On the Design of Contention Managers for Real-Time Software Transactional Memory

Mohammed Elshambakey

Preliminary Examination Proposal submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

April 10, 2012
Blacksburg, Virginia

Keywords: Software Transactional Memory, Contention Manager

On the Design of Contention Managers for Real-Time Software Transactional Memory

Mohammed Elshambakey

(ABSTRACT)

NEEDS TO BE WRITTEN

Contents

List of Figures

List of Tables

Chapter 1

Introduction

NEEDS TO BE WRITTEN

Table 1.1: The Graduate School wants captions above the tables.

x	1	2
1	1	2
2	2	4

Chapter 2

ECM and RCM

We consider software transactional memory (STM) for concurrency control in multicore embedded real-time software. We investigate real-time contention managers (CMs) for resolving transactional conflicts, including those based on dynamic and fixed priorities, and establish upper bounds on transactional retries and task response times. We identify the conditions under which STM (with the proposed CMs) is superior to lock-free synchronization.

2.1 G-EDF/EDF CM Response Time

Since only one atomic section among many that share the same object can commit at any time under STM, those atomic sections execute in sequential order. A task τ_i's atomic sections are interfered by other tasks that share the same objects with τ_i. The EDF CM will abort and retry an atomic section of τ_i, $s_i^k(\theta)$ due to a conflicting atomic section of τ_j, $s_j^l(\theta)$, if the absolute deadline of τ_j is less than or equal to the absolute deadline of τ_i. Hereafter, we will use *ECM* to refer to a multiprocessor system scheduled by G-EDF and resolves STM conflicts using the EDF CM.

The maximum number of times a task τ_j interferes with τ_i is given in [**?**] and is illustrated in Figure 2.1. Here, the deadline of an instance of τ_j coincides with that of τ_i, and τ_j^1 is delayed by its maximum jitter J_j, which causes all or part of τ_j^1's execution to overlap within T_i. From Figure 2.1, it is seen that τ_j's maximum workload that interferes with τ_i (when there are no atomic sections) in T_i is:

$$\begin{aligned} W_{ij}\left(T_i\right) &\le \left\lfloor \frac{T_i}{T_j} \right\rfloor c_j + min\left(c_j, T_i - \left\lfloor \frac{T_i}{T_j} \right\rfloor T_j\right) \\ &\le \left\lceil \frac{T_i}{T_j} \right\rceil c_j \qquad (2.1) \end{aligned}$$

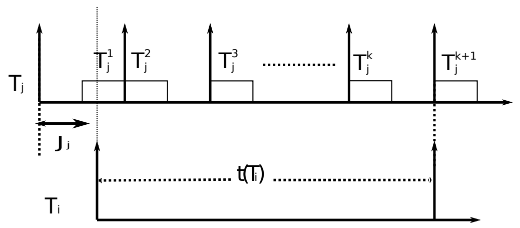


Figure 2.1: Maximum interference between two tasks, running on different processors, under G-EDF

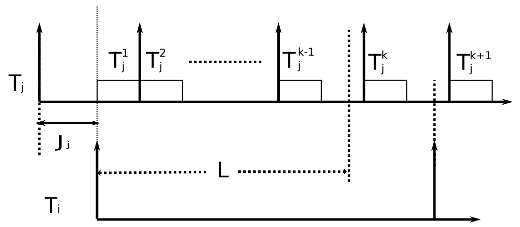


Figure 2.2: Maximum interference during an interval L of T_i

For an interval $L < T_i$, the worst case pattern of interference is shown in Figure 2.2. Here, τ_j^1 contributes by all its c_j, and d_j^{k-1} does not have to coincide with L, as τ_j^{k-1} has a higher priority than that of τ_i. The workload of τ_j is:

$$W_{ij}\left(L\right) \leq \left(\left\lceil \frac{L - c_j}{T_j} \right\rceil + 1\right) c_j \tag{2.2}$$

Thus, the overall workload, over an interval R is:

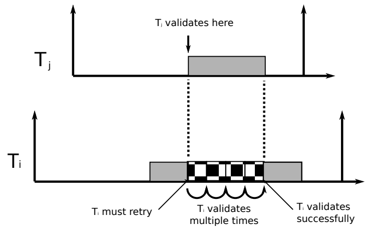
$$W_{ij}\left(R\right) = min\left(W_{ij}\left(R\right), W_{ij}\left(T_i\right)\right) \tag{2.3}$$

where $W_{ij}(R)$ is calculated by (2.2) if $R < T_i$, otherwise, it is calculated by (2.1).

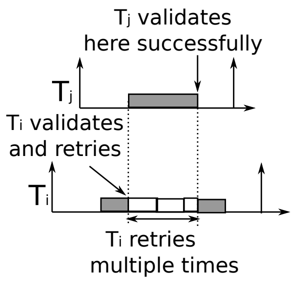
2.1.1 Retry Cost of Atomic Sections

Claim 1. *Under ECM, a task T_i's maximum retry cost during T_i is upper bounded by:*

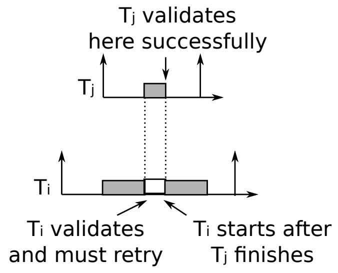
$$\begin{aligned} RC\left(T_i\right) \;\; \leq & \;\; \sum_{\theta \in \theta_i} \Bigg(\bigg(\sum_{\tau_j \in \gamma_i(\theta)} \bigg(\left\lceil \frac{T_i}{T_j} \right\rceil \sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta) \\ & + \;\; s_{max}(\theta)\big)\bigg)\bigg) - s_{max}(\theta) + s_{i_{max}}(\theta) \Bigg) \end{aligned} \tag{2.4}$$



(a) Early validation



(b) Lazy validation with $len(s_i^k(\theta)) \leq len(s_j^l(\theta))$


Tj validates here successfully
Tj
Ti
Ti validates and must retry
Ti starts after Tj finishes

(c) Lazy validation with $len(s_i^k(\theta)) > len(s_j^l(\theta))$

Figure 2.3: Retry of $s_i^k(\theta)$ due to $s_j^l(\theta)$

Proof. Consider two instances τ_i^a and τ_j^b, where $d_j^b \leq d_i^a$. When a shared object conflict occurs, the EDF CM will commit the atomic section of τ_j^b while aborting and retrying that of τ_i^a. Thus, an atomic section of τ_i^a, $s_i^k(\theta)$, will experience its maximum delay when it is at the end of its atomic section, and the conflicting atomic section of τ_j^b, $s_j^l(\theta)$, starts, because the whole $s_i^k(\theta)$ will be repeated after $s_j^l(\theta)$.

Validation (i.e., conflict detection) in STM is usually done in two ways [?]: a) eager (pessimistic), in which conflicts are detected at access time, and b) lazy (optimistic), in which conflicts are detected at commit time. Despite the validation time incurred (either eager or lazy), $s_i^k(\theta)$ will retry for the same time duration, which is $len(s_j^l(\theta)+s_i^k(\theta))$. Then, $s_i^k(\theta)$ can commit successfully unless it is interfered by another conflicting atomic section, as shown in Figure 2.3.

In Figure 2.3(a), $s_j^l(\theta)$ validates at its beginning, due to early validation, and a conflict is detected. So τ_i^a retries multiple times (because at the start of each retry, τ_i^a validates) during the execution of $s_j^l(\theta)$. When τ_j^b finishes its atomic section, τ_i^a executes its atomic section.

In Figure 2.3(b), τ_i^a validates at its end (due to lazy validation), and detects a conflict with τ_j^b. Thus, it retries, and because its atomic section length is shorter than that of τ_j^b, it validates again within the execution interval of $s_j^l(\theta)$. However, the EDF CM retries it again. This process continues until τ_j^b finishes its atomic section. If τ_i^a's atomic section length is longer than that of τ_j^b, τ_i^a would have incurred the same retry time, because τ_j^b will validate when τ_i^a is retrying, and τ_i^a will retry again, as shown in Figure 2.3(c). Thus, the retry cost of $s_i^k(\theta)$ is $len(s_i^k(\theta)+s_j^l(\theta))$.

If multiple tasks interfere with τ_i^a or interfere with each other and τ_i^a (see the two interference examples in Figure 2.4), then, in each case, each atomic section of the shorter deadline tasks contributes to the delay of $s_i^p(\theta)$ by its total length, plus a retry of some atomic section in the longer deadline tasks. For example, $s_j^l(\theta)$ contributes by $len(s_j^l(\theta)+s_i^p(\theta))$ in both Figures 2.4(a) and 2.4(b). In Figure 2.4(b), $s_k^y(\theta)$ causes a retry to $s_j^l(\theta)$, and $s_h^w(\theta)$ causes a retry to $s_k^y(\theta)$.

Since we do not know in advance which atomic section will be retried due to another, we can safely assume that, each atomic section (that shares the same object with τ_i^a) in a shorter deadline task contributes by its total length, in addition to the maximum length between all atomic sections that share the same object, $len(s_{max}(\theta))$. Thus,

$$W_i^p\left(s_j^k\left(\theta\right)\right) \le len\left(s_j^k\left(\theta\right)+s_{max}\left(\theta\right)\right) \tag{2.5}$$

Thus, the total contribution of all atomic sections of all other tasks that share objects with a task τ_i to the retry cost of τ_i during T_i is:

$$\begin{aligned} RC\left(T_i\right) \;\le\; & \sum_{\theta\in\theta_i}\sum_{\tau_j\in\gamma_i(\theta)}\Bigg(\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta) \\ & + \; s_{max}(\theta)\big)\Bigg) \end{aligned} \tag{2.6}$$

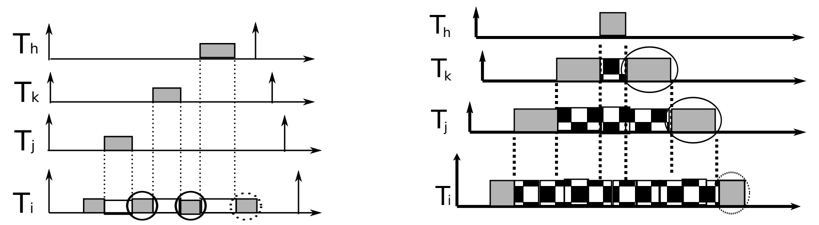
Here, $\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\left(s_j^l\left(\theta\right)+s_{max}\left(\theta\right)\right)$ is the contribution of all instances of τ_j during T_i. This contribution is added to all tasks. The last atomic section to execute is $s_i^p(\theta)$ (τ_i's atomic section that was delayed by conflicting atomic sections of other tasks). One of the other atomic sections (e.g., $s_m^n(\theta)$) should have a contribution $len(s_m^n(\theta)+s_{i_{max}}(\theta))$, instead of $len(s_m^n(\theta)+s_{max}(\theta))$. That is why one $s_{max}(\theta)$ should be subtracted, and $s_{i_{max}}(\theta)$ should be added (i.e., $s_{i_{max}}(\theta)-s_{max}(\theta)$). Claim follows. □

Claim 2. *Claim 1's retry bound can be minimized as:*

$$RC(T_i) \le \sum_{\theta\in\theta_i} min(\Phi_1,\Phi_2) \tag{2.7}$$

where Φ_1 is calculated by (2.4) for one object θ (not the sum of objects in θ_i), and

$$\begin{aligned} \Phi_2 \;=\; & \Bigg(\sum_{\tau_j\in\gamma_i(\theta)}\Bigg(\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta) \\ & +s_{max}^*(\theta)\big)\Bigg)\Bigg) - \bar{s}_{max}(\theta) + s_{i_{max}}(\theta) \end{aligned} \tag{2.8}$$



(a) Other atomic sections interfere only with $s_i^p(\theta)$

(b) All atomic sections interfere with each other and $s_i^p(\theta)$

○ Replaced in calculations by $s_{max}(\theta)$

◌ Replaced in calculations by $s_{i_{max}}(\theta)$

Figure 2.4: Retry of $s_i^p(\theta)$ due to other atomic sections

where s^*_{max} is the maximum atomic section between all tasks, except τ_j, accessing θ. $\bar{s}_{max}(\theta)$ is the second maximum atomic section between all tasks accessing θ.

Proof. (2.4) can be modified by noting that a task τ_j's atomic section may conflict with those of other tasks, but not with τ_j. This is because, tasks are assumed to arrive sporadically, and each instance finishes before the next begins. Thus, (2.5) becomes:

$$W_i^p\left(s_j^k(\theta)\right) \leq len\left(s_j^k(\theta) + s^*_{max}(\theta)\right) \tag{2.9}$$

To see why $\bar{s}_{max}(\theta)$ is used instead of $s_{max}(\theta)$, the maximum-length atomic section of each task that accesses θ is grouped into an array, in non-increasing order of their lengths. $s_{max}(\theta)$ will be the first element of this array, and $\bar{s}_{max}(\theta)$ will be the next element, as illustrated in Figure 2.5, where the maximum atomic section of each task that accesses θ is associated with its corresponding task. According to (2.9), all tasks but τ_j will choose $s_{j_{max}}(\theta)$ as the value of $s^*_{max}(\theta)$. But when τ_j is the one whose contribution is studied, it will choose $s_{k_{max}}(\theta)$, as it is the maximum one not associated with τ_j. This way, it can be seen that the maximum value always lies between the two values $s_{jmax}(\theta)$ and $s_{kmax}(\theta)$. Of course, these two values can be equal, or the maximum value can be associated with τ_i itself, and not with any one of the interfering tasks. In the latter case, the chosen value will always be the one associated with τ_i, which still lies between the two largest values.

This means that the subtracted $s_{max}(\theta)$ in (2.4) must be replaced with one of these two values ($s_{max}(\theta)$ or $\bar{s}_{max}(\theta)$). However, since we do not know which task will interfere with τ_i, the minimum is chosen, as we are determining the worst case retry cost (as this value is going to be subtracted), and this minimum is the second maximum.

Since it is not known a-priori whether Φ_1 will be smaller than Φ_2 for a specific θ, the minimum of Φ_1 and Φ_2 is taken as the worst-case contribution for θ in $RC(T_i)$. □

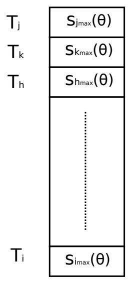


Figure 2.5: Values associated with $s^*_{max}(\theta)$

2.1.2 Upper Bound on Response Time

To obtain an upper bound on the response time of a task τ_i, the term $RC(T_i)$ must be added to the workload of other tasks during the non-atomic execution of τ_i. But this requires modification of the WCET of each task as follows. c_j of each interfering task τ_j should be inflated to accommodate the interference of each task τ_k, $k \neq j, i$. Meanwhile, atomic regions that access shared objects between τ_j and τ_i should not be considered in the inflation cost, because they have already been calculated in τ_i's retry cost. Thus, τ_j's inflated WCET becomes:

$$c_{ji} = c_j - \left(\sum_{\theta \in (\theta_j \wedge \theta_i)} len\left(s_j(\theta)\right) \right) + RC(T_{ji}) \tag{2.10}$$

where, c_{ji} is the new WCET of τ_j relative to τ_i; the sum of lengths of all atomic sections in τ_j that access object θ is $\sum_{\theta \in (\theta_j \wedge \theta_i)} len(s_j(\theta))$; and $RC(T_{ji})$ is the $RC(T_j)$ without including the shared objects between τ_i and τ_j. The calculated WCET is relative to task τ_i, as it changes from task to task. The upper bound on the response time of τ_i, denoted R_i^{up}, can be calculated iteratively, using a modification of Theorem 6 in [?], as follows:

$$R_i^{up} = c_i + RC(T_i) + \left\lfloor \frac{1}{m} \sum_{j \neq i} W_{ij}(R_i^{up}) \right\rfloor \tag{2.11}$$

where R_i^{up}'s initial value is $c_i + RC(T_i)$.

$W_{ij}(R_i^{up})$ is calculated by (2.3), and $W_{ij}(T_i)$ is calculated by (2.1), with c_j replaced by c_{ji}, and changing (2.2) as:

$$W_{ij}(L) = max \begin{cases} \left(\left\lceil \frac{L - \left(c_{ji} + \sum_{\theta \in (\theta_j \wedge \theta_i)} len(s_j(\theta))\right)}{T_j} \right\rceil + 1 \right) c_{ji} \\ \left\lceil \frac{L - c_j}{T_j} \right\rceil .c_{ji} + c_j - \sum_{\theta \in (\theta_j \wedge \theta_i)} len(s_j(\theta)) \end{cases} \tag{2.12}$$

(2.12) compares two terms, as we have two cases:

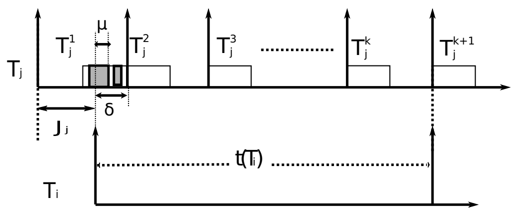


Figure 2.6: Atomic sections of job τ_j^1 contributing to period T_i

Case 1. τ_j^1 (shown in Figure 2.2) contributes by c_{ji}. Thus, other instances of τ_j will begin after this modified WCET, but the sum of the shared objects' atomic section lengths is removed from c_{ji}, causing other instances to start earlier. Thus, the term $\sum_{\theta \in (\theta_i \wedge \theta_j)} len(s_j(\theta))$ is added to c_{ji} to obtain the correct start time.

Case 2. τ_j^1 contributes by its c_j, but the sum of the shared atomic section lengths between τ_i and τ_j should be subtracted from the contribution of τ_j^1, as they are already included in the retry cost.

It should be noted that subtraction of the sum of the shared objects' atomic section lengths is done in the first case to obtain the correct start time of other instances, while in the second case, this is done to get the correct contribution of τ_j^1. The maximum is chosen from the two terms in (2.12), because they differ in the contribution of their τ_j^1s, and the number of instances after that.

2.1.2.1 Tighter Upper Bound

To tighten τ_i's response time upper bound, $RC(\tau_i)$ needs to be calculated recursively over duration R_i^{up}, and not directly over T_i, as done in (2.11). So, (2.7) must be changed to include the modified number of interfering instances. And if R_i^{up} still extends to T_i, a situation like that shown in Figure 2.6 can happen.

To counter the situation in Figure 2.6, atomic sections of τ_j^1 that are contained in the interval δ are the only ones that can contribute to $RC(T_i)$. Of course, they can be lower, but cannot be greater, because τ_j^1 has been delayed by its maximum jitter. Hence, no more atomic sections can interfere during the duration $[d_j^1 - \delta, d_j^1]$.

For simplicity, we use the following notations:

- $\lambda_1(j,\theta) = \sum_{\forall s_j^l(\theta) \in \left[d_j^1 - \delta, d_j^1\right]} len\left(s_j^{l^*}(\theta) + s_{max}(\theta)\right)$
- $\chi_1(i,j,\theta) = \left\lfloor \frac{T_i}{T_j} \right\rfloor \sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta) + s_{max}(\theta)\right)$
- $\lambda_2(j,\theta) = \sum_{\forall s_j^l(\theta) \in \left[d_j^1 - \delta, d_j^1\right]} len\left(s_j^{l^*}(\theta) + s_{max}^*(\theta)\right)$

- $\chi_2\left(i,j,\theta\right)=\left\lfloor\frac{T_i}{T_j}\right\rfloor\sum_{\forall s_j^l(\theta)}len\left(s_j^l\left(\theta\right)+s_{max}^*\left(\theta\right)\right)$

Here, $s_j^{l^*}\left(\theta\right)$ is the part of $s_j^l\left(\theta\right)$ that is included in the interval δ. Thus, if $s_j^l(\theta)$ is partially included in δ, it contributes by its included length μ.

Now, (2.7) can be modified as:

$$RC\left(T_i\right)\leq\sum_{\theta\in\theta_i}min\begin{cases}\begin{cases}\left(\left(\sum_{\tau_j\in\gamma_i(\theta)}\lambda_1\left(j,\theta\right)+\chi_1\left(i,j,\theta\right)\right)\right.\\\left.-s_{max}\left(\theta\right)+s_{i_{max}}\left(\theta\right)\right)\end{cases}\\\begin{cases}\left(\left(\sum_{\tau_j\in\gamma_i(\theta)}\lambda_2\left(j,\theta\right)+\chi_2\left(i,j,\theta\right)\right)\right.\\\left.-\bar{s}_{max}\left(\theta\right)+s_{i_{max}}\left(\theta\right)\right)\end{cases}\end{cases}\tag{2.13}$$

Now, we compute $RC(L)$, where L does not extend to the last instance of τ_j. Let:

- $\upsilon\left(L,j\right)=\left\lceil\frac{L-c_j}{T_j}\right\rceil+1$
- $\lambda_3\left(j,\theta\right)=\sum_{\forall s_j^l(\theta)}len\left(s_j^l\left(\theta\right)+s_{max}\left(\theta\right)\right)$
- $\lambda_4\left(j,\theta\right)=\sum_{\forall s_j^l(\theta)}len\left(s_j^l\left(\theta\right)+s_{max}^*\left(\theta\right)\right)$

Now, (2.7) becomes:

$$RC\left(L\right)\leq\sum_{\theta\in\theta_i}min\begin{cases}\begin{cases}\left(\sum_{\tau_j\in\gamma_i(\theta)}\left(\upsilon\left(L,j\right)\lambda_3\left(j,\theta\right)\right)\right)\\-s_{max}\left(\theta\right)+s_{i_{max}}\left(\theta\right)\end{cases}\\\begin{cases}\left(\sum_{\tau_j\in\gamma_i(\theta)}\left(\upsilon\left(L,j\right)\lambda_4\left(j,\theta\right)\right)\right)\\-\bar{s}_{max}\left(\theta\right)+s_{i_{max}}\left(\theta\right)\end{cases}\end{cases}\tag{2.14}$$

Thus, an upper bound on $RC(\tau_i)$ is given by:

$$RC(R_i^{up})\leq min\begin{cases}RC(R_i^{up})\\RC(T_i)\end{cases}\tag{2.15}$$

where $RC(R_i^{up})$ is calculated by (2.14) if R_i^{up} does not extend to the last interfering instance of τ_j; otherwise, it is calculated by (2.13). The final upper bound on τ_i's response time can be calculated as in (2.11) by replacing $RC(T_i)$ with $RC(R_i^{up})$.

2.2 G-RMA/RMA CM Response Time

As G-RMA is a fixed priority scheduler, a task τ_i will be interfered by those tasks with priorities higher than τ_i (i.e., $p_j > p_i$). Upon a conflict, the RMA CM will commit the transaction that belongs to the higher priority task. Hereafter, we use *RCM* to refer to a multiprocessor system scheduled by G-RMA and resolves STM conflicts by the RMA CM.

2.2.1 Maximum Task Interference

Figure 2.7 illustrates the maximum interference caused by a task τ_j to a task τ_i under G-RMA. As τ_j is of higher priority than τ_i, τ_j^k will interfere with τ_i even if it is not totally included in T_i. Unlike the G-EDF case shown in Figure 2.6, where only the δ part of τ_j^1 is considered, in G-RMA, τ_j^k can contribute by the whole c_j, and all atomic sections contained in τ_j^k must be considered. This is because, in G-EDF, the worst-case pattern releases τ_i^a before d_j^1 by δ time units, and τ_i^a cannot be interfered before it is released. But in G-RMA, τ_i^a is already released, and can be interfered by the whole τ_j^k, even if this makes it infeasible.

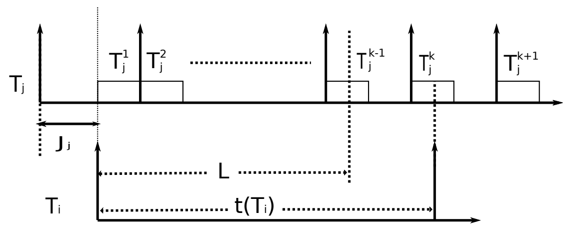


Figure 2.7: Max interference of τ_j to τ_i in G-RMA

Thus, the maximum contribution of τ_j^b to τ_i^a for any duration L can be deduced from Figure 2.7 as $W_{ij}(L) = \left(\left\lceil \frac{L-c_j}{T_j} \right\rceil + 1\right) c_j$, where L can extend to T_i. Note the contrast with ECM, where L cannot be extended directly to T_i, as this will have a different pattern of worst case interference from other tasks.

2.2.2 Retry Cost of Atomic Sections

Claim 3. *Under RCM, a task τ_i's retry cost over duration L, which can extend to T_i, is upper bounded by:*

$$\begin{aligned} RC(L) \;\le\; & \sum_{\theta \in \theta_i} \Bigg(\Bigg(\sum_{\tau_j^*} \Bigg(\Bigg(\left\lceil \frac{L - c_j}{T_j} \right\rceil + 1 \Bigg) \pi(j, \theta) \Bigg) \Bigg) \\ & - \; s_{max}^{min}(\theta) + s_{i_{max}}(\theta) \Bigg) \end{aligned} \tag{2.16}$$

where:

- $\tau_j^* = \{\tau_j | (\tau_j \in \gamma_i(\theta)) \wedge (p_j > p_i)\}$
- $\pi(j,\theta) = \sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta) + s_{max}^j(\theta)\right)$
- $s_{max}^{min}(\theta) = min_{\forall \tau_j^*}\{s_{max}^j(\theta)\}$

Proof. The worst case interference pattern for RCM is the same as that for ECM for an interval L, except that, in RCM, L can extend to the entire T_i, but in ECM, it cannot, as the interference pattern of τ_j to τ_i changes. Thus, (2.14) can be used to calculate τ_i's retry cost, with some modifications, as we do not have to obtain the minimum of the two terms in (2.14), because τ_j's atomic sections will abort and retry only atomic sections of tasks with lower priority than τ_j. Thus, $s_{max}(\theta)$, $s_{max}^*(\theta)$, and $\bar{s}_{max}(\theta)$ are replaced by $s_{max}^{min}(\theta)$, which is the minimum of the set of maximum-length atomic sections of tasks with priority lower than τ_j and share object θ with τ_i. This is because, the maximum length atomic section of tasks other than τ_j differs according to j. Besides, as τ_i's atomic sections can be aborted only by atomic sections of higher priority tasks, not all $\tau_j \in \gamma(\theta)$ are considered, but only the subset of tasks in $\gamma(\theta)$ with priority higher than τ_i (i.e., τ_j^*). Claim follows. □

2.2.3 Upper Bound on Response Time

The response time upper bound can be computed using Theorem 7 in [?] with a modification to include the effect of retry cost. The upper bound is given by:

$$R_i^{up} = c_i + RC(R_i^{up}) + \left\lfloor \frac{1}{m} \sum_{j \neq i} W_{ij}(R_i^{up}) \right\rfloor \tag{2.17}$$

where $W_{ij}(R_i^{up})$ is calculated as in (2.12), c_{ji} is calculated by (2.10), and RC is calculated by (2.16).

2.3 STM versus Lock-Free

We now would like to understand when STM will be beneficial compared to lock-free synchronization. The retry-loop lock-free approach in [?] is the most relevant to our work.

2.3.1 ECM versus Lock-Free

Claim 4. *For ECM's schedulability to be better or equal to that of [?]'s retry-loop lock-free approach, the size of s_{max} must not exceed one half of that of r_{max}, where r_{max} is the* *maximum execution cost of a single iteration of any lock-free retry loop of any task. With low number of conflicting tasks, the size of s_{max} can be at most the size of r_{max}.*

Proof. Equation (2.15) can be upper bounded as:

$$RC\left(T_i\right) \leq \sum_{\tau_j \in \gamma_i}\left(\sum_{\theta \in \theta_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil \sum_{\forall s_j^l(\theta)}\left(2.s_{max}\right)\right)\right) \tag{2.18}$$

where $s_j^l\left(\theta\right)$, $s_{i_{max}}\left(\theta\right)$, $s_{max}^*\left(\theta\right)$, and $\bar{s}_{max}\left(\theta\right)$ are replaced by s_{max}, and the order of the first two summations are reversed by each other, with γ_i being the set of tasks that share objects with τ_i. These changes are done to simplify the comparison.

Let $\sum_{\theta\in\theta_i}\sum_{\forall s_j^l(\theta)} = \beta_{i,j}^*$, and $\alpha_{edf} = \sum_{\tau_j\in\gamma_i}\left\lceil\frac{T_i}{T_j}\right\rceil .2\beta_{i,j}^*$. Now, (2.18) can be modified as:

$$RC\left(T_i\right) = \alpha_{edf}.s_{max} \tag{2.19}$$

The loop retry cost is given by:

$$\begin{aligned} LRC\left(T_i\right) &= \sum_{\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right).\beta_{i,j}.r_{max} \\ &= \alpha_{free}.r_{max} \end{aligned} \tag{2.20}$$

where $\beta_{i,j}$ is the number of retry loops of τ_j that accesses the same object as that accessed by some retry loop of τ_i, and $\alpha_{free} = \sum_{\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right).\beta_{i,j}$. Since the shared objects are the same in both STM and lock free, $\beta_{i,j} = \beta_{i,j}^*$. Thus, STM achieves equal or better schedulability than lock-free if the total utilization of the STM system is less than or equal to that of the lock-free system:

$$\begin{aligned} \sum_{\tau_i}\frac{c_i + \alpha_{edf}.s_{max}}{T_i} &\leq \sum_{\tau_i}\frac{c_i + \alpha_{free}.r_{max}}{T_i} \\ \therefore \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i}\alpha_{free}/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \end{aligned} \tag{2.21}$$

Let $\bar{\alpha}_{free} = \sum_{\tau_j\in\gamma_i}\left\lceil\frac{T_i}{T_j}\right\rceil .\beta_{i,j}$, $\hat{\alpha}_{free} = \sum_{T_j\in\gamma_i}\beta_{i,j}$, and $\alpha_{free} = \bar{\alpha}_{free} + \hat{\alpha}_{free}$. Therefore:

$$\begin{aligned} \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i}(\bar{\alpha}_{free} + \hat{\alpha}_{free})/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \\ &= \frac{1}{2} + \frac{\sum_{\tau_i}\hat{\alpha}_{free}/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \end{aligned} \tag{2.22}$$

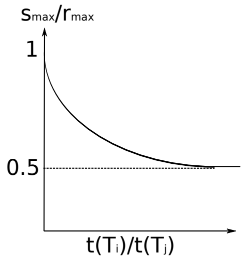


Figure 2.8: Effect of $\left\lceil \frac{T_i}{T_j} \right\rceil$ on $\frac{s_{max}}{r_{max}}$

Let $\zeta_1 = \sum_{\tau_i} \hat{\alpha}_{free}/T_i$ and $\zeta_2 = \sum_{\tau_i} \left(\frac{\alpha_{edf}}{2}\right)/T_i$. The maximum value of $\frac{\zeta_1}{2.\zeta_2} = \frac{1}{2}$, which can happen if $T_j \geq T_i$ $\therefore$ $\left\lceil \frac{T_i}{T_j} \right\rceil = 1$. Then (2.22) $= 1$, which is its maximum value. $T_j \geq T_i$ means that there is a small number of interferences from other tasks to τ_i, and thus low number of conflicts. Therefore, s_{max} is allowed to be as large as r_{max}.

The theoretical minimum value for $\frac{\zeta_1}{2.\zeta_2}$ is 0, which can be asymptotically reached if $T_j \ll T_i$, $\therefore$ $\left\lceil \frac{T_i}{T_j} \right\rceil \rightarrow \infty$ and $\zeta_2 \rightarrow \infty$. Thus, (2.22) $\rightarrow 1/2$.

$\beta_{i,j}$ has little effect on s_{max}/r_{max}, as it is contained in both the numerator and denominator. Irrespective of whether $\beta_{i,j}$ is going to reach its maximum or minimum value, both can be considered constants, and thus removed from (2.22)'s numerator and denominator. However, the number of interferences of other tasks to τ_i, $\left\lceil \frac{T_i}{T_j} \right\rceil$, has the main effect on s_{max}/r_{max}. This is illustrated in Figure 2.8. Claim follows. □

2.3.2 RCM versus Lock-Free

Claim 5. *For RCM's schedulability to be better or equal to that of [?]'s retry-loop lock-free approach, the size of s_{max} must not exceed one half of that of r_{max} for all cases. However, the size of s_{max} can be larger than that of r_{max}, depending on the number of accesses to a task T_i's shared objects from other tasks.*

Proof. Equation (2.16) is upper bounded by:

$$\sum_{(\tau_j \in \gamma_i) \wedge (p_j > p_i)} \left(\left\lceil \frac{T_i - c_j}{T_j} \right\rceil + 1 \right) .2.\beta_{i,j}.s_{max} \qquad (2.23)$$

Consider the same assumptions as in Section 2.3.1. Let $\alpha_{rma} = \sum_{(\tau_j \in \gamma_i) \wedge (p_j > p_i)} \left(\left\lceil \frac{T_i - c_j}{T_j} \right\rceil + 1 \right) .2.\beta_{i,j}$.

Now, the ratio s_{max}/r_{max} is upper bounded by:

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{T_i} \alpha_{free}/t\,(T_i)}{\sum_{T_i} \alpha_{rma}/t\,(T_i)} \tag{2.24}$$

The main difference between RCM and lock-free is that RCM is affected only by the higher priority tasks, while lock-free is affected by all tasks (just as in ECM). Besides, RCM is still affected by $2.\beta_{i,j}$ (just as in ECM). The subtraction of c_j in the numerator of (2.23) may not have a significant effect on the ratio of (2.24), as the loop retry cost can also be modified to account for the effect of the first interfering instance of task T_j. Therefore, $\alpha_{free} = \sum_{\tau_j \in \gamma_i} \left(\left\lceil \frac{T_i - c_j}{T_j} \right\rceil + 1 \right) \beta_{i,j}$.

Let tasks in the denominator of (2.24) be given indexes k instead of i, and l instead of j. Let tasks in both the numerator and denominator of (2.24) be arranged in the non-increasing priority order, so that $i = k$ and $j = l$. Let α_{free} in (2.24) be divided into two parts: $\bar{\alpha}_{free}$ that contains only tasks with priority higher than τ_i, and $\hat{\alpha}_{free}$ that contains only tasks with priority lower than τ_i. Now, (2.24) becomes:

$$\begin{aligned} \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i} (\bar{\alpha}_{free} + \hat{\alpha}_{free})/T_i}{\sum_{\tau_k} \alpha_{rma}/T_k} \\ &= \frac{1}{2} + \frac{\sum_{\tau_i} \hat{\alpha}_{free}/T_i}{\sum_{\tau_k} \alpha_{rma}/T_k} \end{aligned} \tag{2.25}$$

For convenience, we introduce the following notations:

$$\begin{aligned} \zeta_1 &= \sum_{\tau_i} \frac{\sum_{(\tau_j \in \gamma_i) \wedge (p_j < p_i)} \left(\left\lceil \frac{T_i - c_j}{T_j} \right\rceil + 1 \right) \beta_{i,j}}{T_i} \\ &= \sum_{T_i} \hat{\alpha}_{free}/T_i \\ \zeta_2 &= \sum_{\tau_k} \frac{\sum_{(\tau_l \in \gamma_k) \wedge (p_l > p_k)} \left(\left\lceil \frac{T_k - c_l}{T_l} \right\rceil + 1 \right) \beta_{k,l}}{T_k} \\ &= \frac{1}{2} \sum_{\tau_k} \alpha_{rma}/T_k \end{aligned}$$

τ_j is of lower priority than τ_i, which means $D_j > D_i$. Under G-RMA, this means, $T_j > T_i$. Thus, $\left\lceil \frac{T_i - c_j}{T_j} \right\rceil = 1$ for all τ_j and $\zeta_1 = \sum_{\tau_i} (\sum_{(\tau_j \in \gamma_i) \wedge (p_j < p_i)} (2.\beta_{i,j}))/T_i$. Since ζ_1 contains all τ_j of lower priority than τ_i and ζ_2 contains all τ_l of higher priority than τ_k, and tasks are arranged in the non-increasing priority order, then for each $\tau_{i,j}$, there exists $\tau_{k,l}$ such that $i = l$ and $j = k$. Figure 2.9 illustrates this, where 0 means that the pair i, j does not exist

$$\begin{array}{cccccc} & j & 1 & 2 & \cdots & n \\ i & & & & & \\ 1 & & 0 & 1 & \cdots & 1 \\ 2 & & 0 & 0 & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \ddots & 1 \\ n & & 0 & 0 & \cdots & 0 \end{array} \qquad \begin{array}{cccccc} & l & 1 & 2 & \cdots & n \\ k & & & & & \\ 1 & & 0 & 0 & \cdots & 0 \\ 2 & & 1 & 0 & & \vdots \\ \vdots & & \vdots & \ddots & \ddots & 0 \\ n & & 1 & \cdots & 1 & 0 \end{array}$$

Figure 2.9: Task association for lower priority tasks than T_i and higher priority tasks than T_k

in ζ_1, and the pair k, l does not exist in ζ_2' (i.e., there is no task τ_l that will interfere with τ_k in ζ_2), and 1 means the opposite.

Thus, it can be seen that both the matrices are transposes of each other. Consequently, for each $\beta_{i,j}$, there exists $\beta_{k,l}$ such that $i = l$ and $j = k$. But the number of times τ_j accesses a shared object with τ_i may not be the same as the number of times τ_i accesses that same object. Thus, $\beta_{i,j}$ does not have to be the same as $\beta_{k,l}$, even if i, j and k, l are transposes of each other. Therefore, we can analyze the behavior of s_{max}/r_{max} based on the three parameters $\beta_{i,j}$, $\beta_{k,l}$, and $\left\lceil \frac{T_k - c_l}{T_l} \right\rceil$. If $\beta_{i,j}$ is increased so that $\beta_{i,j} \to \infty$, $\therefore$ (2.25) $\to \infty$. This is because, $\beta_{i,j}$ represents the number of times a lower priority task τ_j accesses shared objects with a higher priority task τ_i. While this number has a greater effect in lock-free, it does not have any effect under RCM, because lower priority tasks do not affect higher priority ones. Hence, s_{max} is allowed to be much greater than r_{max}.

Although the minimum value for $\beta_{i,j}$ is 1, mathematically, if $\beta_{i,j} \to 0$, then (2.25) $\to 1/2$. Here, changing $\beta_{i,j}$ does not affect the retry cost of RCM, but it does affect the retry cost of lock-free, because the contention between tasks is reduced. Thus, s_{max} is reduced in this case to a little more than half of r_{max} ("a little more" because the minimum value of $\beta_{i,j}$ is actually 1, not 0).

The change of s_{max}/r_{max} with respect to $\beta_{i,j}$ is illustrated in Figure 2.10(a). If $\beta_{k,l} \to \infty$, then (2.25) $\to 1/2$. This is because, $\beta_{k,l}$ represents the number of times a higher priority task τ_l accesses shared objects with a lower priority task τ_k. Under RCM, this will increase the retry cost, thus reducing s_{max}/r_{max}. But if $\beta_{k,l} \to 0$, then (2.25)$\to \infty$. This is due to the lower contention from a higher priority task τ_l to a lower priority task τ_k, which reduces the retry cost under RCM and allows s_{max} to be very large compared with r_{max}. Of course, the actual minimum value for $\beta_{k,l}$ is 1, and is illustrated in Figure 2.10(b).

The third parameter that affects s_{max}/r_{max} is T_k/T_l. If $T_l \ll T_k$, then $\left\lceil \frac{T_k - c_l}{T_l} \right\rceil \to \infty$, and (2.25) $\to 1/2$. This is due to a high number of interferences from a higher priority task τ_l to a lower priority task τ_k, which increases the retry cost under RCM, and consequently reduces s_{max}/r_{max}.

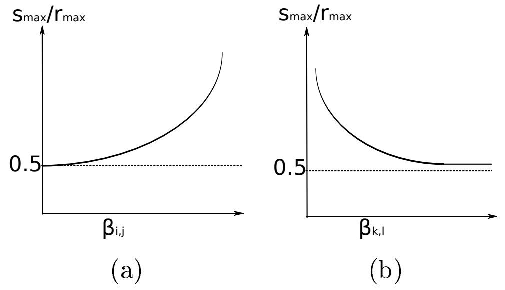


Figure 2.10: Change of s_{max}/r_{max}: a) $\frac{s_{max}}{r_{max}}$ versus $\beta_{i,j}$ and b) $\frac{s_{max}}{r_{max}}$ versus $\beta_{k,l}$

If $T_l = T_k$ (which is the maximum value for T_l as $D_l \leq D_k$, because τ_l has a higher priority than τ_k), then $\left\lceil \frac{T_k - c_l}{T_l} \right\rceil \rightarrow 1$ and $\zeta_2 = \sum_{\tau_k} \frac{\sum_{(\tau_l \in \gamma_k) \wedge (p_l > p_k)} 2\beta_{k,l}}{t_k}$. This means that the system will be controlled by only two parameters, $\beta_{i,j}$ and $\beta_{k,l}$, as in the previous two cases, illustrated in Figures 2.10(a) and 2.10(b). Claim follows. □

2.4 Conclusions

Under both ECM and RCM, a task incurs $2.s_{max}$ retry cost for each of its atomic sections due to a conflict with another task's atomic section. Retries under RCM and lock-free are affected by a larger number of conflicting task instances than under ECM. While task retries under ECM and lock-free are affected by all other tasks, retries under RCM are affected only by higher priority tasks.

STM and lock-free have similar parameters that affect their retry costs—i.e., the number of conflicting jobs and how many times they access shared objects. The s_{max}/r_{max} ratio determines whether STM is better or as good as lock-free. For ECM, this ratio cannot exceed 1, and it can be 1/2 for higher number of conflicting tasks. For RCM, for the common case, s_{max} must be 1/2 of r_{max}, and in some cases, s_{max} can be larger than r_{max} by many orders of magnitude.

Our work raises several questions. For example, what are the typical range of values for the different parameters that affect the retry cost (and hence the response time)? How tight is our retry and response time bounds in practice? Can real-time CMs be designed for other multiprocessor real-time schedulers (e.g., partitioned, semi-partitioned), and those that dynamically improve application timeliness behavior? These are important directions for further work.

Chapter 3

LCM

We consider software transactional memory (STM) concurrency control for multicore real-time software. We present a novel contention manager (CM) for resolving transactional conflicts, called length-based CM (or LCM). We upper bound transactional retries and response times under LCM, when used with G-EDF and G-RMA schedulers. We identify the conditions under which LCM outperforms previous real-time STM CMs and lock-free synchronization. Our implementation and experimental studies reveal that G-EDF/LCM and G-RMA/LCM have shorter or comparable retry costs and response times than other synchronization techniques.

3.1 Length-based CM

LCM resolves conflicts based on the priority of conflicting jobs, besides the length of the interfering atomic section, and the length of the interfered atomic section. This is in contrast to ECM and RCM [?], where conflicts are resolved using the priority of the conflicting jobs. This strategy allows lower priority jobs, under LCM, to retry for lesser time than that under ECM and RCM, but higher priority jobs, sometimes, wait for lower priority ones with bounded priority-inversion.

3.1.1 Design and Rationale

For both ECM and RCM, $s_i^k(\theta)$ can be totally repeated if $s_j^l(\theta)$ — which belongs to a higher priority job τ_j^b than τ_i^a — conflicts with $s_i^k(\theta)$ at the end of its execution, while $s_i^k(\theta)$ is just about to commit. Thus, LCM, shown in Algorithm 1, uses the remaining length of $s_i^k(\theta)$ when it is interfered, as well as $len(s_j^l(\theta))$, to decide which transaction must be aborted. If p_i^k was greater than p_j^l, then $s_i^k(\theta)$ would be the one that commits, because it belongs

Algorithm 1: LCM

```
Data: $s_i^k(\theta) \rightarrow$ interfered atomic section.
$s_j^l(\theta) \rightarrow$ interfering atomic section.
$\psi \rightarrow$ predefined threshold $\in [0, 1]$.
$\delta_i^k(\theta) \rightarrow$ remaining execution length of $s_i^k(\theta)$
Result: which atomic section of $s_i^k(\theta)$ or $s_j^l(\theta)$ aborts
1  if $p_i^k > p_j^l$ then
2      $s_j^l(\theta)$ aborts;
3  else
4      $c_{ij}^{kl} = len(s_j^l(\theta))/len(s_i^k(\theta))$;
5      $\alpha_{ij}^{kl} = ln(\psi)/(ln(\psi) - c_{ij}^{kl})$;
6      $\alpha = \left(len(s_i^k(\theta)) - \delta_i^k(\theta)\right)/len(s_i^k(\theta))$;
7      if $\alpha \leq \alpha_{ij}^{kl}$ then
8          $s_i^k(\theta)$ aborts;
9      else
10         $s_j^l(\theta)$ aborts;
11     end
12 end
```

to a higher priority job, and it started before $s_j^l(\theta)$ (step 2). Otherwise, c_{ij}^{kl} is calculated (step 4) to determine whether it is worth aborting $s_i^k(\theta)$ in favor of $s_j^l(\theta)$, because $len(s_j^l(\theta))$ is relatively small compared to the remaining execution length of $s_i^k(\theta)$ (explained further).

We assume that:

$$c_{ij}^{kl} = len(s_j^l(\theta))/len(s_i^k(\theta)) \tag{3.1}$$

where $c_{ij}^{kl} \in]0, \infty[$, to cover all possible lengths of $s_j^l(\theta)$. Our idea is to reduce the opportunity for the abort of $s_i^k(\theta)$ if it is close to committing when interfered and $len(s_j^l(\theta))$ is large. This abort opportunity is increasingly reduced as $s_i^k(\theta)$ gets closer to the end of its execution, or $len(s_j^l(\theta))$ gets larger.

On the other hand, as $s_i^k(\theta)$ is interfered early, or $len(s_j^l(\theta))$ is small compared to $s_i^k(\theta)$'s remaining length, the abort opportunity is increased even if $s_i^k(\theta)$ is close to the end of its execution. To decide whether $s_i^k(\theta)$ must be aborted or not, we use a threshold value $\psi \in [0, 1]$ that determines α_{ij}^{kl} (step 5), where α_{ij}^{kl} is the maximum percentage of $len(s_i^k(\theta))$ below which $s_j^l(\theta)$ is allowed to abort $s_i^k(\theta)$. Thus, if the already executed part of $s_i^k(\theta)$ — when $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ — does not exceed $\alpha_{ij}^{kl} len(s_i^k(\theta))$, then $s_i^k(\theta)$ is aborted (step 8). Otherwise, $s_j^l(\theta)$ is aborted (step 10).

The behavior of LCM is illustrated in Figure 3.1. In this figure, the horizontal axis corresponds to different values of α ranging from 0 to 1, and the vertical axis corresponds to different values of abort opportunities, $f(c_{ij}^{kl}, \alpha)$, ranging from 0 to 1 and calculated by (3.2):

$$f(c_{ij}^{kl}, \alpha) = e^{\frac{-c_{ij}^{kl}\alpha}{1-\alpha}} \tag{3.2}$$

where c_{ij}^{kl} is calculated by (3.1).

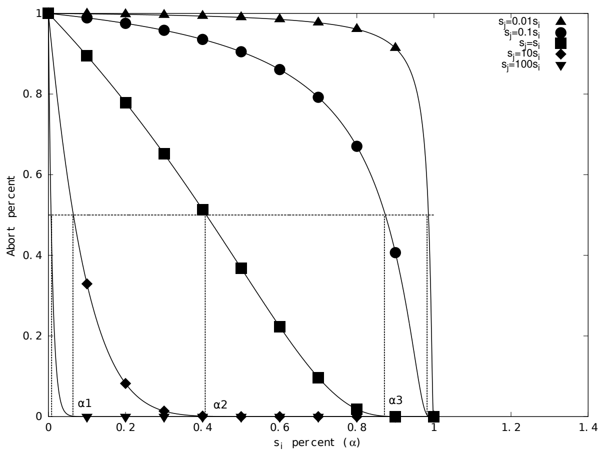


Figure 3.1: Interference of $s_i^k(\theta)$ by various lengths of $s_j^l(\theta)$

Figure 3.1 shows one atomic section $s_i^k(\theta)$ (whose α changes along the horizontal axis) interfered by five different lengths of $s_j^l(\theta)$. For a predefined value of $f(c_{ij}^{kl}, \alpha)$ (denoted as ψ in Algorithm 1), there corresponds a specific value of α (which is α_{ij}^{kl} in Algorithm 1) for each curve. For example, when $len(s_j^l(\theta)) = 0.1 \times len(s_i^k(\theta))$, $s_j^l(\theta)$ aborts $s_i^k(\theta)$ if the latter has not executed more than $\alpha 3$ percentage (shown in Figure 3.1) of its execution length. As $len(s_j^l(\theta))$ decreases, the corresponding α_{ij}^{kl} increases (as shown in Figure 3.1, $\alpha 3 > \alpha 2 > \alpha 1$).

Equation (3.2) achieves the desired requirement that the abort opportunity is reduced as $s_i^k(\theta)$ gets closer to the end of its execution (as $\alpha \to 1$, $f(c_{ij}^{kl}, 1) \to 0$), or as the length of the conflicting transaction increases (as $c_{ij}^{kl} \to \infty$, $f(\infty, \alpha) \to 0$). Meanwhile, this abort opportunity is increased as $s_i^k(\theta)$ is interfered closer to its release (as $\alpha \to 0$, $f(c_{ij}^{kl}, 0) \to 1$), or as the length of the conflicting transaction decreases (as $c_{ij}^{kl} \to 0$, $f(0, \alpha) \to 1$).

LCM is not a centralized CM, which means that, upon a conflict, each transactions has to decide whether it must commit or abort.

Claim 6. *Let $s_j^l(\theta)$ interfere once with $s_i^k(\theta)$ at α_{ij}^{kl}. Then, the maximum contribution of $s_j^l(\theta)$ to $s_i^k(\theta)$'s retry cost is:*

$$W_i^k(s_j^l(\theta)) \leq \alpha_{ij}^{kl} len\Big(s_i^k(\theta)\Big) + len\Big(s_j^l(\theta)\Big) \tag{3.3}$$

Proof. If $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ at a Υ percentage, where $\Upsilon < \alpha_{ij}^{kl}$, then the retry cost of $s_i^k(\theta)$ is $\Upsilon len(s_i^k(\theta)) + len(s_j^l(\theta))$, which is lower than that calculated in (3.3). Besides, if $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ after α_{ij}^{kl} percentage, then $s_i^k(\theta)$ will not abort. □

Claim 7. *An atomic section of a higher priority job, τ_j^b, may have to abort and retry due to a lower priority job, τ_i^a, if $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ after the α_{ij}^{kl} percentage. τ_j's retry time, due to $s_i^k(\theta)$ and $s_j^l(\theta)$, is upper bounded by:*

$$W_j^l(s_i^k(\theta)) \leq \Big(1 - \alpha_{ij}^{kl}\Big) len\Big(s_i^k(\theta)\Big) \tag{3.4}$$

Proof. It is derived directly from Claim 6, as $s_j^l(\theta)$ will have to retry for the remaining length of $s_i^k(\theta)$. □

Claim 8. *A higher priority job, τ_i^z, suffers from priority inversion for at most number of atomic sections in τ_i^z.*

Proof. Assuming three atomic sections, $s_i^k(\theta)$, $s_j^l(\theta)$ and $s_a^b(\theta)$, where $p_j > p_i$ and $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ after α_{ij}^{kl}. Then $s_j^l(\theta)$ will have to abort and retry. At this time, if $s_a^b(\theta)$ interferes with the other two atomic sections, and the LCM decides which transaction to commit based on comparison between each two transactions. So, we have the following cases:-

- $p_a < p_i < p_j$, then $s_a^b(\theta)$ will not abort any one because it is still in its beginning and it is of the lowest priority. So. τ_j is not indirectly blocked by τ_a.
- $p_i < p_a < p_j$ and even if $s_a^b(\theta)$ interferes with $s_i^k(\theta)$ before α_{ia}^{kb}, so, $s_a^b(\theta)$ is allowed abort $s_i^k(\theta)$. Comparison between $s_j^l(\theta)$ and $s_a^b(\theta)$ will result in LCM choosing $s_j^l(\theta)$ to commit and abort $s_a^b(\theta)$ because the latter is still beginning, and τ_j is of higher priority. If $s_a^b(\theta)$ is not allowed to abort $s_i^k(\theta)$, the situation is still the same, because $s_j^l(\theta)$ was already retrying until $s_i^k(\theta)$ finishes.
- $p_a > p_j > p_i$, then if $s_a^b(\theta)$ is chosen to commit, this is not priority inversion for τ_j because τ_a is of higher priority.
- if τ_a preempts τ_i, then LCM will compare only between $s_j^l(\theta)$ and $s_a^b(\theta)$. If $p_a < p_j$, then $s_j^l(\theta)$ will commit because of its task's higher priority and $s_a^b(\theta)$ is still at its beginning, otherwise, $s_j^l(\theta)$ will retry, but this will not be priority inversion because τ_a is already of higher priority than τ_j. If τ_a does not access any object but it preempts τ_i, then CM will choose $s_j^l(\theta)$ to commit as only already running transactions are competing together.

So, by generalizing these cases to any number of conflicting jobs, it is seen that when an atomic section, $s_j^l(\theta)$, of a higher priority job is in conflict with a number of atomic sections belonging to lower priority jobs, $s_j^l(\theta)$ can suffer from priority inversion by only one of them. So, each higher priority job can suffer priority inversion at most its number of atomic section. Claim follows. □

Claim 9. *The maximum delay suffered by $s_j^l(\theta)$ due to lower priority jobs is caused by the maximum length atomic section accessing object θ, which belongs to a lower priority job than τ_j^b that owns $s_j^l(\theta)$.*

Proof. Assume three atomic sections, $s_i^k(\theta)$, $s_j^l(\theta)$, and $s_h^z(\theta)$, where $p_j > p_i$, $p_j > p_h$, and $len(s_i^k(\theta)) > len(s_h^z(\theta))$. Now, $\alpha_{ij}^{kl} > \alpha_{hj}^{zl}$ and $c_{ij}^{kl} < c_{hj}^{zl}$. By applying (3.4) to obtain the contribution of $s_i^k(\theta)$ and $s_h^z(\theta)$ to the priority inversion of $s_j^l(\theta)$ and dividing them, we get:

$$\frac{W_j^l(s_i^k(\theta))}{W_j^l(s_h^z(\theta))} = \frac{\left(1-\alpha_{ij}^{kl}\right) len(s_i^k(\theta))}{\left(1-\alpha_{hj}^{zl}\right) len(s_h^z(\theta))}$$

By substitution for αs from (3.2):

$$= \frac{(1-\frac{ln\psi}{ln\psi-c_{ij}^{kl}})len(s_i^k(\theta))}{(1-\frac{ln\psi}{ln\psi-c_{hj}^{zl}})len(s_h^z(\theta))} = \frac{(\frac{-c_{ij}^{kl}}{ln\psi-c_{ij}^{kl}})len(s_i^k(\theta))}{(\frac{-c_{hj}^{zl}}{ln\psi-c_{hj}^{zl}})len(s_h^z(\theta))}$$

$\because ln\psi \leq 0$ and $c_{ij}^{kl}, c_{hj}^{kl} > 0, \therefore$ by substitution from (3.1)

$$= \frac{len(s_j^l(\theta))/(ln\psi - c_{ij}^{kl})}{len(s_j^l(\theta))/(ln\psi - c_{hj}^{zl})} = \frac{ln\psi - c_{hj}^{zl}}{ln\psi - c_{ij}^{kl}} > 1$$

Thus, as the length of the interfered atomic section increases, the delay suffered by the interfering atomic section increases. Claim follows. □

3.1.2 Response Time of G-EDF/LCM

Claim 10. *$RC(T_i)$ for a task τ_i under G-EDF/LCM is upper bounded by:*

$$\begin{aligned} RC(T_i) &= \Bigg(\sum_{\forall\tau_h\in\gamma_i}\sum_{\forall\theta\in\theta_i\wedge\theta_h}\Bigg(\left\lceil\frac{T_i}{T_h}\right\rceil\sum_{\forall s_h^l(\theta)} len\Big(s_h^l(\theta)\Big) \\ &+ \alpha_{max}^{hl} len\Big(s_{max}^h(\theta)\Big)\Bigg)\Bigg) \\ &+ \sum_{\forall s_i^y(\theta)}\Big(1-\alpha_{max}^{iy}\Big) len\Big(s_{max}^i(\theta)\Big) \end{aligned} \quad (3.5)$$

where α_{max}^{hl} is the α value that corresponds to ψ due to the interference of $s_{max}^h(\theta)$ by $s_h^l(\theta)$. α_{max}^{iy} is the α value that corresponds to ψ due to the interference of $s_{max}^i(\theta)$ by $s_i^y(\theta)$.

Proof. The maximum number of higher priority instances of τ_h that can interfere with τ_i^x is $\left\lceil\frac{T_i}{T_h}\right\rceil$, as shown in Figure 3.2, where one instance of τ_h and τ_h^p coincides with the absolute deadline of τ_i^x.

By using Claims 6, 7, **??**, and 9, and Claim 1 in [?] to determine the effect of atomic sections belonging to higher and lower priority instances of interfering tasks to τ_i^x, Claim follows. □

Response time of τ_i is calculated by (11) in [?].

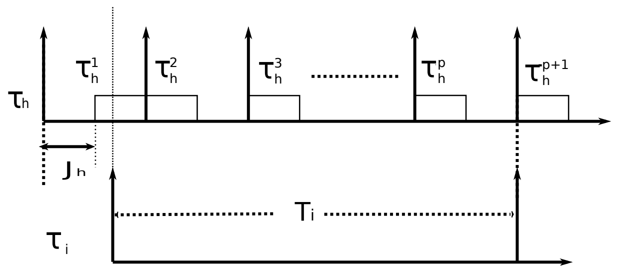


Figure 3.2: τ_h^p has a higher priority than τ_i^x

3.1.3 Schedulability of G-EDF/LCM and ECM

We now compare the schedulability of G-EDF/LCM with ECM [?] to understand when G-EDF/LCM will perform better. Toward this, we compare the total utilization of ECM with that of G-EDF/LCM. For each method, we inflate the c_i of each task τ_i by adding the retry cost suffered by τ_i. Thus, if method A adds retry cost $RC_A(T_i)$ to c_i, and method B adds retry cost $RC_B(T_i)$ to c_i, then the schedulability of A and B are compared as:

$$\begin{aligned}\sum_{\forall \tau_i} \frac{c_i + RC_A(T_i)}{T_i} &\leq \sum_{\forall \tau_i} \frac{c_i + RC_B(T_i)}{T_i} \\ \sum_{\forall \tau_i} \frac{RC_A(T_i)}{T_i} &\leq \sum_{\forall \tau_i} \frac{RC_B(T_i)}{T_i} \qquad (3.6)\end{aligned}$$

Thus, schedulability is compared by substituting the retry cost added by the synchronization methods in (3.6).

Claim 11. *Let s_{max} be the maximum length atomic section accessing any object θ. Let α_{max} and α_{min} be the maximum and minimum values of α for any two atomic sections $s_i^k(\theta)$ and $s_j^l(\theta)$. Given a threshold ψ, schedulability of G-EDF/LCM is equal or better than ECM if for any task τ_i:*

$$\frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil \qquad (3.7)$$

Proof. Under ECM, $RC(T_i)$ is upper bounded by:

$$RC(T_i) \leq \sum_{\forall \tau_h \in \gamma_i} \sum_{\forall \theta \in \ (\theta_i \wedge \theta_h)} \left(\left\lceil \frac{T_i}{T_h} \right\rceil \sum_{\forall s_h^z(\theta)} 2len(s_{max}) \right) \qquad (3.8)$$

with the assumption that all lengths of atomic sections of (4) and (8) in [?] and (3.5) are replaced by s_{max}. Let α_{max}^{hl} in (3.5) be replaced with α_{max}, and α_{max}^{iy} in (3.5) be replaced with α_{min}. As α_{max}, α_{min}, and $len(s_{max})$ are all constants, (3.5) is upper bounded by:

$$\begin{aligned} RC(T_i) &\leq \Bigg(\sum_{\forall \tau_h \in \gamma_i} \sum_{\forall \theta \in \theta_i \wedge \theta_h} \Bigg(\left\lceil \frac{T_i}{T_h} \right\rceil \sum_{\forall s_h^l(\theta)} (1+\alpha_{max}) \\ & \qquad len\Big(s_{max}\Big)\Bigg)\Bigg) + \sum_{\forall s_i^y(\theta)} \Big(1-\alpha_{min}\Big) len\Big(s_{max}\Big) \end{aligned} \tag{3.9}$$

If β_1^{ih} is the total number of times any instance of τ_h accesses shared objects with τ_i, then $\beta_1^{ih} = \sum_{\forall \theta \in (\theta_i \wedge \theta_h)} \sum_{\forall s_h^z(\theta)}$. Furthermore, if β_2^i is the total number of times any instance of τ_i accesses shared objects with any other instance, $\beta_2^i = \sum_{\forall s_i^y(\theta)}$, *where θ is shared with another task*. Then, $\beta_i = max\{max_{\forall \tau_h \in \gamma_i}\{\beta_1^{ih}\}, \beta_2^i\}$ is the maximum number of accesses to all shared objects by any instance of τ_i or τ_h. Thus, (3.8) becomes:

$$RC(T_i) \leq \sum_{\tau_h \in \gamma_i} 2 \left\lceil \frac{T_i}{T_h} \right\rceil \beta_i len(s_{max}) \tag{3.10}$$

and (3.9) becomes:

$$\begin{aligned} RC(T_i) &\leq \beta_i len(s_{max}) \Bigg((1-\alpha_{min}) \\ & \qquad + \sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \Bigg) \end{aligned} \tag{3.11}$$

We can now compare the total utilization of G-EDF/LCM with that of ECM by comparing (3.9) and (3.11) for all τ_i:

$$\begin{aligned} & \sum_{\forall \tau_i} \frac{(1-\alpha_{min}) + \sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right)}{T_i} \\ \leq & \sum_{\forall \tau_i} \frac{\sum_{\forall \tau_h \in \gamma_i} 2 \left\lceil \frac{T_i}{T_h} \right\rceil}{T_i} \end{aligned} \tag{3.12}$$

(3.12) is satisfied if for each τ_i, the following condition is satisfied:

$$(1-\alpha_{min}) + \sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right) \leq 2 \sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil$$

$$\therefore \frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil$$

Claim follows. □

3.1.4 G-EDF/LCM versus Lock-free

We consider the retry-loop lock-free synchronization for G-EDF given in [?]. This lock-free approach is the most relevant to our work.

Claim 12. *Let s_{max} denote $len(s_{max})$ and r_{max} denote the maximum execution cost of a single iteration of any retry loop of any task in the retry-loop lock-free algorithm in [?]. Now, G-EDF/LCM achieves higher schedulability than the retry-loop lock-free approach if the upper bound on s_{max}/r_{max} under G-EDF/LCM ranges between 0.5 and 2 (which is higher than that under ECM).*

Proof. From [?], the retry-loop lock-free algorithm is upper bounded by:

$$RL(T_i) = \sum_{\tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_i r_{max} \tag{3.13}$$

where β_i is as defined in Claim 11. The retry cost of τ_i in G-EDF/LCM is upper bounded by (3.11). By comparing G-EDF/LCM's total utilization with that of the retry-loop lock-free algorithm, we get:

$$\sum_{\forall \tau_i} \frac{\left((1-\alpha_{min}) + \sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right)\right) \beta_i s_{max}}{T_i}$$

$$\leq \sum_{\forall \tau_i} \frac{\sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_i r_{max}}{T_i}$$

$$\therefore \frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_i}{T_i}}{\sum_{\forall \tau_i} \frac{\left((1-\alpha_{min}) + \sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right)\right) \beta_i}{T_i}} \tag{3.14}$$

Let the number of tasks that have shared objects with τ_i be ω (i.e., $\sum_{\tau_h \in \gamma_i} = \omega \geq 1$ since at least one task has a shared object with τ_i; otherwise, there is no conflict between tasks). Let the total number of tasks be n, so $1 \leq \omega \leq n-1$, and $\left\lceil \frac{T_i}{T_h} \right\rceil \in [1, \infty[$. To find the minimum and maximum values for the upper bound on s_{max}/r_{max}, we consider the following cases:

- $\alpha_{min} \to 0, \alpha_{max} \to 0$

$\therefore$ (3.14) will be:

$$\frac{s_{max}}{r_{max}} \leq 1 + \frac{\sum_{\forall \tau_i} \frac{\omega - 1}{T_i}}{\sum_{\forall \tau_i} \frac{1 + \sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil}{T_i}} \tag{3.15}$$

By substituting the edge values for ω and $\left\lceil \frac{T_i}{T_h} \right\rceil$ in (3.15), we derive that the upper bound on s_{max}/r_{max} lies between 1 and 2.

- $\alpha_{min} \rightarrow 0, \alpha_{max} \rightarrow 1$

(3.14) becomes

$$\frac{s_{max}}{r_{max}} \leq 0.5 + \frac{\sum_{\forall \tau_i} \frac{\omega - 0.5}{T_i}}{\sum_{\forall \tau_i} \frac{1 + 2\sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil}{T_i}} \tag{3.16}$$

By applying the edge values for ω and $\left\lceil \frac{T_i}{T_h} \right\rceil$ in (3.16), we derive that the upper bound on s_{max}/r_{max} lies between 0.5 and 1.

- $\alpha_{min} \rightarrow 1, \alpha_{max} \rightarrow 0$

This case is rejected since $\alpha_{min} \leq \alpha_{max}$.

- $\alpha_{min} \rightarrow 1, \alpha_{max} \rightarrow 1$

$\therefore$ (3.14) becomes:

$$\frac{s_{max}}{r_{max}} \leq 0.5 + \frac{\sum_{\tau_i} \frac{\omega}{T_i}}{2\sum_{\tau_i} \frac{\sum_{\forall \tau_h \in \gamma_i} \left\lceil \frac{T_i}{T_h} \right\rceil}{T_i}} \tag{3.17}$$

By applying the edge values for ω and $\left\lceil \frac{T_i}{T_h} \right\rceil$ in (3.17), we derive that the upper bound on s_{max}/r_{max} lies between 0.5 and 1, which is similar to that achieved by ECM.

Summarizing from the previous cases, the upper bound on s_{max}/r_{max} lies between 0.5 and 2, whereas for ECM [?], it lies between 0.5 and 1. Claim follows.

$\square$

3.1.5 Response Time of G-RMA/LCM

Claim 13. *Let* $\lambda_2(j,\theta) = \sum_{\forall s_j^l(\theta)} len(s_j^l(\theta)) + \alpha_{max}^{jl} len(s_{max}^j(\theta))$*, where* α_{max}^{jl} *is the* α *value corresponding to* ψ *due to the interference of* $s_{max}^j(\theta)$ *by* $s_j^l(\theta)$*. The retry cost of any task* τ_i *under G-RMA/LCM during* T_i *is given by:*

$$\begin{aligned} RC\left(T_i\right) &= \sum_{\forall \tau_j^*} \left(\sum_{\theta \in (\theta_i \wedge \theta_j)} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) \lambda_2(j,\theta) \right) \right) \\ &+ \sum_{\forall s_i^y(\theta)} \left(1 - \alpha_{max}^{iy}\right) len\left(s_{max}^i(\theta)\right) \end{aligned} \tag{3.18}$$

where $\tau_j^* = \{\tau_j | (\tau_j \in \gamma_i) \wedge (p_j > p_i)\}$.

Proof. Under G-RMA, all instances of a higher priority task, τ_j, can conflict with a lower priority task, τ_i, during T_i. (3.3) can be used to determine the contribution of each conflicting atomic section in τ_j to τ_i. Meanwhile, all instances of any task with lower priority than τ_i can conflict with τ_i during T_i. Claims 7 and **??** can be used to determine the contribution of conflicting atomic sections in lower priority tasks to τ_i. Using the previous notations and Claim 3 in [**?**], the Claim follows. □

The response time is calculated by (17) in [**?**] with replacing $RC(R_i^{up})$ with $RC(T_i)$.

3.1.6 Schedulability of G-RMA/LCM and RCM

Claim 14. *Under the same assumptions of Claims 11 and 13, G-RMA/LCM's schedulability is equal or better than RCM if:*

$$\frac{1-\alpha_{min}}{1-\alpha_{max}} \le \sum_{\forall \tau_j^*} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) \tag{3.19}$$

Proof. Under the same assumptions as that of Claims 11 and 13, (3.18) can be upper bounded as:

$$\begin{aligned} RC(T_i) &\le \sum_{\forall \tau_j^*} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) (1+\alpha_{max}) len(s_{max}) \beta_i \right) \\ &+ (1-\alpha_{min}) len(s_{max}) \beta_i \end{aligned} \tag{3.20}$$

For RCM, (16) in [**?**] for $RC(T_i)$ is upper bounded by:

$$RC(T_i) \le \sum_{\forall \tau_j^*} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) 2\beta_i len(s_{max})$$

By comparing the total utilization of G-RMA/LCM with that of RCM, we get:

$$\begin{aligned} &\textstyle\sum_{\forall \tau_i} \frac{len(s_{max})\beta_i \left((1-\alpha_{min}) + \sum_{\forall \tau_j^*} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) (1+\alpha_{max}) \right) \right)}{T_i} \\ \le \quad &\textstyle\sum_{\forall \tau_i} \frac{2 len(s_{max}) \beta_i \sum_{\forall \tau_j^*} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right)}{T_i} \end{aligned} \tag{3.21}$$

(3.21) is satisfied if $\forall \tau_i$ (3.19) is satisfied. Claim follows. □

3.1.7 G-RMA/LCM versus Lock-free

Although [?] considers retry-loop lock-free synchronization for G-EDF systems, [?] also applies for G-RMA systems.

Claim 15. *Let s_{max} denote $len(s_{max})$ and r_{max} denote the maximum execution cost of a single iteration of any retry loop of any task in the retry-loop lock-free algorithm in [?]. G-RMA/LCM achieves higher schedulability than the retry-loop lock-free approach if the upper bound on s_{max}/r_{max} under G-RMA/LCM is no less than 0.5. Upper bound on s_{max}/r_{max} can extend to large values when α_{min} and α_{max} are very large.*

Proof. Retry cost for G-RMA/LCM is upper bounded by (3.18). Let $\gamma_i = \tau_j^* \cup \bar{\tau_j}$, where τ_j^* is the set of higher priority tasks than τ_i sharing objects with τ_i. $\bar{\tau_j}$ is the set of lower priority tasks than τ_i sharing objects with it. We follow the same definitions of β_i, r_{max} and $RL(T_i)$ given in the proof of Claim (12). Schedulability of G-RMA/LCM equals or exceeds schedulability of retry-loop lock-free if

$$\begin{aligned}\frac{s_{max}}{r_{max}} \leq & \frac{\sum_{\forall \tau_i} \frac{\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j}\right\rceil+1\right)}{T_i}}{\sum_{\forall \tau_i} \frac{\left(1-\alpha_{min}\right)+\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j}\right\rceil+1\right)(1+\alpha_{max})}{T_i}} \\ + & \frac{2\sum_{\forall \tau_i}\frac{\sum_{\forall \bar{\tau_j}}}{T_i}}{\sum_{\forall \tau_i} \frac{\left(1-\alpha_{min}\right)+\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j}\right\rceil+1\right)(1+\alpha_{max})}{T_i}}\end{aligned} \tag{3.22}$$

If $p_j < p_i$, $\therefore \left\lceil \frac{T_i}{T_j} \right\rceil = 1$ because the system assumes impilicit deadline tasks and uses G-RMA scheduler. Let ω_1 be size of τ_i^* and ω_2 be size of $\bar{\tau_i}$. $\therefore \omega_1^i \geq 1$ and $\omega_2^i \geq 1$. Otherwise, there is no conflict with τ_i. To find the maximum and minimum upper bounds for s_{max}/r_{max}, the following cases are considered:

- $\alpha_{min} \rightarrow 0$, $\alpha_{max} \rightarrow 0$

$$\therefore \frac{s_{max}}{r_{max}} \leq 1 + \frac{\sum_{\forall \tau_i} \frac{2\omega_2^i - 1}{T_i}}{\sum_{\forall \tau_i} \frac{1+\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j}\right\rceil+1\right)}{T_i}} \tag{3.23}$$

As the second term in (3.23) is alawys positive (because $\omega_2^i \geq 1$), then the minimum upper bound on s_{max}/r_{max} is 1. To get the maximum upper bound on s_{max}/r_{max}, let $\left\lceil \frac{T_i}{T_j} \right\rceil$ approaches its minimum value 1, $\omega_1^i \to 0$, $\omega_2^i \to n-1$ (the maximum and minimum values for ω_1^i and ω_2^i respectively. n is number of tasks), then

$$\therefore \frac{s_{max}}{r_{max}} \leq (2n-2)$$

Of course, n cannot be lower than 2. Otherwise, there will be no conflicting tasks.

- $\alpha_{min} \to 0$, $\alpha_{max} \to 1$

$$\frac{s_{max}}{r_{max}} \leq \frac{1}{2} + \frac{\sum_{\forall \tau_i} \frac{4\omega_2^i - 1}{T_i}}{2\sum_{\forall \tau_i} \frac{1+2\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right)}{T_i}} \tag{3.24}$$

The minimum upper bound for s_{max}/r_{max} is 0.5. This can happen when $T_i \gg T_j$. To get the maximum upper bound on s_{max}/r_{max}, let $\left\lceil \frac{T_i}{T_j} \right\rceil$ approaches its minimum value 1, $\omega_2^i \to n-1$, $\omega_1^i \to 0$, then

$$\frac{s_{max}}{r_{max}} \leq 2n-2$$

- $\alpha_{min} \to 1$, $\alpha_{max} \to 0$

This case is rejected because α_{max} must be greater or equal to α_{min}.

- $\alpha_{min} \to 1$, $\alpha_{max} \to 1$

$$\frac{s_{max}}{r_{max}} \leq \frac{1}{2} + \frac{\sum_{\forall \tau_i} \frac{\omega_2^i}{T_i}}{\sum_{\forall \tau_i} \frac{\sum_{\tau_j^*}\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right)}{T_i}} \tag{3.25}$$

The minimum upper bound for s_{max}/r_{max} is 0.5. This can happen when $T_i \gg T_j$. To get the maximum upper bound on s_{max}/r_{max}, let $\left\lceil \frac{T_i}{T_j} \right\rceil$ approaches its minimum value 1, $\omega_2^i \to n-1$, $\omega_1^i \to 0$, then

$$\frac{s_{max}}{r_{max}} \to \infty$$

From the previous cases, we can derive that upper bound on s_{max}/r_{max} extends from 0.5 to large values. Claim follows.

□

Having established LCM's retry and response time upper bounds, and the conditions under which it outperforms ECM, RCM, and lock-free synchronization, we now would like to understand how LCM's retry and response times in practice (i.e., on average) compare with that of competitor methods. Since this can only be understood experimentally, we implement LCM and the competitor methods and conduct experimental studies.

Table 3.1: Task sets. (a) Task set 1: 5-task set; (b) Task set 2: 10-task set; (c) Task set 3: 12-task set

(a)

	$T_i(\mu s)$	$c_i(\mu s)$
τ_1	500000	150000
τ_2	1000000	227000
τ_3	1500000	410000
τ_4	3000000	299000
τ_5	5000000	500000

(b)

	$T_i(\mu s)$	$c_i(\mu s)$
τ_1	400000	75241
τ_2	750000	69762
τ_3	1200000	267122
τ_4	1500000	69863
τ_5	2400000	152014
τ_6	4000000	286301
τ_7	7500000	493150
τ_8	10000000	794520
τ_9	15000000	1212328
τ_{10}	20000000	1775342

(c)

	$T_i(\mu s)$	$c_i(\mu s)$
τ_1	400000	58195
τ_2	750000	53963
τ_3	1000000	206330
τ_4	1200000	53968
τ_5	1500000	117449
τ_6	2400000	221143
τ_7	3000000	290428
τ_8	4000000	83420
τ_9	7500000	380917
τ_{10}	10000000	613700
τ_{11}	15000000	936422
τ_{12}	20000000	1371302

3.1.8 Experimental Setup

We used the ChronOS real-time Linux kernel [?] and the RSTM library [?]. We modified RSTM to include implementations of ECM, RCM, G-EDF/LCM, and G-RMA/LCM contention managers, and modified ChronOS to include implementations of G-EDF and G-RMA schedulers.

For the retry-loop lock-free implementation, we used a loop that reads an object and attempts to write to the object using a compare-and-swap (CAS) instruction. The task retries until the CAS succeeds.

We use an 8 core, 2GHz AMD Opteron platform. The average time taken for one write operation by RSTM on any core is $0.0129653375\mu s$, and the average time taken by one CAS-loop operation on any core is $0.0292546250\ \mu s$.

We used the periodic task set shown in Table 3.1. Each task runs in its own thread and has a set of atomic sections. Atomic section properties are probabilistically controlled (for experimental evaluation) using three parameters: the maximum and minimum lengths of any atomic section within the task, and the total length of atomic sections within any task. All task atomic sections access the same object, and do write operations on the object (thus, contention is the highest).

3.1.9 Results

Figure 3.3 shows the retry cost (RC) for each task in the three task sets in Table 3.1, where each task has a single atomic section of length equal to 0.2 of the task length. Each data point in the figure has a confidence level of 0.95. We observe that G-EDF/LCM and G-RMA/LCM achieve shorter or comparable retry cost than ECM and RCM. Since all tasks are initially released at the same time, and due to the specific nature of task properties, tasks with lower IDs somehow have higher priorities under the G-EDF scheduler. Note that tasks with lower IDs have higher priorities under G-RMA, since tasks are ordered in non-decreasing order of their periods.

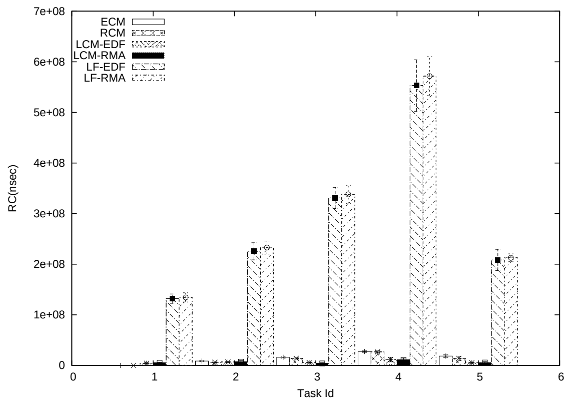
Thus, we observe that G-EDF/LCM and G-RMA/LCM achieve comparable retry costs to ECM and RCM for some tasks with lower IDs. But when task ID increases, LCM — for both schedulers — achieves much shorter retry costs than ECM and RCM. This is because, higher priority tasks in LCM can be delayed by lower priority tasks, which is not the case for ECM and RCM. However, as task priority decreases, LCM, by definition, prevents higher priority tasks from aborting lower priority ones if a higher priority task interferes with a lower priority one after a specified threshold. In contrast, under ECM and RCM, lower priority tasks abort in favor of higher priority ones. G-EDF/LCM and G-RMA/LCM also achieve shorter retry costs than the retry-loop lock-free algorithm.

Figure 3.4 shows the response time of each task of the task sets in Table 3.1 with a confidence level of 0.95. (Again, each task's atomic section length is equal to half of the task length.) We observe that G-EDF/LCM and G-RMA/LCM achieve shorter response time than the retry-loop lock-free algorithm, and shorter or comparable response time than ECM and RCM.

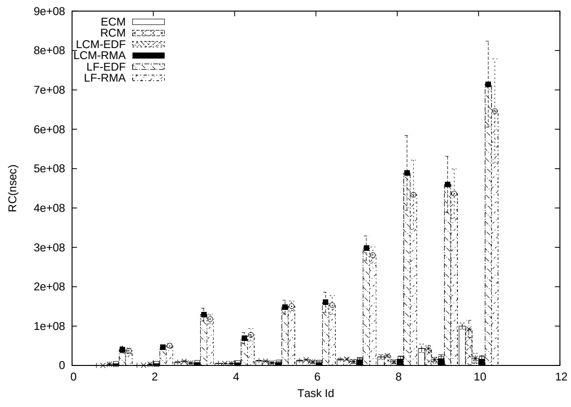
We repeated the experiments by varying the number and length of atomic sections. Due to space limitation, only subset of results are shown in Figures 3.5 to 3.8. A complete set of results are shown in Appendix B in [?]. Each figure has three parameters x, y, z in the lable. x specifies the relative total length of all atomic sections to the length of the task. y specifies the maximum relative length of any atomic section to the length of the task. z specifies the minimum relative length of any atomic section to the length of the task. These figures show a consistent trend with previous results.

3.2 Conclusions

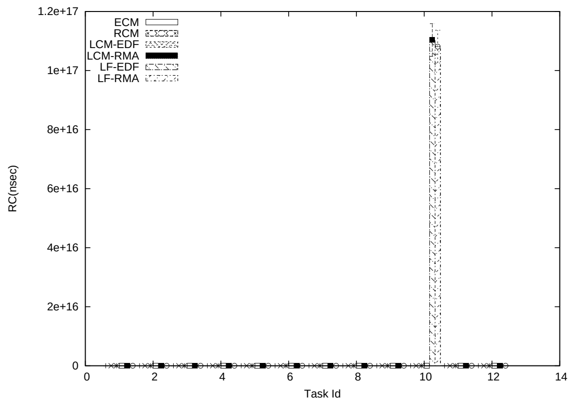
In ECM and RCM, a task incurs at most $2s_{max}$ retry cost for each of its atomic section due to conflict with another task's atomic section. With LCM, this retry cost is reduced to $(1 + \alpha_{max})s_{max}$ for each aborted atomic section. In ECM and RCM, tasks do not retry due to lower priority tasks, whereas in LCM, they do so. In G-EDF/LCM, retry due to a lower priority job is encountered only from a task τ_j's last job instance during τ_i's period.



(a) Task set 1

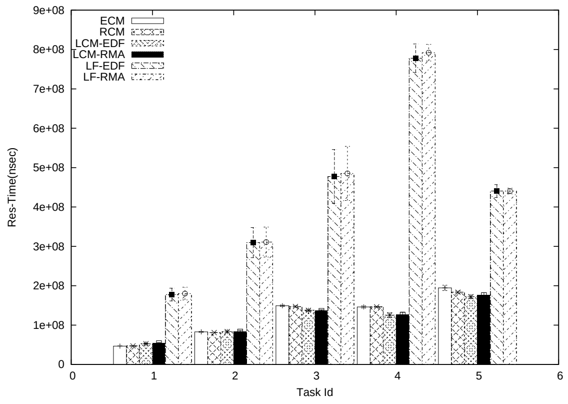


(b) Task set 2

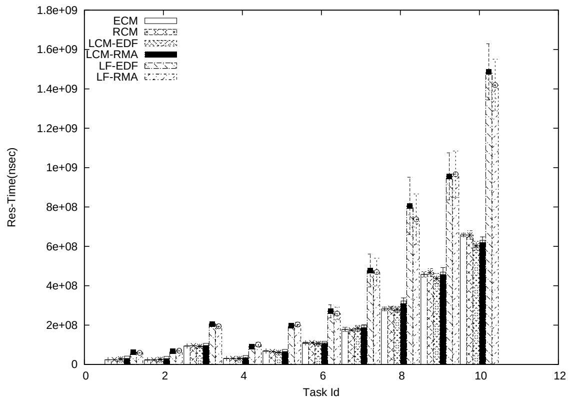


(c) Task set 3

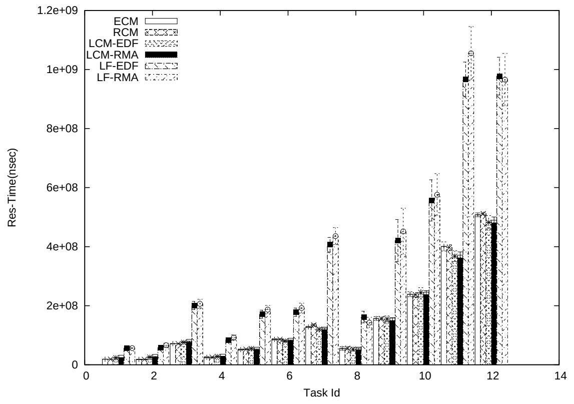
Figure 3.3: Task retry costs under LCM and competitor synchronization methods



(a) Task set 1

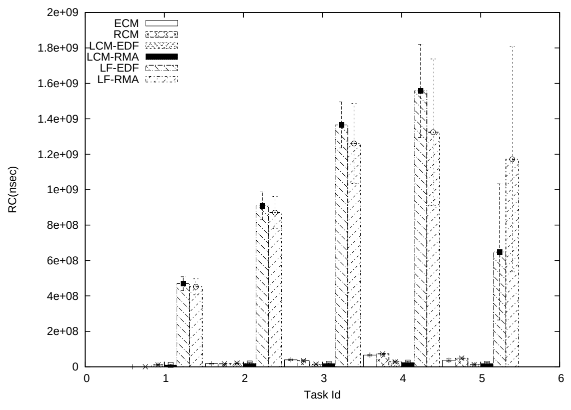


(b) Task set 2

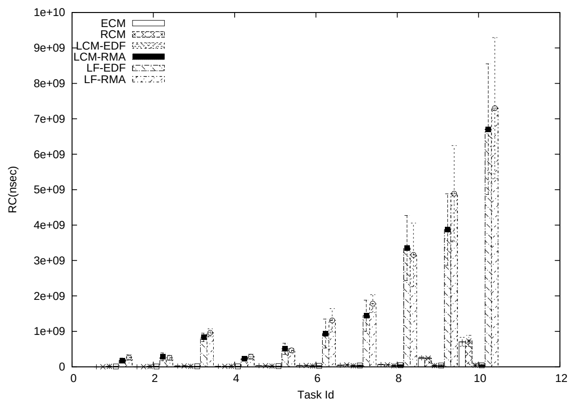


(c) Task set 3

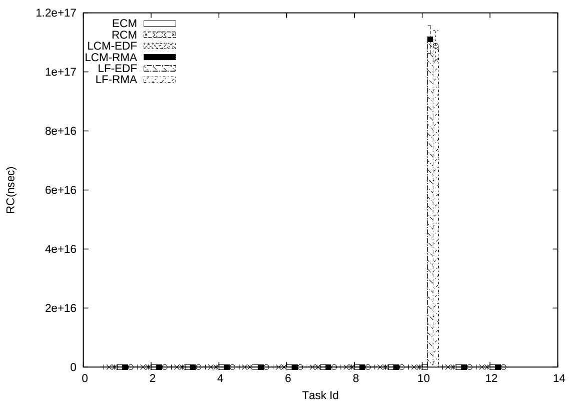
Figure 3.4: Task response times under LCM and competitor synchronization methods



(a) Task set 1

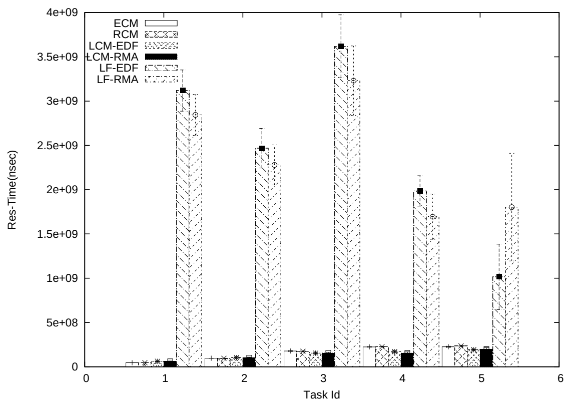


(b) Task set 2

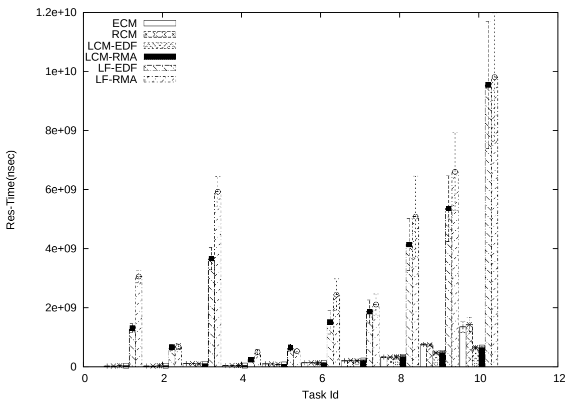


(c) Task set 3

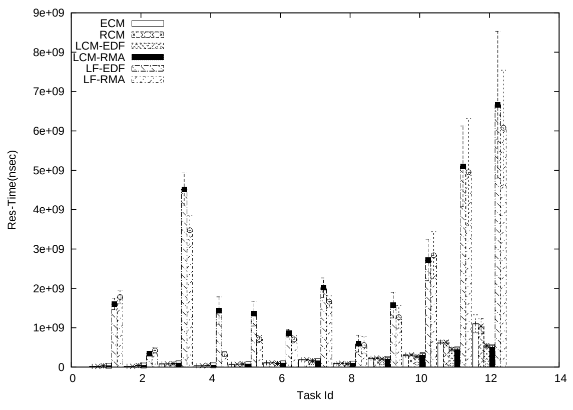
Figure 3.5: Task retry costs under LCM and competitor synchronization methods (0.5,0.2,0.2)



(a) Task set 1

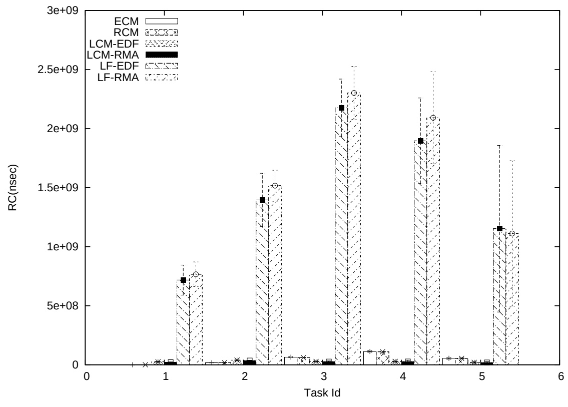


(b) Task set 2

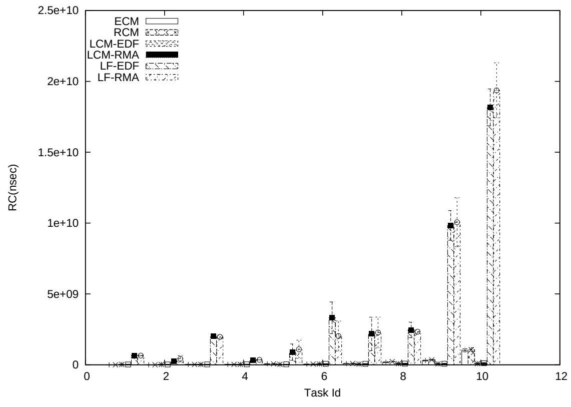


(c) Task set 3

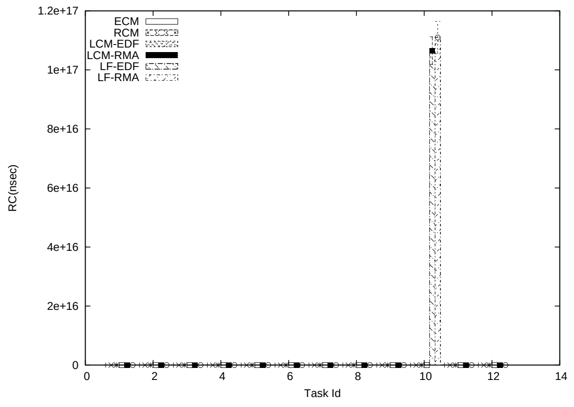
Figure 3.6: Task response times under LCM and competitor synchronization methods (0.5,0.2,0.2)



(a) Task set 1

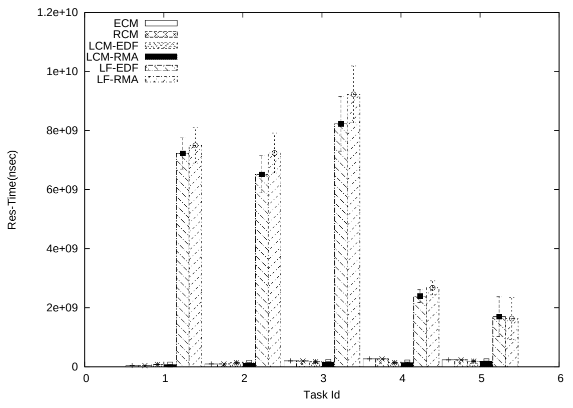


(b) Task set 2

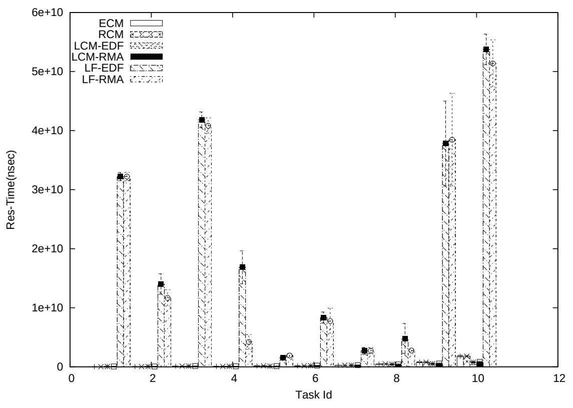


(c) Task set 3

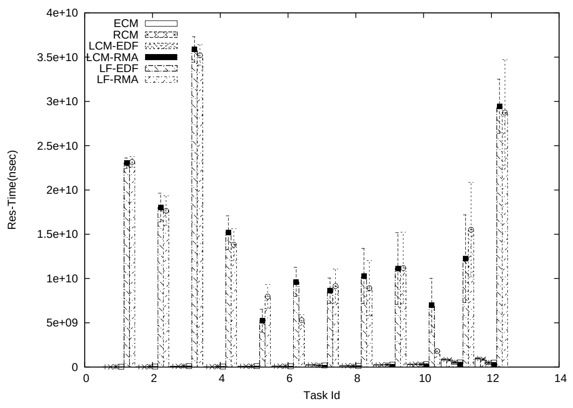
Figure 3.7: Task retry costs under LCM and competitor synchronization methods (0.8,0.5,0.2)



(a) Task set 1



(b) Task set 2



(c) Task set 3

Figure 3.8: Task response times under LCM and competitor synchronization methods (0.8,0.5,0.2)

This is not the case with G-RMA/LCM, because, each higher priority task can be aborted and retried by any job instance of lower priority tasks. Schedulability of G-EDF/LCM and G-RMA/LCM is better or equal to ECM and RCM, respectively, by proper choices for α_{min} and α_{max}. Schedulability of G-EDF/LCM is better than retry-loop lock-free synchronization for G-EDF if the upper bound on s_{max}/r_{max} is between 0.5 and 2, which is higher than that achieved by ECM. G-RMA/LCM acheives higher schedulability than retry-loop lock-free if s_{max}/r_{max} is not less than 0.5. For high values of α in G-RMA/LCM, s_{max}/r_{max} can extend to large values.