuan time quantum-nya dilakukan secara dinamis dengan memanfaatkan rumus desil ke-8 (D8). Pada analisa hasil percobaan yang dilakukan, terlihat bahwa terjadi perbaikan kinerja algoritma ini dalam jumlah context switch yang diperlukan, average waiting time, dan average turnaround time.*

***Kata kunci—*** *round robin, ready queue, waiting time, context switch, turnaround time*

**Abstract**

*Round Robin is a popular CPU scheduling algorithm in interactive and multiprogramming systems. This is the result of the ability of Round Robin algorithm to minimize the amount of waiting time in short processes which becomes a main problem in First Come First Serve (FCFS) algorithm. Performance improvement proposed in this algorithm is done by combining Shortest Job First (SJF) scheduling concept by sorting all processes in ready queue and Round Robin scheduling where it's time quantum determined dynamically by utilizing the eighth decile (D8) formula. Experimental analysis result shows preformance improvement in terms of context switch needed, average waiting time and turnaround time.*

***Keyword—*** *round robin, ready queue, waiting time, context switch, turnaround time*

## 1. Pendahuluan

### 1.1 Latar Belakang

Penetapan besarnya *time quantum* merupakan tantangan yang memiliki konsekuensi besar terhadap kinerja sebuah sistem operasi yang menerapkan algoritma *Round Robin*. Jika *time quantum* terlalu kecil, maka akan mengakibatkan jumlah *context switch* yang terjadi sangat besar. Sebaliknya, jika *time quantum* terlalu besar, akan menyebabkan algoritma *Round Robin* memiliki masalah yang hampir sama dengan algoritma FCFS.

Silberschatz, dkk (2013) menyatakan bahwa peningkatan besar *time quantum*, tidak selalu meningkatkan *average turnaround time* dari serangkaian proses. Akan tetapi, average turnaround time dapat meningkat jika mayoritas proses dapat diselesaikan dalam satu *time quantum*. Selain itu, berdasarkan "*rule of thumb*" (Behera, 2011) bahwa 80 persen dari *CPU bursts* harus lebih pendek dibanding besarnya *time quantum*, penulis mengajukan alternatif perbaikan metode untuk menetapkan *time quantum* pada *Round Robin* traditional.

### 1.2 Perumusan Masalah

Kinerja algoritma *Round Robin* sangat tergantung pada besarnya *time quantum* yang diberikan pada proses-proses yang akan dieksekusi. Bagaimana cara menetapkan besarnya *time quantum* agar kinerja algoritma *Round Robin* dapat meningkat?

### 1.3 Tujuan Penelitian

Metode mendapatkan time quantum yang tepat dapat diterapkan pada algoritma *Round Robin* sehingga dapat menurunkan *average waiting time, average turnaroud time* dan jumlah *context switch* yang dibutuhkan saat processor harus mengeksekusi sejumlah proses yang berada pada *ready queue*.

### 1.4 Manfaat Penelitian

Penelitian ini menghasilkan sebuah algoritma yang dapat meningkatkan kinerja algoritma *Round Robin* tradisional.

## 2. Tinjauan Pustaka

Dalam beberapa tahun terakhir, beberapa penelitian untuk meningkatkan kinerja *Round Robin* scheduling telah dilakukan. Algoritma MDTQRR (Behera, 2011), menghitung *time quantum* sebanyak 2 (dua) kali dalam satu siklus *Round Robin* dan mempertimbangkan *arrival time* dalam mengembangkan algoritma tersebut. Algoritma SRBRR (Varma, 2012), mengalokasikan prosesor ke sebuah proses yang memiliki *burst time* terpendek ke dalam algoritma *Round Robin* dengan *time quantum* yang dihitung dengan bantuan median dan *burst time* tertinggi. Algoritma AMMRR (Banerjee, 2012) menetapkan *time quantum* secara dinamis dengan memanfaatkan nilai rata-rata *burst time* seluruh proses dan rata-rata nilai maksimal pada proses tersebut.

### 2.1 Landasan Teori

CPU *scheduling* berkenaan dengan pengaturan pengalokasian processor ke proses-proses yang membutuhkannya. Dalam pengaturan tersebut, CPU *scheduling* memiliki beberapa pilihan *scheduling algorithms* yang dapat digunakan, yang masing-masingnya memiliki beragam karakteristik. Tujuan utama dari penggunaan *scheduling algorithms* (Pant, 2011) ini adalah untuk meminimalkan *resource starvation* dan untuk memastikan adanya *fairness* diantara sumberdaya yang memanfaat processor.

Dalam melaksanakan tugasnya, *scheduler*, biasanya memperhatikan beberapa aspek yang menjadi parameter efisiensi (Goel, 2012) dari sebuah *scheduling algorithms*, diantaranya :

**a. *CPU Utilization*.**

Adalah rasio antara *busy time* dari sebuah *processor* dengan waktu yang dihabiskan oleh sebuah proses sampai proses tersebut selesai.

***Processor Utilization* = (*Processor busy time*) / (*Processor busy time* + *Processor idle time*)**

Tingkat penggunaan CPU yang diinginkan adalah setinggi mungkin, nilainya dapat berkisar dari 0 sampai 100 persen. Pada sistem sebenarnya, tingkat penggunaan ini mulai dari 40 persen sam 90 persen.

***b. Throughput.***

Merupakan jumlah proses yang dapat diselesaikan per unit waktu.

**Throughput = (Number of processes completed) / (Time Unit).**

Selain tergantung pada panjang rata-rata dari sebuah proses, *throughput* juga tergantung pada aturan *scheduling algorithm* yang digunakan.

c. ***Turnaround Time***.

*Turnaround time* adalah interval waktu yang dibutuhkan antara masuknya sebuah proses ke *ready queue* sampai proses tersebut selesai.

**tat = t (process completed) – t (process submitted).**

Didalamnya termasuk waktu eksekusi sebenarnya ditambah dengan waktu yang dibutuhkan untuk menunggu sumberdaya termasuk processor.

***d. Waiting Time.***

Adalah jumlah waktu yang dibutuhkan oleh sebuah proses dalam *ready queue.* Jenis *CPU scheduling* yang digunakan sangat mempengaruhi *waiting time*.

**wt = t (start) - t(arrival)**

***e. Response Time***

Response Time adalah jumlah waktu yang dibutuhkan untuk memulai respon terhadap sebuah *request.* Kriteria ini merupakan kriteria yang sangat penting pada *interactive systems.*

**rt = t(first response) – t(submission of request).**

Dari sudut pandang user, *response time* merupakan parameter yang lebih baik dibanding *turnaround time*.

Pada CPU *scheduling*, terdapat usaha untuk meminimalkan *response time* dan memaksimalkan jumlah *interactive users* yang menerima *response time* yang dapat ditolerir. Selain itu, *turnaorund time* dan *response time* yang diinginkan adalah seminimal mungkin. Sebaliknya, CPU *scheduling* berusaha untuk memaksimalkan CPU *utilization* dan *throughput.*

Tolak ukur yang akan digunakan sebagai pembanding pada paper ini adalah:

1. *Turnaround Time*(TAT)
   Nilai rata-rata *turnaround time* sebaiknya seminimal mungkin.
2. *Waiting Time* (WT)
   Nilai rata-rata w*aiting time* sebaiknya seminimal mungkin.
3. *Context Switch* (CS)
   Jumlah *context switch* yang terjadi sebaiknya seminimal mungkin.

### *ROUND ROBIN SCHEDULING ALGORITHM*

*Round Robin* merupakan tipe penjadwalan yang *decision mode*-nya termasuk dalam kategori *preemptive scheduling* yang mengalokasikan jumlah sumber daya pemrosesan yang sama pada proses yang membutuhkan (Pinedo, 2012). Pada *decision mode* ini, proses yang sedang berjalan (dieksekusi *processor*) dapat diinterupsi dan dipindahkan ke *ready state* oleh sistem operasi. Keputusan untuk menghentikan

proses yang sedang dieksekusi ini dapat terjadi karena terciptanya proses baru, atau karena adanya *request* secara periodik terhadap layanan sistem operasi berdasarkan *clock interrupt (*Stalling, 2012*).*

Setiap proses pada sistem operasi yang menerapkan round robin sebagai *scheduling algorithm*-nya, memiliki interval waktu (*time slice* atau *time quantum*) yang menunjukkan seberapa lama proses tersebut diizinkan untuk dieksekusi oleh *processor* (Doeppner, 2011). Jika sebuah proses masih berjalan setelah satu *time quantum* yang dialokasikan habis, maka CPU akan dialihkan untuk melayani proses lain. Proses yang belum selesai tersebut dikembalikan ke *ready queue dan* akan dilanjutkan eksekusinya oleh CPU pada *quantum* yang didapatkan pada kesempatan berikutnya. Sebaliknya, jika sebuah proses sudah berstatus *blocked* atau *finished* sebelum *quantum*-nya habis, maka CPU akan dialihkan untuk melayani proses lain.

Isu penting dalam *Round Robin* scheduling adalah penetapan besarnya quantum (Tanenbaum, 2008). Besarnya quantum yang digunakan akan berpengaruh langsung terhadap jumlah *preemption* yang akan dilakukan oleh sistem operasi dan jumlah *context switch* yang terjadi pada sistem. Setiap perpindahan layanan *processor* dari satu proses ke proses lain akan membutuhkan *context switch*, dimana *processor* memerlukan waktu untuk menyimpan *state* dari satu proses sebelum berpindah pada proses lain dan memuat *state* dari proses yang akan dilayaninya ke memori. Jika *quantum* yang ditetapkan terlalu kecil, maka akan menyebabkan terlalu banyak *context switch* yang dibutuhkan, *throughput* dan efisiensi CPU menjadi rendah. Akan tetapi, *time quantum* yang terlalu besar akan menyebabkan *response time* sistem tidak bagus terutama pada sistem interaktif, selain itu quantum yang terlalu besar dapat menyebabkan sistem cendrung mempunyai karakteristik seperti sistem yang menggunakan algoritma FCFS (First In First Out) (Goel, 2012).

## 3. Metode Penelitian

Sebelum eksperimen dilakukan, sejumlah asumsi ditetapkan, diantaranya adalah bahwa semua percobaan dilakukan pada lingkungan processor tunggal dimana semua proses tidak tergantung satu sama lain. Selain itu, parameter-parameter yang diperlukan, seperti jumlah proses yang ada pada *ready queue*, *burst time* masing-masing proses, *arrival time* dan besarnya *time quantum* sudah ditentukan sebelum proses dieksekusi oleh processor.

Eksperimen yang dilakukan dalam penelitian ini memiliki beberapa parameter input dan output. Parameter output yang digunakan adalah *average turnaround time, average waiting time* dan jumlah *context switch yang diperlukan.* Sedangkan parameter input yang digunakan adalah jumlah proses, *arrival time*, dan *burst time* masing-masing proses. Jumlah proses dan *burst time* masing-masing proses tersebut kemudian dijadikan dasar untuk penghitungan besarnya *time quantum* yang dihitung dengan memanfaatkan rumus D8 (jika jumlah data yang ada pada *ready queue* besar sama dengan 4 ), yaitu :

$$Di = i(n + 1)/10 \quad (1)$$

dimana,
i : data ke i
n : jumlah proses
Di: desil ke i

Pada kondisi sebaliknya, *time quantum* dihitung menggunakan rumus median data tunggal ganjil atau genap, yaitu :

$$\text{Me Odd} = Y_{(n+1)/2} \quad (2)$$
$$\text{Me Even} = (Y_{n/2} + Y_{1+(n/2)})/2$$

dimana,
Y: letak proses yang menjadi median
n: jumlah proses

Untuk membandingkan kinerja dari algoritma *Round Robin* yang penetapan besarnya *time quantum* seperti yang diusulkan (D8RR) dengan algoritma *Round Robin* tradisional (RR), digunakan serangkaian proses dengan 3 jenis urutan nilai *burst time*, yaitu, menaik, menurun dan acak. D8RR dapat bekerja dengan baik pada data yang jumlahnya cukup besar, akan tetapi, untuk menyederhanakan, disini digunakan serangkaian proses yang terdiri atas lima sampai tujuh buah proses.

Adapun pseudocode dari algoritma yang diusulkan dalam D8RR ini adalah sebagai berikut :

1. Deklarasikan : jumlah proses (n), arrival time (at[]), burst time (bt[]), average waiting time (atat), average waiting time (awt), time quantum (tq), sisa burst time (sbt[])
2. Inisialisasi: atat =0, awt=0
3. Masukkan n, at, bt
4. Urutkan semua proses berdasarkan bt secara menaik.
5. While (n!=0)
   If n >=4 then
   tq = desil ke-8
   if 2<= n < 4
   tq = median
   else
   tq = bt
   sbt[i] = bt[i] – tq
6. Jika sbt[i] = 0,
   Update n
7. Jika ada proses baru masuk,

Update n, kembali ke langkah 4
8. End of while
9. Hitung atat, awt

## 4. Hasil dan Penelitian

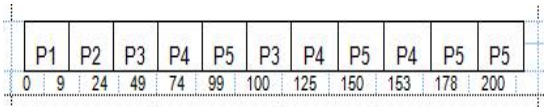
Kasus-1: Diasumsikan terdapat lima buah proses yang memiliki *arriving time* = 0, dan memiliki *burst time* yang besarnya menaik (Tabel 1). Tabel 2 memperlihatkan perbandingan kinerja antara kedua algoritma tersebut. Gambar 1 dan Gambar 2 memperlihatkan Gantt Chart dari kedua algoritma.

**Tabel 1. Proses dan *Burst Time***

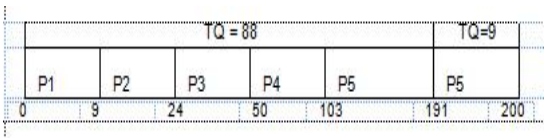
| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 9 |
| P2 | 0 | 15 |
| P3 | 0 | 26 |
| P4 | 0 | 53 |
| P5 | 0 | 97 |

**Tabel 2. Perbandingan antara RR dengan D8RR**

| Algoritma | Time Quantum | Average TAT | Average WT | Context Switch |
|---|---|---|---|---|
| RR | 25 | 97.2 | 57.2 | 10 |
| D8RR | 88, 9 | 77.2 | 37.2 | 5 |




**Gambar 1. Gantt chart RR**




**Gambar 2. Gantt chart D8RR**

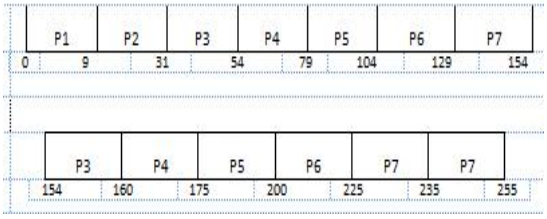
Kasus-2: Diasumsikan terdapat tujuh buah proses yang memiliki *burst time* yang besarnya juga menaik (Tabel 3), akan tetapi memiliki *arriving time* yang berbeda-beda. Tabel 4 memperlihatkan perbandingan kinerja antara kedua algoritma tersebut. Gambar 3 dan Gambar 4 memperlihatkan Gantt Chart dari kedua algoritma.

**Tabel 3. Proses dan *Burst Time***

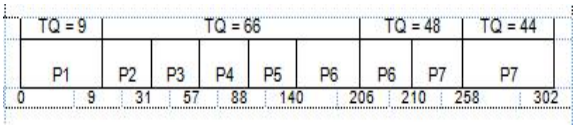
| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 9 |
| P2 | 1 | 22 |
| P3 | 3 | 26 |
| P4 | 5 | 31 |
| P5 | 6 | 52 |
| P6 | 8 | 70 |
| P7 | 19 | 92 |

**Tabel 4. Perbandingan antara RR dengan D8RR**

| Algoritma | Time Quantum | Average TAT | Average WT | Context Switch |
|---|---|---|---|---|
| RR | 25 | 160 | 116 | 12 |
| D8RR | 9,66,48,44 | 159 | 98.6 | 8 |




**Gambar 3. Gantt chart RR**




**Gambar 4. Gantt chart D8RR**

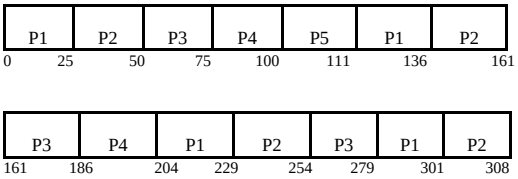
Kasus-3: Diasumsikan terdapat lima buah proses yang memiliki *arriving time* = 0, dan memiliki *burst time* yang besarnya menurun (Tabel 5). Tabel 6 memperlihatkan perbandingan kinerja antara kedua algoritma tersebut. Gambar 5 dan Gambar 6 memperlihatkan Gantt Chart dari kedua algoritma.

**Tabel 5. Proses dan *Burst Time***

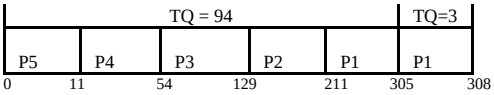
| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 97 |
| P2 | 0 | 82 |
| P3 | 0 | 75 |
| P4 | 0 | 43 |
| P5 | 0 | 11 |

**Tabel 6. Perbandingan antara RR dengan D8RR**

| Algoritma | Time Quantum | Average TAT | Average WT | Context Switch |
|---|---|---|---|---|
| RR | 25 | 240.6 | 179 | 13 |
| D8RR | 94, 3 | 142.6 | 81 | 5 |




**Gambar 5. Gantt chart RR**




**Gambar 6. Gantt chart D8RR**

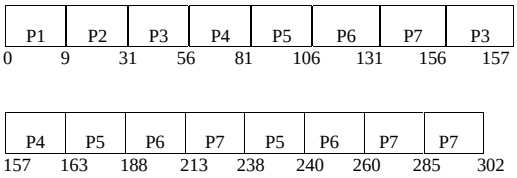
Kasus-4: Diasumsikan terdapat tujuh buah proses yang memiliki *burst time* yang besarnya juga menurun (Tabel 7), akan tetapi memiliki *arriving time* yang berbeda-beda. Tabel 8 memperlihatkan perbandingan kinerja antara kedua algoritma tersebut. Gambar 7 dan Gambar 8 memperlihatkan Gantt Chart dari kedua algoritma.

**Tabel 8. Proses dan *Burst Time***

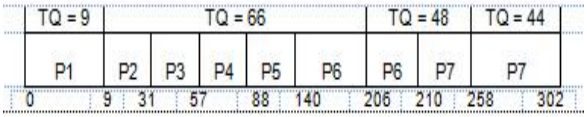
| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 81 |
| P2 | 8 | 77 |
| P3 | 12 | 50 |
| P4 | 15 | 40 |
| P5 | 21 | 35 |
| P6 | 45 | 20 |
| P7 | 82 | 15 |

**Tabel 9. Perbandingan antara RR dengan D8RR**

| Algoritma | Time Quantum | Average TAT | Average WT | Context Switch |
|---|---|---|---|---|
| RR | 25 | 231 | 169 | 15 |
| D8RR | 81,66,13, 2 | 214 | 168 | 8 |




**Gambar 7. Gantt chart RR**




**Gambar 8. Gantt chart D8RR**

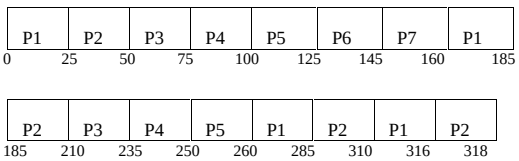
Kasus-5: Diasumsikan terdapat lima buah proses yang memiliki *arriving time* = 0, dan memiliki *burst time* yang besarnya acak (Tabel 9). Tabel 10 memperlihatkan perbandingan kinerja antara kedua algoritma tersebut. Gambar 9 dan Gambar 10 memperlihatkan Gantt Chart dari kedua algoritma.

**Tabel 9. Proses dan *Burst Time***

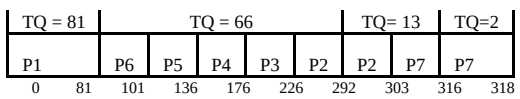
| Proses | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 60 |
| P2 | 0 | 40 |
| P3 | 0 | 82 |
| P4 | 0 | 4 |
| P5 | 0 | 91 |

**Tabel 10. Perbandingan antara RR dengan D8RR**

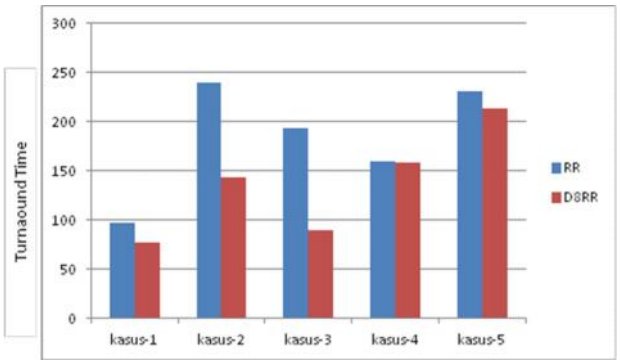
| Algoritma | Time Quantum | Average TAT | Average WT | Context Switch |
|---|---|---|---|---|
| RR | 25 | 193 | 137,6 | 13 |
| D8RR | 89,2 | 88,8 | 33,4 | 5 |



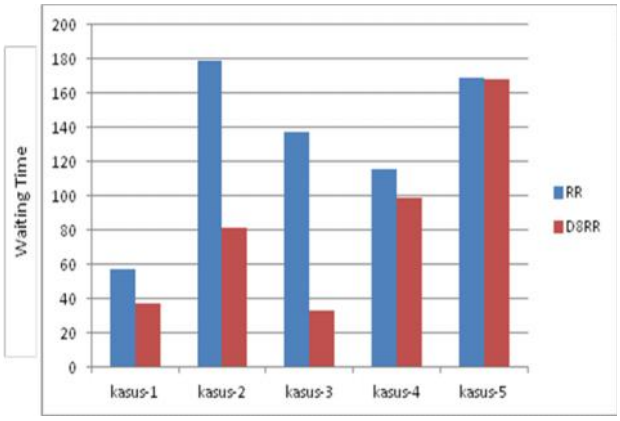

**Gambar 9. Gantt chart RR**



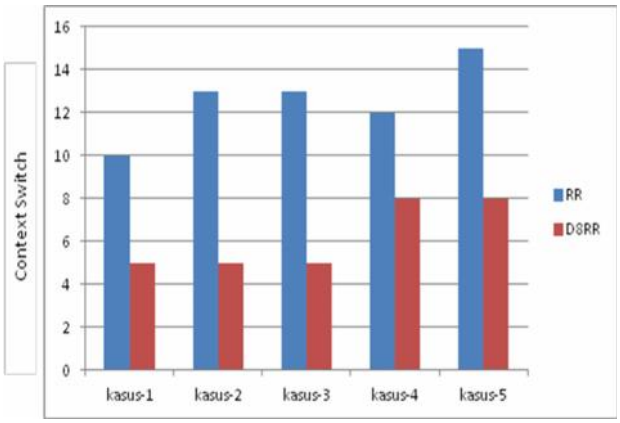

**Gambar 10. Gantt chart D8RR**




**Gambar 11. Perbandingan Turnaround Time RR dan D8RR**




**Gambar 12. Perbandingan Waiting Time RR dan D8RR**




**Gambar 13. Perbandingan Context Switch RR dan D8RR**

## 5. Kesimpulan dan Saran

Dari perbandingan diatas, terlihat bahwa algoritma *Round Robin* yang besar *time quantum*-nya dihitung menggunakan algoritma yang diusulkan (D8RR) memiliki kinerja yang lebih baik dibanding algoritma *Round Robin* tradisional (RR) dalam hal *average turnaround time, average waiting time* dan jumlah *context switch* yang dibutuhkan. Dengan demikian penerapan algoritma D8RR pada *processor* tunggal akan dapat menghemat penggunaan memory, dan memperkecil *overhead.* Pada penelitian selanjutnya, *response time* dan *throughput* sebaiknya dimasukkan sebagai parameter yang dijadikan tolak ukur dalam perbandingan. Kedua parameter tersebut juga sangat penting untuk diperhitungkan dalam *interactive systems*, baik dari sudut pandang *user* maupun *processor*.

## 6. Daftar Pustaka

Banerjee, Pallab, 2012, *Performance Evaluation of a New Proposed Average Mid Max Round Robin (AMMRR) Scheduling Algorithm with Round Robin Scheduling Algorithm, IJARCSSE*, Vol. 2 No 8, pp.143-151.

Behera, H., S., dkk. 2011. *Comparative Performance Analysis of Multi-Dynamic Time Quantum Round Robin (MDTQRR) Algorithm with Arrival Time*, IJCSE, Vol. 2 No 2, pp. 262-271.

Doeppner, Thomas, W. 2011. *Operating Systems in Depth*, Wiley.

Goel, Neetu, 2012, *A Comparative Study of CPU Scheduling Algorithms*, IJGIP, Vol.2 No 4, pp. 245-251.

Pant, Alka, 2011, *A Comparison between FCFS and Mixed Scheduling, IJCST*, Vol. 2 No 2, pp. 76-79.

Pinedo, Michael, L., 2012, *Scheduling: Theory, Algorithms and Systems*, Third Edition, Springer.

Silberschatz, Abraham, dkk, 2013, *Operating System Concepts*, Nineth Edition, Wiley.

Stallings, William, 2012. *Operating Sysems Internals and Design Principals*, 2012, Seventh Edition, Prentice Hall.

Tanenbaum, Andrew, S., 2008, *Modern Operating Systems*, Third Edition, Prentice Hall.

Varma, Surendra, P., *A Possible Time Quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) Algorithm*, IJARCSSE, Vol. 2 No 11, pp. 228-237.