GLOBAL SEEK OPTIMIZATION IN REAL-TIME DATABASE TRANSACTIONS: A NEW APPROACH

1SALIM Y. AMDANI
Assoc. Professor and Head, Dept. of CSE, B. N. C. O. E, Pusad (India)
salimamndani@yahoo.com

2DR. M. S. ALI
Prof. & Principal, P. R. M. C. E. & M, Badnera (India)
softalis@gmail.com

3ANUPAMA C. GIRAM
M.E. Student, Dept. of CSE, B. N. C. O. E, Pusad (India)
anu26giram@gmail.com

ABSTRACT

Real time database transactions must satisfy database consistency as well as timing constraints. Timing constraints are defined in terms of deadline. Numbers of algorithms are proposed to schedule real time database transactions in order to increase the overall performance of the disk. In this paper, we have proposed a new approach based on Two-way scan algorithm which gives better result than Two-way scan. In our new approach, Shortest Seek Time (SST) groups are formed like Two-way scan and then priority is assign to each transaction. Transaction having lowest priority value is served first. Consecutive groups are served one after the other. Transactions are served only if they are feasible.

KEYWORDS: EDF, DM-SCAN, Two-way Scan, Deadline, Real-Time, Transactions.

1. INTRODUCTION

Database Management systems have entered the Internet Age. When multiple users want to access information, then the system performance is degraded. This may cause delay and trouble for particular end user. Accessing information in easy way and within certain time constraints, by assessing user’s requirements and then providing them information in time limit is important aspect. In real-time database systems, the correctness of database transaction depends not only on its consistency constraints and to produce correct results but also on the time limit at which a transaction is completed [1].
The disk scheduling problem involves reordering the disk requests in the disk queue so that the disk requests will be serviced with the minimum mechanical motion by employing seek optimization and latency optimization. In the disk scheduling problem we have to consider these parameters. Release time the earliest time at which the task can be started. Deadline the latest time by which the transaction must be completed. Start-time the time at which the transaction is actually started. Fulfill-time the time at which the transaction is actually completed [1][2].

One of the well-known conventional algorithm is SCAN, which scans disk surface back and forth to retrieve the data under disk head until there are no requests. This algorithm has been proved as the best algorithm for maximizing disk throughput, but output result may not meet timing constraint. Secondly, if request arrive to the one end of the disk and just miss arm scan then it may have to wait for a long time before arm scans back. One of the best-known algorithms for scheduling real-time transactions, which serves the transaction in the increasing order of their deadline, is Earliest-Deadline-First (EDF). But EDF results poor performance under overloaded condition and incurs excessive seek-time costs [3][4].

Many algorithms were proposed which combines the features of SCAN type of seek-optimizing algorithms with EDF type of real-time scheduling algorithms. In 1993, Reddy and Wyllie proposed the SCAN-EDF method that first sorts input transactions by the EDF order and, then reschedules transactions with the same deadlines by SCAN. This algorithm resolves only the scan direction for the transactions that have the same deadline. To increase the probability of employing SCAN to reschedule transactions, locally seek-optimizing scheme; i.e., transactions can only be rescheduled by SCAN within a local group DM-SCAN (deadline-modification-SCAN) is proposed to select automatically contiguous transactions that can be rescheduled by SCAN. But in DM-SCAN, transaction cannot be rescheduled to a different group once it belongs to a particular group, even though such a rescheduling derives a better performance [5] [6].

To resolve the drawback of previous approaches, globally seek-optimizing scheduling approach: Two-Way Scan was introduced where local groups are merged depending upon block locations of transactions in consecutive scan groups. We extent Two-Way Scan by assigning priority value to the transactions depending upon seek time to minimize response time. Transactions are served only if they are feasible. Experimental results show that our new approach gives better results than Two-Way Scan algorithm [7].

To illustrate the performance of EDF, DM-SCAN, Two-Way SCAN and our new approach SST following tables 1.1 Parameter Calculations and 1.2 Service Table are used.

Initial disk head location is block 5.
TABLE 1.1 Parameter Calculations

<table>
<thead>
<tr>
<th>Tid</th>
<th>At</th>
<th>Tp</th>
<th>Bs</th>
<th>Sb</th>
<th>Eb</th>
<th>AET</th>
<th>Di</th>
<th>Tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>0</td>
<td>R</td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>3</td>
<td>6</td>
<td>1.2</td>
</tr>
<tr>
<td>T1</td>
<td>2</td>
<td>W</td>
<td>2</td>
<td>17</td>
<td>18</td>
<td>3</td>
<td>8</td>
<td>1.2</td>
</tr>
<tr>
<td>T2</td>
<td>1</td>
<td>W</td>
<td>3</td>
<td>12</td>
<td>14</td>
<td>4.5</td>
<td>10</td>
<td>1.8</td>
</tr>
<tr>
<td>T3</td>
<td>3</td>
<td>R</td>
<td>3</td>
<td>6</td>
<td>8</td>
<td>4.5</td>
<td>12</td>
<td>1.8</td>
</tr>
<tr>
<td>T4</td>
<td>2</td>
<td>R</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>14</td>
<td>2.4</td>
</tr>
<tr>
<td>T5</td>
<td>1</td>
<td>W</td>
<td>4</td>
<td>15</td>
<td>18</td>
<td>6</td>
<td>13</td>
<td>2.4</td>
</tr>
<tr>
<td>T6</td>
<td>6</td>
<td>R</td>
<td>3</td>
<td>13</td>
<td>15</td>
<td>4.5</td>
<td>15</td>
<td>1.8</td>
</tr>
<tr>
<td>T7</td>
<td>2.5</td>
<td>R</td>
<td>5</td>
<td>11</td>
<td>15</td>
<td>7.5</td>
<td>17.5</td>
<td>3</td>
</tr>
<tr>
<td>T8</td>
<td>9</td>
<td>W</td>
<td>5</td>
<td>16</td>
<td>20</td>
<td>7.5</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>T9</td>
<td>4</td>
<td>R</td>
<td>7</td>
<td>9</td>
<td>15</td>
<td>10.5</td>
<td>25</td>
<td>4.2</td>
</tr>
</tbody>
</table>

TABLE 1.2 Service Table

<table>
<thead>
<tr>
<th>Cji</th>
<th>T0</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>0</td>
<td>3</td>
<td>2.1</td>
<td>3.3</td>
<td>4.5</td>
<td>3.6</td>
<td>2.4</td>
<td>3</td>
<td>4.5</td>
<td>4.8</td>
</tr>
<tr>
<td>T1</td>
<td>3.6</td>
<td>0</td>
<td>3.6</td>
<td>5.4</td>
<td>6.6</td>
<td>3.9</td>
<td>3.3</td>
<td>5.1</td>
<td>3.6</td>
<td>6.9</td>
</tr>
<tr>
<td>T2</td>
<td>2.4</td>
<td>2.1</td>
<td>0</td>
<td>4.2</td>
<td>5.4</td>
<td>2.7</td>
<td>2.1</td>
<td>3.9</td>
<td>3.6</td>
<td>5.7</td>
</tr>
<tr>
<td>T3</td>
<td>1.8</td>
<td>4.5</td>
<td>3</td>
<td>0</td>
<td>3.6</td>
<td>4.5</td>
<td>3.3</td>
<td>3.9</td>
<td>5.4</td>
<td>4.5</td>
</tr>
<tr>
<td>T4</td>
<td>2.1</td>
<td>4.2</td>
<td>3.3</td>
<td>2.1</td>
<td>0</td>
<td>4.8</td>
<td>3.6</td>
<td>4.2</td>
<td>5.7</td>
<td>4.8</td>
</tr>
<tr>
<td>T5</td>
<td>3.6</td>
<td>1.5</td>
<td>3.6</td>
<td>5.4</td>
<td>6.6</td>
<td>0</td>
<td>3.3</td>
<td>5.1</td>
<td>3.6</td>
<td>6.9</td>
</tr>
<tr>
<td>T6</td>
<td>2.7</td>
<td>1.8</td>
<td>2.7</td>
<td>4.5</td>
<td>5.7</td>
<td>2.4</td>
<td>0</td>
<td>3</td>
<td>3.3</td>
<td>6</td>
</tr>
<tr>
<td>T7</td>
<td>2.7</td>
<td>1.8</td>
<td>2.7</td>
<td>4.5</td>
<td>5.7</td>
<td>2.4</td>
<td>2.4</td>
<td>0</td>
<td>3.3</td>
<td>6</td>
</tr>
<tr>
<td>T8</td>
<td>4.2</td>
<td>2.1</td>
<td>4.2</td>
<td>6</td>
<td>7.2</td>
<td>3.9</td>
<td>3.9</td>
<td>5.7</td>
<td>0</td>
<td>7.5</td>
</tr>
<tr>
<td>T9</td>
<td>2.7</td>
<td>1.8</td>
<td>2.7</td>
<td>4.5</td>
<td>5.7</td>
<td>2.4</td>
<td>2.4</td>
<td>4.2</td>
<td>3.3</td>
<td>0</td>
</tr>
</tbody>
</table>

2. RELATED WORK

2.1 EDF

In 1973 Liu and Layland [3] suggested the most popular real time disk scheduling algorithm Earliest Deadline First EDF is an analog of FCFS. In EDF transactions are ordered according to deadline and the request with earliest deadline is serviced first. The EDF algorithm is good when the system is lightly loaded, but it degenerates as soon as load increases. Critical task may not get priority over non-critical tasks because the closeness of
deadline is only deciding factor. Under heavy loads, a fundamental weakness of the Earliest Deadline priority policy is that it assigns the highest priority to transactions that are close to missing their deadlines, thus delaying other transactions that might still be able to meet their deadlines. Gaining high priority at this late stage may not leave sufficient time for transactions to complete before their deadlines[7].

2.1.1 Example to illustrate the performance of EDF

EDF arranges the transactions in the increasing order of their deadlines.

Using Table 1.1 EDF schedule for our example is shown below

\[
\begin{align*}
T0 & \quad T1 & \quad T2 & \quad T3 & \quad T4 & \quad T5 & \quad T6 & \quad T7 & \quad T8 & \quad T9 \\
6 & \quad 8 & \quad 10 & \quad 12 & \quad 13 & \quad 14 & \quad 15 & \quad 17.5 & \quad 24 & \quad 25
\end{align*}
\]

----------> Deadlines

Using Table 1.2 timing diagram for the above EDF schedule is

![Timing Diagram for EDF Schedule](image)

Figure 2.1 Timing Diagram for EDF Schedule

2.2 DM-SCAN

In 1998 [5] R. 1.Chang, W K. Shih, and R. C. Chang proposed Deadline-Modification-SCAN (DM-SCAN) that suggests the use of maximum-scannable-groups compute the suitable request group for seek-optimizing with guaranteed real-time requirements for rescheduling. The DM-SCAN identifies and reschedules the maximum-scannable-group repeatedly. In this algorithm, request deadlines are reduced several times during the process of rescheduling to preserve EDF schedule.

To increase the probability of employing the SCAN scheme to reschedule input transactions, DM-SCAN proposed the concept of maximum-scannable-group (MSG). Given an EDF schedule, consecutive transactions that can be rescheduled by SCAN without missing their respective timing constraints can be directly derived by the concept of MSG. Given a set of real-time disk transactions with EDF-ordered \( T = T_1T_2 \ldots T_n \), the MSG \( G_i \) starting from \( T_i \) is defined as the sequential transactions \( G_i = T_iT_{i+1}T_{i+2} \ldots T_{i+m} \) with each task \( T_k \) for \( k = i \)
to \(i + m\) satisfies \(f_k \leq d_i\) and \(r_k \leq s_i\). However, DM-SCAN requires that the input transactions must be EDF-ordered. Therefore, they proposed a deadline modification scheme that transfers a non-EDF schedule into an EDF order by modifying transactions deadlines [4].

A MSG group is said to be a scanned MSG (S.MSG) if the requests in this group are seek-optimized. Otherwise, it is called an un-scanned MSG (U.MSG). Notably, only U.MSG is necessary to be rescheduled by SCAN. Note that these U.MSG groups are not mutually exclusive in nature. For example, we may have \(G_i = T_iT_{i+1}...T_jT_{j+1}...T_{i+p}\) and \(G_j = T_jT_{j+1}...T_{i+p}...T_{i+q}\). It is easy to show that these two MSG groups are not mutually exclusive (\(G_i \cap G_j = \{T_j,T_{j+1}, ..., T_{i+p}\} \neq \emptyset\)) and cannot be rescheduled at the same time. In this approach, a procedure based on the first-come-first-serve (FCFS) policy is applied to select a set of mutually-exclusive U.MSG (MU.MSG, for short) groups for rescheduling [5][6].

For DM-SCAN the input schedule should be in EDF order. If the input schedule is not in EDF order modify the deadlines and make them in EDF order. In our example the following schedule is not in EDF order.

\[
T_0 \ T_1 \ T_2 \ T_3 \ T_4 \ T_5 \ T_6 \ T_7 \ T_8 \ T_9 \ \\
6 \ \ 8 \ \ 10 \ 12 \ 14 \ \ 13 \ \ 15 \ \ 17.5 \ \ 24 \ \ 25 \ \\
\text{Deadline}
\]

Modify the above schedule in EDF order using DM-SCAN algorithm as follows:

As shown in the Figure 2.2 deadline of \(T_4\) is 14 and deadline of \(T_3\) is 13, using above method deadline of \(T_4\) is modified to 13.

After modifying the deadlines apply MSG algorithm to form MSG groups. Groups formed after applying MSG algorithm are:

![Figure 2.2: Deadline Modification of EDF schedule](image-url)
G0 = {T0}
G1 = {T1}
G2 = {T3, T2}
G4 = {T4, T3, T2}
G5 = {T5, T4, T3, T2}
G6 = {T6, T5, T4, T3}
G7 = {T7, T6, T5, T4}
G8 = {T8, T7, T6, T5}
G9 = {T9, T8, T7, T6, T5}

After applying MU_MSG algorithm on the above groups the final groups form are:
G0 = {T0}
G1 = {T1}
G4 = {T4, T3, T2}
G9 = {T9, T8, T7, T6, T5}

Apply SCAN algorithm on each group as shown in the following Figure 2.3.

![Figure 2.3: SCAN Algorithm on MU_MSG Groups](image)

Final schedule for DM-SCAN is: T0 T1 T2 T3 T4 T9 T7 T6 T9 T8

Using Table 1.2 Timing Diagram for DM-SCAN schedule is:

![Figure 2.4: Timing Diagram for DM-SCAN schedule](image)

### 2.3 Two-Way-Scan

To resolve the drawback of local groups, globally seek-optimizing scheduling approach: Two-Way Scan was proposed in 2009 by Myung Sub Lee where local groups are merged depending upon block location of transactions in consecutive scan groups. Two way scan composed of three steps [6][7]:

- Step 1: 
- Step 2: 
- Step 3:
1) **Insertion technique** that can insert inconsecutive requests into proper scan groups called as maximum scanable groups (MSG) when deciding scan groups. Here also we formed MU-MSGs using MU-MSG algorithm. These MSGs and MU-MSGs are formed using same technique as DM-SCAN algorithm.

2) **Scan group merge technique** that can merge consecutive scan groups. If block location of first transaction in next group is less than block location of last transaction in previous group then these two groups are merged to form a new group.

3) **Two-way scan technique** that can decide the direction of scan in an effective way.

2.3.1 Example to illustrate the performance of two-way Scan

**Step I: Insertion Technique**

Schedule of our example is T0 T1 T2 T3 T4 T5 T6 T7 T8 T9

Apply MSG and MU_MSG algorithms on our example schedule to form MSGs and MU_MSGs.

<table>
<thead>
<tr>
<th>MSG</th>
<th>MU_MSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0 = {T0}</td>
<td>G0 = {T0}</td>
</tr>
<tr>
<td>G1 = {T1}</td>
<td>G1 = {T1}</td>
</tr>
<tr>
<td>G2 = {T3, T2}</td>
<td>G4 = {T4, T3, T2}</td>
</tr>
<tr>
<td>G4 = {T4, T3, T2}</td>
<td>G9 = {T9, T8, T7, T6, T5}</td>
</tr>
<tr>
<td>G5 = {T5, T4, T3, T2}</td>
<td></td>
</tr>
<tr>
<td>G6 = {T6, T5, T4, T3}</td>
<td></td>
</tr>
<tr>
<td>G7 = {T7, T6, T5, T4}</td>
<td></td>
</tr>
<tr>
<td>G8 = {T8, T7, T6, T5}</td>
<td></td>
</tr>
<tr>
<td>G9 = {T9, T8, T7, T6, T5}</td>
<td></td>
</tr>
</tbody>
</table>

**Step II: Scan Group Merge Technique**

Consecutive Scan Groups are merged by using the following condition

\[
\begin{align*}
\text{Block location of first transaction in next group} & < \text{Block location of last transaction in previous group}
\end{align*}
\]

After merging consecutive scan groups final groups formed are:

G0 = {T0} and Gnew2 = {T1, T4, T3, T2, T9, T8, T7, T6, T5}

**Step III: Two-Way Scan Technique.**

Using Figure 5.11 after applying Two-way Scan technique on G0 and Gnew2 we get the final schedule as follows:
3. **PROPOSED APPROACH**

Many approaches were proposed which combines the features of SCAN type of seek-optimizing algorithms with EDF type of real-time scheduling algorithms like SCAN-EDF and DM-SCAN. In SCAN-EDF, group is made up of transactions having the same deadline. Similarly DM-SCAN automatically selects groups of consecutive transactions and these groups are named as MSGs. However in SCAN-EDF and DM-SCAN once a transaction belongs to a certain group, it cannot be rescheduled to a different group. This type of scheduling is called locally seek-optimizing schemes i.e. transactions can be reschedule by SCAN within a local group.

To resolve the drawback of previous approaches, globally seek-optimizing scheduling approach: Two-Way Scan was introduced where local groups are merged depending upon block locations of transactions in consecutive scan groups.

In our approach after finding MSG and MU_MSG scan group merge technique is applied to merge the groups. We then assign priority to each transaction and transaction having lowest priority value is served first. Consecutive groups are served one after the other. Transactions are served only if they are feasible.

There are three steps in our approach:

**Step I: Insertion technique** that can insert inconsecutive requests into proper scan groups called as maximum scanable groups(MSG) when deciding scan groups. MU-MSGs are formed using MU-MSG algorithm. These MSGs and MU-MSGs are formed using same technique as DM-SCAN algorithm.

**Step II: Scan group merge technique** that can merge consecutive scan groups. Same as Two-Way Scan approach.
Step III: Shortest Seek Time approach here priorities is assign to each transaction in each group and serve the transactions according to their priority value. Transactions are served only if they are feasible.

Following steps are used to assign priorities to the transactions [9]:

1) Assign weight to each transaction (Ti) using the following equation
   \[ W_i = \beta^{i-1} \]
   where \( \beta \geq 1 \)

   We assume \( \beta = 2 \),

2) Find the distance ‘di’ from current disk location for each transaction in a group using the following equation. Here we calculate distance ‘di’ in terms of seek time.

   Seek Time = abs(current disk head – start block of Ti) * 0.3

3) Calculate priority value P.V.

   \[ P.V. = W_i \times d_i \]

   Now, serve the transactions according to their priority values. The transaction having minimum priority value will be served first [8].

   If two transactions have same priority value then transactions having less Release Time will be served first. Transactions are served only if they are feasible.

3.1 Example to illustrate the performance of our new approach (SST)

Step I: Insertion technique

Apply MSG and MU_MSG algorithms on our example schedule to form MSGs and MU_MSGs.

<table>
<thead>
<tr>
<th>MSG</th>
<th>MU_MSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0 = {T0}</td>
<td>G0 = {T0}</td>
</tr>
<tr>
<td>G1 = {T1}</td>
<td>G1 = {T1}</td>
</tr>
<tr>
<td>G2 = {T3, T2}</td>
<td>G4 = {T4, T3, T2}</td>
</tr>
<tr>
<td>G4 = {T4, T3, T2}</td>
<td>G9 = {T9, T8, T7, T6, T5}</td>
</tr>
<tr>
<td>G5 = {T5, T4, T3, T2}</td>
<td></td>
</tr>
<tr>
<td>G6 = {T6, T5, T4, T3}</td>
<td></td>
</tr>
<tr>
<td>G7 = {T7, T6, T5, T4}</td>
<td></td>
</tr>
<tr>
<td>G8 = {T8, T7, T6, T5}</td>
<td></td>
</tr>
<tr>
<td>G9 = {T9, T8, T7, T6, T5}</td>
<td></td>
</tr>
</tbody>
</table>

Step II: Scan Group Merge Technique

This step is same as Two-Way Scan algorithm. Group forms after merging are:

G0 = {T0}
Gnew2 = {T1, T4, T3, T2, T9, T8, T7, T6, T5}
Step III: Shortest Seek Time approach
Assign weight $W_i$ to each transaction ($T_i$)

- $T_0 \rightarrow W_0 = \beta^{0-1} = 2^{-1} = 0.5$
- $T_1 \rightarrow W_1 = \beta^{1-1} = 2^0 = 1$
- $T_2 \rightarrow W_2 = \beta^{2-1} = 2^1 = 2$
- $T_3 \rightarrow W_3 = \beta^{3-1} = 2^2 = 4$
- $T_4 \rightarrow W_4 = \beta^{4-1} = 2^3 = 8$
- $T_5 \rightarrow W_5 = \beta^{5-1} = 2^4 = 16$
- $T_6 \rightarrow W_6 = \beta^{6-1} = 2^5 = 32$
- $T_7 \rightarrow W_7 = \beta^{7-1} = 2^6 = 64$
- $T_8 \rightarrow W_8 = \beta^{8-1} = 2^7 = 128$
- $T_9 \rightarrow W_9 = \beta^{9-1} = 2^8 = 256$

1) Find the distance ‘$d_i$’ from current disk head position for each transaction in a group
   For Group $G_0$
   - There is only one transaction $T_0$. After servicing $T_0$
     - $T_0 \rightarrow d_0 = 1.5$
     - Current head position = End block of $T_0 = 11$
   For Group $G_{new2}$
     - $T_1 \rightarrow d_1 = (11 - 17) * 0.3 = 1.8$
     - $T_2 \rightarrow d_2 = (11 - 12) * 0.3 = 0.3$
     - $T_3 \rightarrow d_3 = (11 - 6) * 0.3 = 1.5$
     - $T_4 \rightarrow d_4 = (11 - 4) * 0.3 = 2.1$
     - $T_5 \rightarrow d_5 = (11 - 15) * 0.3 = 1.2$
     - $T_6 \rightarrow d_6 = (11 - 13) * 0.3 = 0.6$
     - $T_7 \rightarrow d_7 = (11 - 11) * 0.3 = 0$
     - $T_8 \rightarrow d_8 = (11 - 6) * 0.3 = 1.5$
     - $T_9 \rightarrow d_9 = (11 - 9) * 0.3 = 0.6$

2) Calculate priority value P.V. for each transactions
   \[ P.V. = W_i * d_i \]
   - $T_1 \rightarrow W_1 * d_1 = 1.8$
   - $T_2 \rightarrow W_2 * d_2 = 0.6$
   - $T_3 \rightarrow W_3 * d_3 = 6.0$
   - $T_4 \rightarrow W_4 * d_4 = 16.8$
   - $T_5 \rightarrow W_5 * d_5 = 19.2$
   - $T_6 \rightarrow W_6 * d_6 = 19.2$
   - $T_7 \rightarrow W_7 * d_7 = 0$
   - $T_8 \rightarrow W_8 * d_8 = 192$
   - $T_9 \rightarrow W_9 * d_9 = 153.6$
Serve the transactions according to their priority value. The transaction having minimum priority value will be served first.

If two transactions have priority values then check their Release Time. The transaction having less Release Time will be served first. In our example transactions T5 and T6 are having same priority value but, the release of T5 is 1 and T6 is 6. So the final schedule will be

T0 T7 T2 T1 T3 T4 T5 T6 T9 T8

Now check the feasibility of all the transactions in the schedule using following procedure.

\[
\text{Access}(n) = \text{Total Transmission Time} \\
\text{If } (\text{current time} + \text{Access}(n)) \leq \text{Deadline } \text{ then Feasible} \\
\text{else Not Feasible}
\]

Feasible Transactions are: T0 T7 T2 T5 T6 T9 T8
Not Feasible Transactions are: T1 T3 T4

Using Table 1.2 Timing Diagram for Shortest Seek Time Schedule is:

![Timing Diagram](image)

Figure 2.6: Timing Diagram for Shortest Seek Time schedule

### 3.2 Performance Graph

The following performance graph for the above example shows that in our proposed approach number of hit transactions is more than other approaches.
4. CONCLUSION

Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector. Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head. Transmission time is the time the disk page must be made to spin by the disk head. Lot of research is going on to minimize seek time.

In this paper, various seek optimizing real time disk scheduling algorithms were investigated and algorithms like Earliest Deadline First (EDF), Deadline-Modification Scan (DM-SCAN) and Two-way scan were simulated. A new approach is proposed where after merging the groups the priority value is assign depending upon seek time to all the transactions in each group and transactions are arranged as per priority values in each group. Transactions are served only if they are feasible. Experimental results after simulation shows that our new approach performs better than previous approaches under different work load conditions.

REFERENCES


