Concurrency Control for Multicore Embedded Real-Time Software Using STM

Journal:	*Transactions on Embedded Computing Systems*
Manuscript ID:	Draft
Manuscript Type:	Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Elshambakey, Mohammed; Mucsat, IRI; Virginia Tech, ECE Ravindran, Binoy; Virginia Tech, ECE
Computing Classification Systems:	real time, STM, Contention manager

Concurrency Control for Multicore Embedded Real-Time Software Using STM

We consider software transactional memory (STM) 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 retry-loop lock-free synchronization. We also 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 Global EDF and Global RMA schedulers. The conditions under which LCM is superior to the other real-time CMs and lock-free synchronization techniques is provided, and it is showed that, LCM achieves higher schedulability for larger atomic section lengths than that with past CMs, and also when compared to the retry loop-length of lock-free synchronization. Presented work is analytical as the questions we try to answer is analytical, so our results. Experimental evaluation will be done in future work.

1. INTRODUCTION

Embedded systems sense physical processes and control their behavior, typically through feedback loops. Since physical processes are concurrent, computations that control them must also be concurrent, enabling them to process multiple streams of sensor input and control multiple actuators, all concurrently. Often, such computations need to concurrently read/write shared data objects. Typically, they must also process sensor input and react in a timely manner.

The de facto standard for programming concurrency is the threads abstraction, and the de facto synchronization abstraction is locks. Lock-based concurrency control has significant programmability, scalability, and compositionality challenges [Herlihy 2006]. Transactional memory (TM) is an alternative synchronization model for shared in-memory data objects that promises to alleviate these difficulties. With TM, programmers write concurrent code using threads, but organize code that read/write shared objects as transactions, which appear to execute atomically. Two transactions conflict if they access the same object and one access is a write. When that happens, a contention manager (or CM) [Guerraoui et al. 2005] resolves the conflict by aborting one and allowing the other to proceed to commit, yielding (the illusion of) atomicity. Aborted transactions are re-started, often immediately. In addition to a simple programming model, TM provides performance comparable or superior to highly concurrent fine-grained locking and lock-free approaches [Saha et al. 2006], and is composable [Harris et al. 2005]. Multiprocessor TM has been proposed in hardware, called HTM (e.g., [McDonald 2009]), and in software, called STM (e.g., [Shavit and Touitou 1995]), with the usual tradeoffs: HTM provides strong atomicity [McDonald 2009], has lesser overhead, but needs transactional support in hardware; STM is available on any hardware.

Given STM's programmability, scalability, and compositionality advantages, we consider it for concurrency control in multicore embedded real-time software. Doing so will require bounding transactional retries, as real-time threads, which subsume transactions, must satisfy time constraints. Retry bounds in STM are dependent on the CM policy at hand (analogous to the way thread response time bounds are scheduler-dependent). Thus, real-time CM is logical.

Designing a real-time CM can be straightforward. Transactional contention can be resolved using dynamic or fixed priorities of parent threads, resulting in Earliest-Deadline-First (EDF) CM or Rate Monotonic Assignment (RMA)-based CM, respectively. But what upper bounds exist for transactional retries and thread response times under such CMs and respective multicore real-time schedulers, global EDF (G-EDF) and global RMA (G-RMA)? As lock-free protocols and STM do not use locks, how do they compare to each other? i.e., are there upper or lower bounds for transaction lengths below or above which is STM superior to lock-free?

We answer these questions. We consider EDF and RMA CMs, and establish their retry and response time upper bounds, and the conditions under which they outperform lock-free protocols. Our work reveals a key result: for most cases, for G-EDF/EDF CM (denoted as ECM) and G-RMA/RMA CM (denoted as RCM) to be better or as good as lock-free, the atomic section length under STM must not exceed half of the lock-free retry loop-length. However, in some cases, for G-EDF/EDF CM, the atomic section length can reach the lock-free retry loop-length, and for G-RMA/RMA CM, it can even be larger than the lock-free retry loop-length. This means that, STM is more advantageous with G-RMA than with G-EDF. These results, among others, for the first time, provide a fundamental understanding of when to use, and not use, STM concurrency control in multicore embedded real-time software, and constitute the paper's contribution.

Based on the analysis of the previous real time CMs, a fundamental question is raised: is it possible to increase the atomic section length by an alternative CM design, so that STM's schedulability advantage has a larger coverage? We answer this question by designing a novel CM that can be used with both dynamic and fixed priority (global) multicore real-time schedulers: length-based CM or LCM (Section 6.1). LCM resolves conflicts based on the priority of conflicting jobs, besides the length of the interfering atomic section, and the normalized length of the interfered atomic section. We establish LCM's retry and response time upper bounds, when used with G-EDF and with G-RMA schedulers. We identify the conditions under which G-EDF/LCM and G-RMA/LCM outperform lock-free protocols, as well as, ECM and RCM.

We show that, LCM achieves higher schedulability under larger transaction length than that under ECM, and increases the minimum upper limit on the ratio between transaction length and retry-loop length from that under RCM. This greater schedulability advantage of STM thus allows programmers to reap STM's significant programmability and composability benefits for a broader range of multicore real-time software than what was previously possible.

We overview past and related efforts in Section 2. Section 3 outlines the work's preliminaries. Sections 4 and 5 establish response time bounds under G-EDF/EDF CM and G-RMA/RMA CM, respectively. Section 6.1 presents the design rationale of LCM. Section 6.2 establishes upper bounds on transactional retries and response times under G-EDF/LCM, and Section 6.3 compares its schedulability with ECM. Sections 6.4 and 6.5, respectively, do the same for G-RMA/LCM. We compare STM against lock-free approaches in Section 7. We conclude in Section 8.

2. RELATED WORK

Transactional-like concurrency control without using locks, for real-time systems, has been previously studied in the context of non-blocking data structures (e.g., [Anderson et al. 1995]). Despite their numerous advantages over locks (e.g., deadlock-freedom), their programmability has remained a challenge. Past studies show that they are best suited for simple data structures where their retry cost is competitive to the cost of lock-based synchronization [Brandenburg et al. 2008]. In contrast, STM is semantically simpler [Herlihy 2006], and is often the only viable lock-free solution for complex data structures (e.g., red/black tree) [Fahmy 2010] and nested critical sections [Saha et al. 2006].

STM concurrency control for real-time systems has been previously studied in [Manson et al. 2006; Fahmy et al. 2009; Sarni et al. 2009; Schoeberl et al. 2010; Fahmy 2010; Barros and Pinho 2011].

[Manson et al. 2006] proposes a restricted version of STM for uniprocessors. Uniprocessors do not need contention management.

[Fahmy et al. 2009] bounds response times in distributed multiprocessor systems with STM synchronization. They consider Pfair scheduling, limited to small atomic regions with fixed size, and limit transaction execution to span at most two quanta. In contrast, we allow atomic regions with arbitrary duration.

[Sarni et al. 2009] presents real-time scheduling of transactions and serializes transactions based on deadlines. However, the work does not bound retries and response times, nor establishes tradeoffs against lock-free approach. In contrast, we establish such bounds and tradeoffs.

[Schoeberl et al. 2010] proposes real-time HTM, unlike real-time STM that we consider. The work does not describe how transactional conflicts are resolved. In contrast, we show how task response times can be met using different conflict resolution policies. Besides, the retry bound developed in [Schoeberl et al. 2010] assumes that the worst case conflict between atomic sections of different tasks occurs when the sections are released at the same time. However, we show that this is not the worst case. We develop retry and response time upper bounds based on much worse conditions.

The past work that is closest to ours is [Fahmy 2010], which upper bounds retries and response times for EDF CM with G-EDF, and identify the tradeoffs against locking and lock-free protocols. Similar to [Schoeberl et al. 2010], [Fahmy 2010] also assumes that the worst case conflict between atomic sections occurs when the sections are released simultaneously. In addition, we consider RMA CM, besides EDF CM.

The ideas in [Fahmy 2010] are extended in [Barros and Pinho 2011], which presents three real time CM designs. But no retry bounds nor schedulability analysis are presented for those CMs.

3. PRELIMINARIES

We consider a multiprocessor system with m identical processors and n sporadic tasks $\tau_1, \tau_2, \ldots, \tau_n$. The k^{th} instance (or job) of a task τ_i is denoted τ_i^k. Each task τ_i is specified by its worst case execution time (WCET) c_i, its minimum period T_i between any two consecutive instances, and its relative deadline D_i, where $D_i = T_i$. Job τ_i^j is released at time r_i^j and must finish no later than its absolute deadline $d_i^j = r_i^j + D_i$. Under a fixed priority scheduler such as G-RMA, p_i determines τ_i's (fixed) priority and it is constant for all instances of τ_i. Under a dynamic priority scheduler such as G-EDF, a τ_i^j's priority, p_i^j, is determined by its absolute deadline. A task τ_j may interfere with task τ_i for a number of times during a duration L, and this number is denoted as $G_{ij}(L)$. τ_j's workload that interferes with τ_i during L is denoted $W_{ij}(L)$.

Shared objects. A task may need to access (i.e., read, write) shared, in-memory objects while it is executing any of its atomic sections, which are synchronized using STM. The set of atomic sections of task τ_i is denoted s_i. s_i^k is the k^{th} atomic section of τ_i. Each object, θ, can be accessed by multiple tasks. The set of objects accessed by τ_i is θ_i without repeating objects. The set of atomic sections used by τ_i to access θ is $s_i(\theta)$, and the sum of the lengths of those atomic sections is $len(s_i(\theta))$. $s_i^k(\theta)$ is the k^{th} atomic section of τ_i that accesses θ. $s_i^k(\theta)$ executes for a duration $len(s_i^k(\theta))$. If θ is shared by multiple tasks, then $s(\theta)$ is the set of atomic sections of all tasks accessing θ, and the set of tasks sharing θ with τ_i is denoted $\gamma_i(\theta)$. Atomic sections are non-nested, and each atomic section is assumed to access only one object to enable comparison with retry-loop lock-free approach [Devi et al. 2006] (if an atomic section is allowed to access more than one object, more complex lock-free approaches will be needed).

The maximum-length atomic section in τ_i that accesses θ is denoted $s_{i_{max}}(\theta)$, while the maximum one among all tasks is $s_{max}(\theta)$, and the maximum one among tasks with priorities lower than that of τ_i is $s_{max}^i(\theta)$.

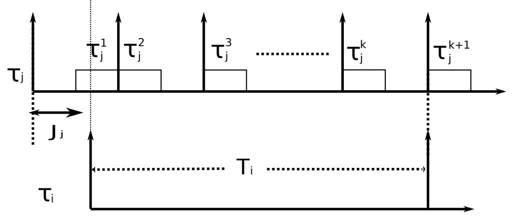


Fig. 1. Maximum interference between two tasks under G-EDF

STM retry cost. If two or more atomic sections conflict, the CM will commit one section and abort and retry the others, increasing the time to execute the aborted sections. The increased time that an atomic section $s_i^p(\theta)$ will take to execute due to interference with another section $s_j^k(\theta)$, is denoted $W_i^p(s_j^k(\theta))$.

The total time that a task τ_i's atomic sections have to retry is denoted $RC(\tau_i)$. When this retry cost is calculated over the task period T_i or an interval L, it is denoted, respectively, as $RC(T_i)$ and $RC(L)$.

4. 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. An atomic section of τ_i, $s_i^k(\theta)$, is aborted and retried by a conflicting atomic section of τ_j, $s_j^l(\theta)$, if $d_j \leq d_i$, by the EDF CM. 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 [Bertogna and Cirinei 2007] and is shown in Figure 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 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:

$$\begin{aligned} W_{ij}\left(T_i\right) &\leq \left\lfloor \frac{T_i}{T_j} \right\rfloor c_j + min\left(c_j, T_i - \left\lfloor \frac{T_i}{T_j} \right\rfloor T_j\right) \\ &\leq \left\lceil \frac{T_i}{T_j} \right\rceil c_j \end{aligned} \tag{1}$$

For an interval $L < T_i$, the worst case pattern of interference is shown in Figure 2 where 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, and the workload of τ_j is:

$$W_{ij}\left(L\right) \leq \left(\left\lceil \frac{L - c_j}{T_j} \right\rceil + 1\right) c_j \tag{2}$$

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

$$W_{ij}\left(R\right) = min\left(W_{ij}\left(R\right), W_{ij}\left(T_i\right)\right) \tag{3}$$

where $W_{ij}(R)$ is calculated by (2) if $R < T_i$, otherwise, it is calculated by (1).

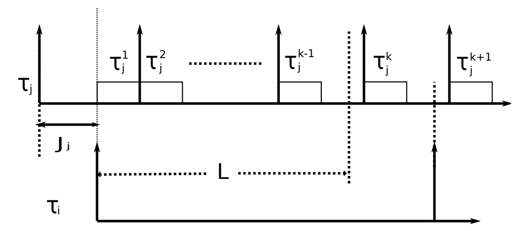
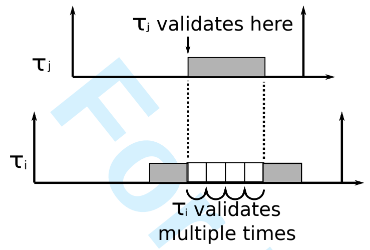
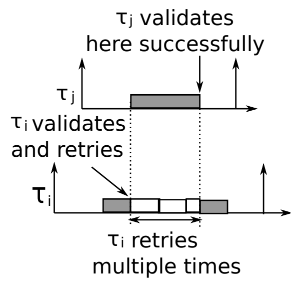


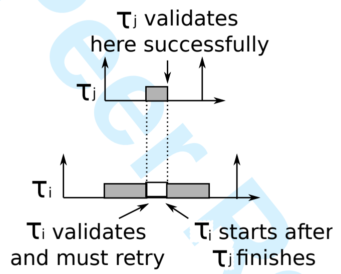
Fig. 2. Maximum interference during part L of T_i



(a) Early validation



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



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

Fig. 3. Retry of $s_i^k(\theta)$ due to $s_j^l(\theta)$

4.1. Retry Cost of Atomic Sections

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

$$RC\left(T_i\right) \leq \sum_{\theta \in \theta_i} \left(\Big(\sum_{\tau_j \in \gamma(\theta)} \Big(\left\lceil \frac{T_i}{T_j} \right\rceil \sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta) + s_{max}(\theta)\big) \Big) \Big) - s_{max}(\theta) + s_{i_{max}}(\theta) \right) \quad (4)$$

PROOF. Given 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 its end of the 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 [McDonald 2009]: a) eager (pessimistic), in which conflicts are detected at access time, 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 interferred by another conflicting atomic section, as shown in Figure 3.

In Figure 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 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 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 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 to 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 4(a) and 4(b). In Figure 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 share 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,

$$W_i^p\left(s_j^k(\theta)\right) \le len\left(s_j^k(\theta) + s_{max}(\theta)\right) \tag{5}$$

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:

$$RC(T_i) \le \sum_{\theta\in\theta_i}\sum_{\tau_j\in\gamma_i(\theta)}\left(\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta)+s_{max}(\theta)\big)\right) \tag{6}$$

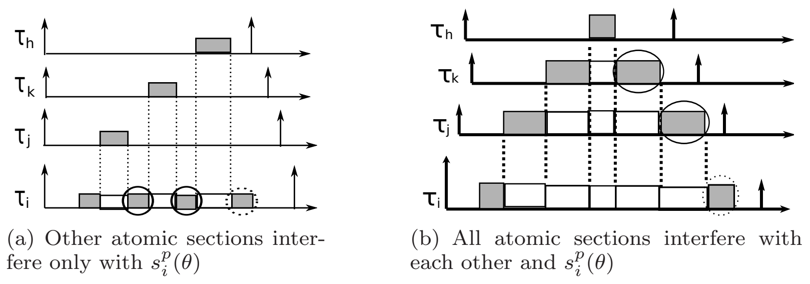
Here, $\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta)+s_{max}(\theta)\right)$ is the contribution of all instances of τ_j during T_i. This contribution is added to all tasks. The last atomic section to execute is $s_i^p(\theta)$ (τ_i's atomic section that was 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:*

$$RC(T_i) \le \sum_{\theta\in\theta_i} min(\Phi_1,\Phi_2) \tag{7}$$

where Φ_1 is calculated by (4) for one object θ (not the sum of $\theta\in\theta_i$), and

$$\Phi_2 = \left(\sum_{\tau_j\in\gamma_i(\theta)}\left(\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)} len\big(s_j^l(\theta)+s^*_{max}(\theta)\big)\right)\right) - \bar{s}_{max}(\theta) + s_{i_{max}}(\theta) \tag{8}$$



(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 $s_{i_{max}}(\theta)$

Fig. 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. (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, (5) becomes:

$$W_i^p\left(s_j^k(\theta)\right) \le len\left(s_j^k(\theta) + s^*_{max}(\theta)\right) \tag{9}$$

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 5, where the maximum atomic section of each task that accesses θ is associated with its corresponding task. According to (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.

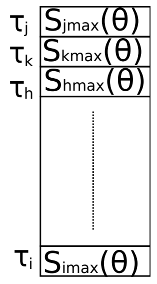


Fig. 5. Values associated with $s^*_{max}(\theta)$

This means that the subtracted $s_{max}(\theta)$ in (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 in 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 $RC(T_i)$. Claim follows. $\square$

4.2. 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 for 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:

$$c_{ji} = c_j - \left(\sum_{\theta \in (\theta_j \wedge \theta_i)} len\left(s_j(\theta)\right) \right) + RC(T_{ji}) \tag{10}$$

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 [Bertogna and Cirinei 2007], as follows:

$$R_i^{up} = c_i + RC(T_i) + \left\lfloor \frac{1}{m} \sum_{j \neq i} W_{ij}(R_i^{up}) \right\rfloor \tag{11}$$

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

$W_{ij}(R_i^{up})$ is calculated by (3), and $W_{ij}(T_i)$ is calculated by (1), with c_j replaced by c_{ji}, and changing (2) as:

$$W_{ij}(L) = max \begin{cases} \left(\left\lceil \frac{L - \left(c_{ji} + \sum_{\theta \in (\theta_j \wedge \theta_i)} len(s_j(\theta))\right)}{T_j} \right\rceil + 1 \right) c_{ji} \\ \left\lceil \frac{L - c_j}{T_j} \right\rceil .c_{ji} + c_j - \sum_{\theta \in (\theta_j \wedge \theta_i)} len(s_j(\theta)) \end{cases} \tag{12}$$

(12) compares between two terms, as we have two cases:

Case 1. τ_j^1 (shown in Figure 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 (12), because they differ in the contribution of their τ_j^1s, and the number of instances after that.

4.2.1. 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 (11). So, (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 6 can happen.

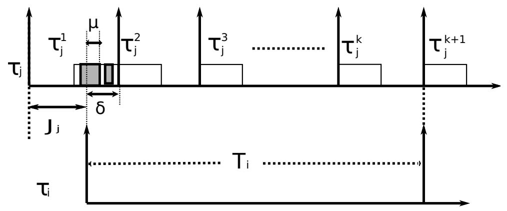


Fig. 6. Atomic sections of job τ_j^1 contributing to period T_i

To counter the situation in Figure 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:

— $\lambda_1(j,\theta) = \sum_{\forall s_j^l(\theta) \in [d_j^1-\delta, d_j^1]} len\left(s_j^{l^*}(\theta) + s_{max}(\theta)\right)$

— $\chi_1(i,j,\theta) = \left\lfloor \frac{T_i}{T_j} \right\rfloor \sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta) + s_{max}(\theta)\right)$

— $\lambda_2(j,\theta) = \sum_{\forall s_j^l(\theta) \in [d_j^1-\delta, d_j^1]} len\left(s_j^{l^*}(\theta) + s^*_{max}(\theta)\right)$

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

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

Now, (7) can be modified as:

$$RC(T_i) \leq \sum_{\theta \in \theta_i} min \begin{cases} \left(\left(\sum_{\tau_j \in \gamma_i(\theta)} \lambda_1(j,\theta) + \chi_1(i,j,\theta)\right) - s_{max}(\theta) + s_{i_{max}}(\theta)\right) \\ \left(\left(\sum_{\tau_j \in \gamma_i(\theta)} \lambda_2(j,\theta) + \chi_2(i,j,\theta)\right) - \bar{s}_{max}(\theta) + s_{i_{max}}(\theta)\right) \end{cases} \tag{13}$$

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

— $\upsilon(L,j) = \left\lceil \frac{L-c_j}{T_j} \right\rceil + 1$

— $\lambda_3(j,\theta) = \sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta) + s_{max}(\theta)\right)$

— $\lambda_4(j,\theta) = \sum_{\forall s_j^l(\theta)} len\left(s_j^l(\theta) + s^*_{max}(\theta)\right)$

Now, (7) becomes:

$$RC(L) \leq \sum_{\theta \in \theta_i} min \begin{cases} \left(\sum_{\tau_j \in \gamma_i(\theta)} (\upsilon(L,j)\lambda_3(j,\theta))\right) - s_{max}(\theta) + s_{i_{max}}(\theta) \\ \left(\sum_{\tau_j \in \gamma_i(\theta)} (\upsilon(L,j)\lambda_4(j,\theta))\right) - \bar{s}_{max}(\theta) + s_{i_{max}}(\theta) \end{cases} \tag{14}$$

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

$$RC(R_i^{up}) \leq min \begin{cases} RC(R_i^{up}) \\ RC(T_i) \end{cases} \tag{15}$$

Where $RC(R_i^{up})$ is calculated by (14) if R_i^{up} does not extend to the last interfering instance of τ_j, otherwise, it is calculated by (13). The final upper bound on τ_i's response time can be calculated as in (11) by replacing $RC(T_i)$ with $RC(R_i^{up})$.

5. 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. We will use *RCM* to refer to a multiprocessor system scheduled by G-RMA and resolves STM conflicts by the RMA CM.

5.1. Maximum Task Interference

Figure 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 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.

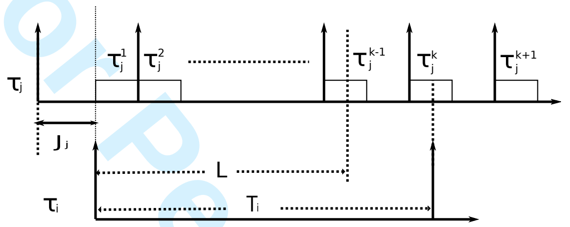


Fig. 7. Max interference of T_j to T_i in G-RMA

Thus, the maximum contribution of τ_j^b to τ_i^a for any duration L can be deduced from Figure 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. In contrast to ECM where L cannot be extended directly to T_i, as this will have a different pattern of worst case interference from other tasks.

5.2. 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:*

$$RC\left(L\right) \leq \sum_{\theta \in \theta_i} \left(\left(\sum_{\tau_j^*} \left(\left(\left\lceil \frac{L-c_j}{T_j} \right\rceil + 1 \right) \pi\left(j,\theta\right) \right) \right) - s_{max}^j\left(\theta\right) + s_{i_{max}}\left(\theta\right) \right) \tag{16}$$

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

PROOF. Since 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. So, (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 (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 $s_{max}^j(\theta)$, which is the maximum-length atomic section of tasks with priority lower than τ_j and share object θ with τ_i. 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^*). □

5.3. Upper Bound on Response Time

The response time upper bound can be computed by Theorem 7 in [Bertogna and Cirinei 2007] with some modification to include the effect of retry cost. Thus, this upper bound is given by:

$$R_i^{up} = c_i + RC(R_i^{up}) + \left\lfloor \frac{1}{m} \sum_{j \neq i} W_{ij}(R_i^{up}) \right\rfloor \tag{17}$$

where $W_{ij}(R_i^{up})$ is calculated as in (12), c_{ji} is calculated by (10), and RC is calculated by (16).

6. LENGTH-BASED CONTENTION MANAGER

LCM resolves conflicts based on the priority of conflicting jobs, besides the length of the interfering atomic section, and the normalized length of the interfered atomic section. So, its design is less straight forward than ECM and RCM as they depend only on the priority of the conflicting jobs. This modification in design allows lower priority jobs, under LCM, to retry for less time than ECM and RCM, but higher priority jobs, sometimes, wait for lower priority ones, but this will not result in indirect blocking as will be illustrated.

6.1. LCM: 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 task τ_j than τ_i — interferes with $s_i^k(\theta)$ at the end of its execution, while $s_i^k(\theta)$ is just about to commit. Thus, LCM 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 $s_i^k(\theta)$ is the one which interferes with $s_j^l(\theta)$, then $s_j^l(\theta)$ commits because it belongs to the higher priority job and it started before $s_i^k(\theta)$.

We assume that

$$len(s_j^l(\theta)) = c_m len(s_i^k(\theta)) \tag{18}$$

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

On the other side, 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 abortion opportunity is increased even if $s_i^k(\theta)$ is close to its end of execution. To decide whether $s_i^k(\theta)$ should be aborted or not, we use a threshold value $\psi \in [0, 1]$, that determines the length percentage of $s_i^k(\theta)$ below which $s_i^k(\theta)$ will abort due to $s_j^l(\theta)$. This percentage value is denoted α^{jl} as it differs according to s_j^l. If the abort percent is 0, it means not to abort, and 1 means to abort.

The behavior of LCM is illustrated in Figure 8. The figure represents five different lengths of $s_j^l(\theta)$ interfering with $s_i^k(\theta)$ at all points of $s_i^k(\theta)$. For a specific curve (which means a specific length for $s_j^l(\theta)$), ψ determines the percentage of $len(s_i^k(\theta))$ below which $s_i^k(\theta)$ will be aborted. For example, for $len(s_j^l(\theta)) = 0.1 \times len(s_i^k(\theta))$, $s_i^k(\theta)$ will be aborted by $s_j^l(\theta)$ if the latter interferes with $s_i^k(\theta)$ no later than $s_i^k(\theta)$ reaches $\alpha 3$ percentage of its length ($\alpha 3$ is shown in Figure 8). After that, $s_j^l(\theta)$ will have to retry. As $len(s_j^k(\theta))$ decreases, the

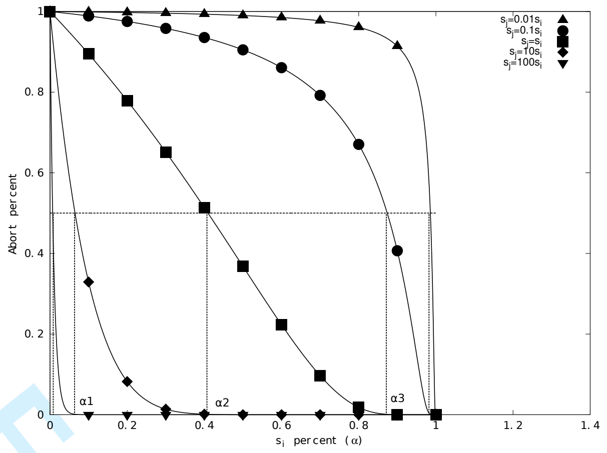


Fig. 8. Interference of $s_i^k(\theta)$ by various lengths of $s_j^l(\theta)$

opportunity that it will abort $s_i^k(\theta)$ at a higher percentage α_{max} increases (as illustrated in Figure 8, $\alpha 3 > \alpha 2 > \alpha 1$ for reduced length of $s_j^l(\theta)$). The function that is used to represent the different curves in Figure 8 is:

$$f(c_m, \alpha) = e^{\frac{-c_m \alpha}{1-\alpha}} \tag{19}$$

where c_m is fixed for a specific curve and is calculated by (18), but α changes along each curve, with a specific value of α corresponds to ψ. This function achieves the desired requirement that the abortion opportunity is reduced as $s_i^k(\theta)$ gets closer to its end of execution (as $\alpha \to 1$, $f(c_m, 1) \to 0$), or as the length of the conflicting transaction is large (as $c_m \to \infty$, $f(\infty, \alpha) \to 0$). Meanwhile, this abortion opportunity is increased as $s_i^k(\theta)$ is interfered closer to its release (as $\alpha \to 0$, $f(c_m, 0) \to 1$), or as the length of the conflicting transaction decreases (as $c_m \to 0$, $f(0, \alpha) \to 1$). Note that all lengths of $s_i^k(\theta)$ are normalized to the same unit length. This way, different values of $s_j^l(\theta)$ interfere with different lengths of $s_i^k(\theta)$ at the same percentage α^{jl}, but the actual length of the interference for different lengths of $s_i^k(\theta)$ differ according to $len(s_i^k(\theta))$ i.e., let $len(s_i^k(\theta)) \neq len(s_i^{k+1}(\theta))$. Then, for one $s_j^l(\theta)$, both $s_i^k(\theta)$ and $s_i^{k+1}(\theta)$ will be interfered at the same value α^{jl}, but $\alpha^{jl} len(s_i^k(\theta))$ differs from $\alpha^{jl} len(s_i^{k+1}(\theta))$. The normalization of different lengths of $s_i^k(\theta)$ is done to simplify calculations.

As $s_j^l(\theta)$ belongs to a higher priority job than $s_i^k(\theta)$, if $s_j^l(\theta)$ starts before or at the same start time of $s_i^k(\theta)$, then $s_i^k(\theta)$ will have to abort and retry until $s_j^l(\theta)$ finishes execution. But if $s_j^l(\theta)$ starts after $s_i^k(\theta)$, then the comparison illustrated previously will be applied.

CLAIM 4. *Let $s_j^l(\theta)$ interfere with $s_i^k(\theta)$ at α^{jl} percentage which corresponds to the threshold value ψ. Then, the maximum contribution of $s_j^l(\theta)$ to the retry cost of $s_i^k(\theta)$ is:*

$$W_i^k(s_j^l(\theta)) \leq \alpha^{jl} len\Big(s_i^k(\theta)\Big) + len\Big(s_j^l(\theta)\Big) \tag{20}$$

PROOF. If $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ at a Υ percentage, where $\Upsilon < \alpha^{jl}$, then the retry cost of $s_i^k(\theta)$ will be $\Upsilon len(s_i^k(\theta)) + len(s_j^l(\theta))$, which is lower than that calculated in (20). Besides, if $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ after α^{jl} percentage, then $s_i^k(\theta)$ will not abort. □

Claim 5. *An atomic section of a higher priority task, τ_j, may have to abort and retry due to a lower priority task, τ_i, if $s_j^l(\theta)$ interferes with $s_i^k(\theta)$ after the α^{jl} percentage. This retrial time of τ_j, due to $s_i^k(\theta)$ and $s_j^l(\theta)$, is upper bounded by:*

$$W_j^l(s_i^k(\theta)) \leq \Big(1 - \alpha^{jl}\Big) len\Big(s_i^k(\theta)\Big) \tag{21}$$

Proof. It is derived directly from Claim 4, as $s_j^l(\theta)$ will have to retry for the remaining length of $s_i^k(\theta)$. □

Claim 6. *There is no indirect blocking in LCM.*

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^{jl} that corresponds to ψ. 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^{ab} that corresponds to the threshold value ψ. So, $s_a^b(\theta)$ is allowed abort $s_i^k(\theta)$. LCM will choose $s_j^l(\theta)$ 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, and when it is time to compare between $s_j^l(\theta)$ and $s_a^b(\theta)$, $s_j^l(\theta)$ will be chosen because it is of higher priority. So, τ_j is not indirectly blocked by τ_a.
— $p_a > p_j > p_i$, then if $s_a^b(\theta)$ is chosen to commit, this is not indirect blocking 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, otherwise, $s_j^l(\theta)$ will retry, but this will not be indirect blocking because τ_a is already of higher priority than τ_j.

So, it is seen that there is no indirect blocking by LCM. Claim follows. □

6.2. Response time of G-EDF/LCM

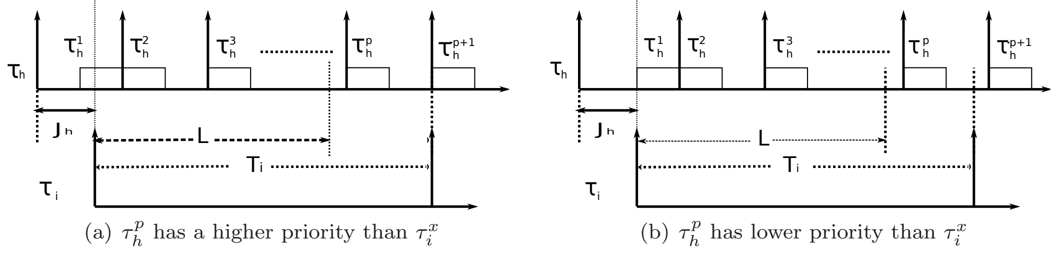
It is desired to determine the response time when LCM is used with G-EDF. So, the following claims are introduced.

Claim 7. *When all instances of τ_h interfering with one instance of τ_i, τ_i^x, are of higher priority than τ_i^x (as shown in Figure 9(a)), then the retry cost of τ_i over T_i due to these instances of τ_h is upper bounded by*

$$\Phi_i(h) = \sum_{\theta \in \theta_i \wedge \theta_h} \left(\left\lceil \frac{T_i}{T_h} \right\rceil \sum_{\forall s_h^l(\theta)} len\Big(s_h^l(\theta)\Big) + \alpha_{max}^{hl} len\Big(s_{max}^*(\theta)\Big) \right) \tag{22}$$

where $s_{max}^*(\theta)$ is the maximum length atomic section, not associated with τ_h, that accesses θ, and α_{max}^{hl} is the maximum percentage of $len(s_{max}^*(\theta))$ during which $s_h^l(\theta)$ causes $s_{max}^*(\theta)$ to retry.

Proof. If the absolute deadline of one instance of τ_h, τ_h^p as shown in Figure 9(a), coincides with the absolute deadline of τ_i^x, then all interfering instances of τ_h with τ_i^x will have a higher priority than τ_i^x, and Claim 4 will be used to determine the retry cost of each atomic section in τ_i^x due to atomic sections in τ_h. By combining Claim 2, Claim follows. □



(a) τ_h^p has a higher priority than τ_i^x (b) τ_h^p has lower priority than τ_i^x

Fig. 9. Interference to job τ_i by higher and lower priority jobs

CLAIM 8. *For an instance τ_i^x interfered by multiple instances of τ_h, and the last interfering instance, τ_h^p, has a higher absolute deadline than τ_i^x as shown in Figure 9(b). Then the retry cost of τ_i due to interfering instances of τ_h over T_i is calculated as*

$$\Phi_i^*(h) = \sum_{\theta \in \theta_i \wedge \theta_h} \left(\left\lfloor \frac{T_i}{T_h} \right\rfloor \sum_{\forall s_h^l(\theta)} len\left(s_h^l(\theta)\right) + \alpha_{max}^{hl} len\left(s_{max}^*(\theta)\right) \right) + \sum_{\forall s_i^y(\theta)} \left(1 - \alpha_{max}^{iy}\right) len\left(s_{h_{max}}(\theta)\right) \tag{23}$$

where the first sum is the same as that calculated by (22) except it only includes the contribution of instance τ_h^1 to τ_h^{p-1} instead of instances τ_h^1 to τ_h^p, the second sum is the upper bound on retry cost of τ_i^x due to atomic section in τ_h^p, and α_{max}^{iy} is the maximum percentage of $len(s_{h_{max}}(\theta))$ after which $s_i^y(\theta)$ will be forced to abort and retry.

PROOF. Under G-EDF, there can only be at most one instance of τ_h, τ_h^p, interfering with τ_i^x, that can have a lower priority (i.e., larger absolute deadline) than τ_i^x. This is obtained by shifting Figure 9(a) to the right to give Figure 9(b). While τ_h^p cannot affect the non-atomic operation of τ_i^x, because of its lower priority, τ_h^p can abort and retry atomic sections of τ_i^x. So, Claim 4 is used to calculate retry cost of τ_i^x due to instances τ_h^1 to τ_h^{p-1}, and Claim 5 is used to calculate retry cost of τ_i^x due to τ_h^p. Since τ_i^x's priority is higher than that of τ_h^p, any atomic section $s_i^y(\theta)$ in τ_i^x can be aborted by only one conflicting atomic section, $s_h^z(\theta)$, of τ_h^p. This is because, after $s_h^z(\theta)$, $s_i^y(\theta)$ will start at most at the same time as any further conflicting atomic section in τ_h^z, and since $s_i^y(\theta)$ belongs to a higher priority job, LCM will commit it first. This means that $s_i^y(\theta)$ cannot be blocked by two or more atomic sections of τ_h^p. On the other hand, one atomic section in τ_h^p, $s_h^z(\theta)$, can block multiple atomic sections in τ_i^x, because any atomic section with a suitable length in any other task can cause $s_h^z(\theta)$ to retry multiple times, causing multiple atomic sections in τ_i^x to interfere with $s_h^z(\theta)$. Claim follows. □

CLAIM 9. *The total retry cost of τ_i due to all other tasks $\tau_h \in \gamma_i$ over T_i is upper bounded by:*

$$RC(T_i) \leq \sum_{\forall \tau_h \in \gamma_i} max\{\Phi_i(h), \Phi_i^*(h)\} \tag{24}$$

PROOF. The maximum contribution of each conflicting task τ_h with τ_i during T_i (which is the maximum of (22) and (23)) are summed together to give the total retry cost of τ_i. Claim follows. □

Response time of τ_i is calculated by (11) where $RC(T_i)$ is replaced by $CO(T_i)$.

6.3. Schedulability comparison 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 against that of G-EDF/LCM. In each method (including G-EDF/LCM), we inflate the c_i for each 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 follows:

$$\sum_{\forall \tau_i} \frac{c_i + RC_A(T_i)}{T_i} \leq \sum_{\forall \tau_i} \frac{c_i + RC_B(T_i)}{T_i}$$
$$\sum_{\forall \tau_i} \frac{RC_A(T_i)}{T_i} \leq \sum_{\forall \tau_i} \frac{RC_B(T_i)}{T_i} \tag{25}$$

Thus, schedulability is compared by substituting the retry cost added by synchronization methods in (25).

Claim 10. *Let s_{max} be the maximum length atomic section accessing any object θ. Let α_{max} and α_{min} be the maximum and minimum percentages of normalized atomic section s_i during which s_i will abort and retry by the minimum and maximum length atomic sections, respectively. Schedulability of G-EDF/LCM is equal or better than that of ECM if for any task τ_i and τ_h:*

$$\frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \left\lceil \frac{T_i}{T_h} \right\rceil \tag{26}$$

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

$$RC(T_i) \leq \sum_{\forall \tau_h \in \gamma_i} \sum_{\forall \theta \in \ (\theta_i \wedge \theta_h)} \left(\left\lceil \frac{T_i}{T_h} \right\rceil \sum_{\forall s_h^z(\theta)} 2len(s_{max}) \right) \tag{27}$$

with the assumption that all all lengths of atomic sections in (4) are replaced by s_{max}. With the same assumption, there will be only one curve of $len(s_{max})$ in Figure 8, which represents all atomic sections interfering with s_i.

For retry cost caused by higher priority tasks to τ_i, all αs will be replaced with the one that has the maximum percentage of s_i, α_{max}, which results from interference of the minimum length atomic section to s_i.

For retry cost caused by lower priority tasks to τ_i, α will be replaced with the minimum one, α_{min}, which results from interference of the maximum length atomic section to s_i. α_{max}, α_{min}, and $len(s_{max})$ are all constants. Thus, in G-EDF/LCM, (22) is upper bounded by:

$$\Phi_i(h) \leq \sum_{\forall \theta \in (\theta_i \wedge \theta_h)} \left(\left\lceil \frac{T_i}{T_h} \right\rceil \sum_{\forall s_h^z(\theta)} (1+\alpha_{max}) len(s_{max}) \right) \tag{28}$$

and (23) is upper bounded by:

$$\Phi_h^*(T_i) \leq \sum_{\forall \theta \in (\theta_i \wedge \theta_h)} \left(\sum_{\forall s_i^y(\theta)} \Big((1-\alpha_{min}) len(s_{max}) \Big) + \left\lfloor \frac{T_i}{T_h} \right\rfloor \sum_{\forall s_h^z(\theta)} \Big((1+\alpha_{max}) len(s_{max}) \Big) \right) \tag{29}$$

$RC(T_i)$ is calculated by (24). If β_1 is the total number of times any instance of τ_h accesses shared objects with τ_i, then $\beta_1 = \sum_{\theta \in (\theta_i \wedge \theta_h)} \sum_{\forall s_h^z(\theta)}$. And if β_2 is the total number of times any instance of τ_i accesses shared objects with τ_h, $\beta_2 = \sum_{\theta \in (\theta_i \wedge \theta_h)} \sum_{\forall s_i^y(\theta)}$. Then $\beta_{i,h} = max(\beta_1, \beta_2)$ is the maximum number of accesses to all shared objects by any instance of τ_i or τ_h. Thus, (27) becomes:

$$RC(T_i) \leq \sum_{\tau_h \in \gamma_i} 2 \left\lceil \frac{T_i}{T_h} \right\rceil \beta_{i,h} s_{max} \tag{30}$$

and (28) becomes:

$$\Phi_i(h) \leq \left\lceil \frac{T_i}{T_h} \right\rceil \beta_{i,h}(1+\alpha_{max})len(s_{max}) \tag{31}$$

and (29) becomes:

$$\Phi_h^*(T_i) \leq \beta_{i,h} len(s_{max}) \left((1-\alpha_{min}) + \left\lfloor \frac{T_i}{T_h} \right\rfloor (1+\alpha_{max}) \right) \tag{32}$$

The retry cost of τ_i in (31) and (32) can be combined into one equation that represents an upper bound for both of them . This upper bound for the retry cost of τ_i due to τ_h is:

$$RC_h(T_i) = \left((1-\alpha_{min}) + \left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right) \beta_{i,h} s_{max} \tag{33}$$

We can now compare the total utilization of G-EDF/LCM with that of ECM:

$$\forall \tau_i \frac{\sum_{\forall \tau_h \in \gamma_i} \left((1-\alpha_{min}) + \left\lceil \frac{T_i}{T_h} \right\rceil (1+\alpha_{max}) \right) \beta_{i,h}}{T_i} \leq \forall \tau_i \frac{\sum_{\forall \tau_h \in \gamma_i} 2 \left\lceil \frac{T_i}{T_h} \right\rceil \beta_{i,h}}{T_i} \tag{34}$$

(34) is satisfied if for each τ_i and τ_h, the following condition is satisfied:

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

$$\therefore \frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \left\lceil \frac{T_i}{T_h} \right\rceil$$

Claim follows. □

6.4. Response time of G-RMA/LCM

CLAIM 11. *Let $\lambda_2(j,\theta) = \sum_{\forall s_j^l(\theta)} len(s_j^l(\theta)) + \alpha_{max}^{jl} len(s_{max}^j(\theta))$, and $\chi_2(i,h,\theta) = \sum_{\forall s_i^y(\theta)} (1-\alpha_{max}^{iy}) len(s_{h_{max}}(\theta))$. Now, the retry cost of any task τ_i under G-RMA/LCM for any duration L is given by:*

$$RC(L) = \sum_{\forall \tau_j^*} \left(\sum_{\theta \in (\theta_i \wedge \theta_j)} \left(\left(\left\lceil \frac{L-c_j}{T_j} \right\rceil + 1 \right) \lambda_2(j,\theta) \right) \right) + \sum_{\forall \bar{\tau}_h} \left(\sum_{\theta \in (\theta_i \wedge \theta_h)} \left(\left(\left\lceil \frac{L-c_h}{T_h} \right\rceil + 1 \right) \chi_2(i,h,\theta) \right) \right) \tag{35}$$

where L can extend to T_i, $\tau_j^* = \{\tau_j | (\tau_j \in \gamma_i) \wedge (p_j > p_i)\}$, and $\bar{\tau}_h = \{\tau_h | (\tau_h \in \gamma_i) \wedge (p_h < p_i)\}$.

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. (20) will be used to determine the contribution of each conflicting atomic section in τ_j to τ_i. Meanwhile, all instances of any task, τ_h, with lower priority than τ_i, can conflict with τ_i during T_i, and (21) will be used to determine the contribution of each conflicting atomic section in τ_h to τ_i Besides, due to the fixed priority of all instances of τ_j^* and $\bar{\tau}_h$, the equations used to calculate the final cost and response time of τ_i over an interval L, can be directly extended to the interval T_i. Using the previous notations and Claim 3, Claim follows. □

The response time is calculated by (17).

6.5. Schedulability Comparison of G-RMA/LCM with RCM

CLAIM 12. *Under the same assumptions as that of Claims 10 and 11, G-RMA/LCM's schedulability is equal or better than that of RCM if:*

$$\frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \frac{\sum_{\forall\tau_i} \frac{\sum_{\forall\tau_j^*}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)\beta_{ij}}{T_i}}{2\sum_{\forall\tau_i}\frac{\sum_{\forall\bar{\tau}_h}\beta_{ih}}{T_i}} \tag{36}$$

PROOF. Under the same assumptions as that of Claims 10 and 11, (35) can be upper bounded as:

$$RC(T_i) \leq \sum_{\forall\tau_j^*}\left(\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)(1+\alpha_{max})len(s_{max})\beta_{ij}\right)+\sum_{\forall\bar{\tau}_h}\left(\left(\left\lceil\frac{T_i}{T_h}\right\rceil+1\right)(1-\alpha_{min})len(s_{max})\beta_{ih}\right) \tag{37}$$

For RCM, (16) for $RC(T_i)$ is upper bounded by:

$$RC(T_i) \leq \sum_{\forall T_j^*}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)2\beta_{ij}s_{max}$$

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

$$\sum_{\forall\tau_i}\frac{\sum_{\forall\bar{\tau}_h}\left(\left\lceil\frac{T_i}{T_h}\right\rceil+1\right)(1-\alpha_{min})\beta_{ih}}{T_i} \leq \sum_{\forall\tau_i}\frac{\sum_{\forall\tau_j^*}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)(1-\alpha_{max})\beta_{ij}}{T_i}$$

$$\therefore \frac{1-\alpha_{min}}{1-\alpha_{max}} \leq \frac{\sum_{\forall\tau_i}\frac{\sum_{\forall\tau_j^*}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)\beta_{ij}}{T_i}}{\sum_{\forall\tau_i}\frac{\sum_{\forall\bar{\tau}_h}\left(\left\lceil\frac{T_i}{T_h}\right\rceil+1\right)\beta_{ih}}{T_i}}$$

Since task relative deadline equals task period, and τ_h is of lower priority than τ_i, $T_i \leq T_h$. Therefore, $\left\lceil\frac{T_i}{T_h}\right\rceil = 1$ for any τ_i and τ_h. Claim follows. □

Thus, if blocking time for each task τ_i is reduced, then the right hand side of (36) is increased, giving a larger upper bound for $\frac{1-\alpha_{min}}{1-\alpha_{max}}$, and giving a wider range for α_{min} and α_{max} to choose from. Hence, by the proper choice of α_{max} and α_{min} in (36), schedulability of G-RMA/LCM can be better or equal to that of RCM.

7. STM VERSUS LOCK-FREE

We now would like to understand when STM will be beneficial compared to retry-loop lock-free approach [Devi et al. 2006]. This retry-loop lock-free approach is the most relevant to our work.

7.1. ECM versus Lock-Free

CLAIM 13. *For ECM's schedulability to be better or equal to that of [Devi et al. 2006]'s retry-loop lock-free approach, the size of s_{max} must not exceed one half of that of r_{max}; with low number of conflicting tasks, the size of s_{max} can be at most the size of r_{max}.*

PROOF. Equation (15) can be upper bounded as:

$$RC(T_i) \leq \sum_{\tau_j\in\gamma_i}\left(\sum_{\theta\in\theta_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil\sum_{\forall s_j^l(\theta)}(2.s_{max})\right)\right) \tag{38}$$

where $s_j^l(\theta)$, $s_{i_{max}}(\theta)$, $s_{max}^*(\theta)$, and $\bar{s}_{max}(\theta)$ 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, (38) can be modified as:

$$RC(T_i) = \alpha_{edf}.s_{max} \tag{39}$$

The loop retry cost is given by:

$$\begin{aligned} RL(T_i) &= \sum_{\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right).\beta_{i,j}.r_{max} \\ &= \alpha_{free}.r_{max} \end{aligned} \tag{40}$$

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, $\alpha_{free} = \sum_{\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right).\beta_{i,j}$, and r_{max} is the maximum execution cost of a single iteration of any retry loop of any task. 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 the lock-free system:

$$\begin{aligned} \sum_{\tau_i}\frac{c_i + \alpha_{edf}.s_{max}}{T_i} &\leq \sum_{\tau_i}\frac{c_i + \alpha_{free}.r_{max}}{T_i} \\ \therefore \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i}\alpha_{free}/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \end{aligned} \tag{41}$$

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:

$$\begin{aligned} \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i}(\bar{\alpha}_{free} + \hat{\alpha}_{free})/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \\ &= \frac{1}{2} + \frac{\sum_{\tau_i}\hat{\alpha}_{free}/T_i}{\sum_{\tau_i}\alpha_{edf}/T_i} \end{aligned} \tag{42}$$

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 (42) = 1, which is its maximum value. $T_j \geq T_i$ means that there is 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, (42) $\rightarrow 1/2$.

$\beta_{i,j}$ has little effect on s_{max}/r_{max}, as it is contained in both 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 (42)'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}, as shown in Figure 10. □

7.2. RCM versus Lock-Free

CLAIM 14. *For RCM's schedulability to be better or equal to that of [Devi et al. 2006]'s retry-loop lock-free approach, the size of s_{max} must not exceed one half of that of r_{max} for all*

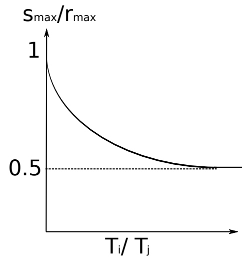


Fig. 10. Effect of $\left\lceil \frac{T_i}{T_j} \right\rceil$ on $\frac{s_{max}}{r_{max}}$

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 (16) is upper bounded by:

$$\sum_{(\tau_j \in \gamma_i) \wedge (p_j > p_i)} \left(\left\lceil \frac{T_i - c_j}{T_j} \right\rceil + 1 \right) .2.\beta_{i,j}.s_{max} \tag{43}$$

Consider the same assumptions as in Section 7.1. 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:

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{T_i} \alpha_{free}/t\,(T_i)}{\sum_{T_i} \alpha_{rma}/t\,(T_i)} \tag{44}$$

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, the RCM is still affected by $2.\beta_{i,j}$ (just as in ECM). The subtraction of c_j in the numerator in (43) may not have a significant effect on the ratio of (44), 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 (44) be given indexes k instead of i, and l instead of j. Let tasks in both the numerator and denominator of (44) be arranged in the non-increasing priority order, so that $i = k$ and $j = l$. Let α_{free}, in (44), 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, (44) becomes:

$$\begin{aligned} \frac{s_{max}}{r_{max}} &\leq \frac{\sum_{\tau_i} (\bar{\alpha}_{free} + \hat{\alpha}_{free})/T_i}{\sum_{\tau_k} \alpha_{rma}/T_k} \\ &= \frac{1}{2} + \frac{\sum_{\tau_i} \hat{\alpha}_{free}/T_i}{\sum_{\tau_k} \alpha_{rma}/T_k} \end{aligned} \tag{45}$$

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 11 illustrates this, where 0 means that the pair i, j does not exist in ζ_1, and the pair k, l does not exist in ζ_2' (i.e., there is no task τ_l that is going to interfere with τ_k in ζ_2), and 1 means the opposite.

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

Fig. 11. Task association for lower priority tasks than T_i and higher priority tasks than T_k

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 $\beta_{i,j} \to \infty$, $\therefore$ (45) $\to \infty$. This is because, $\beta_{i,j}$ represents the number of times a lower priority task τ_j accesses shared objects with the 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, so 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 (45) $\to 1/2$. 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 shown in Figure 12(a). If $\beta_{k,l} \to \infty$, then (45)$\to 1/2$. 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

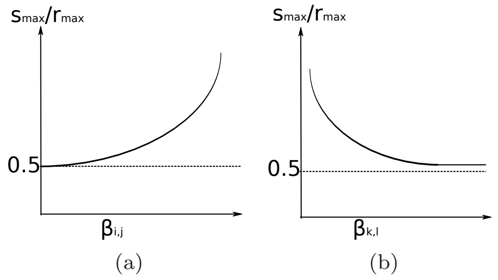


Fig. 12. Change of s_{max}/r_{max}: a) $\frac{s_{max}}{r_{max}}$ versus $\beta_{i,j}$ and b) $\frac{s_{max}}{r_{max}}$ versus $\beta_{k,l}$

retry cost, thus reducing s_{max}/r_{max}. But if $\beta_{k,l} \rightarrow 0$, then (45)$\rightarrow \infty$. 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 12(b).

The third parameter that affects s_{max}/r_{max} is T_k/T_l. If $T_l \ll T_k$, then $\left\lceil \frac{T_k - c_l}{T_l} \right\rceil \rightarrow \infty$, and (45) $\rightarrow 1/2$. This is due to a high number of interferences from a higher priority task τ_l to a lower priority one τ_k, which increases the retry cost under RMA CM, and consequently reduces s_{max}/r_{max}.

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, shown in Figures 12(a) and 12(b). Claim follows. □

7.3. G-EDF/LCM versus lock-free synchronization

CLAIM 15. *If r_{max} is the maximum execution cost of a single iteration of any retry loop of any task in the retry-loop lock-free algorithm in [Devi et al. 2006], then G-EDF/LCM can provide equal or higher $\frac{s_{max}}{r_{max}}$ than what is provided by ECM*

PROOF. From [Devi et al. 2006], the retry-loop lock-free algorithm is upper bounded by:

$$RL(T_i) = \sum_{\tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_{i,h} r_{max} \qquad (46)$$

The final cost of τ_i due to τ_h in G-EDF/LCM is upper bounded by (33). By comparing G-EDF/LCM's total utilization with that of the retry-loop lock-free algorithm, we get:

$$\forall \tau_i \frac{\sum_{\forall \tau_h \in \gamma_i} \left((1 - \alpha_{min}) + \left\lceil \frac{T_i}{T_h} \right\rceil (1 + \alpha_{max}) \right) \beta_{i,h} s_{max}}{T_i} \leq \forall \tau_i \frac{\sum_{\forall \tau_h \in \gamma_i} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_{i,h} r_{max}}{T_i}$$

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

Using (26), we choose $1 - \alpha_{min} = \left\lceil \frac{T_i}{T_h} \right\rceil (1 - \alpha_{max}) - \Delta \left\lceil \frac{T_i}{T_h} \right\rceil$, where $\Delta > 0$. Then:

$$\frac{s_{max}}{r_{max}} \le \frac{\forall\tau_i \frac{\sum_{\forall\tau_h \in\gamma_i}\left(\left\lceil \frac{T_i}{T_h}\right\rceil + 1\right)\beta_{i,h}}{T_i}}{(2-\Delta)\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\left\lceil \frac{T_i}{T_h}\right\rceil\beta_{i,h}}{T_i}}$$

$$= \frac{\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\left\lceil \frac{T_i}{T_h}\right\rceil\beta_{i,h}}{T_i}}{(2-\Delta)\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\left\lceil \frac{T_i}{T_h}\right\rceil\beta_{i,h}}{T_i}} + \frac{\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\beta_{i,h}}{T_i}}{(2-\Delta)\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\left\lceil \frac{T_i}{T_h}\right\rceil\beta_{i,h}}{T_i}}$$

$$= \frac{1}{2-\Delta} + \frac{\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\beta_{i,h}}{T_i}}{(2-\Delta)\forall\tau_i \frac{\sum_{\forall\tau_h\in\gamma_i}\left\lceil \frac{T_i}{T_h}\right\rceil\beta_{i,h}}{T_i}}$$

If $\left\lceil \frac{T_i}{T_h} \right\rceil \to 1$ (as there must be at least one instance of τ_h conflicting with τ_i), then:

$$\frac{s_{max}}{r_{max}} \to \frac{2}{2-\Delta}$$

Thus, $0 \le \Delta < 2$. If $\Delta \to 0$, $\therefore \frac{s_{max}}{r_{max}} \to 1$. But if $\Delta \to 2$, $\therefore \frac{s_{max}}{r_{max}} \to \infty$. This means that, in case of low conflict among tasks and by proper choice of Δ, the length of s_{max} can be equal to or much greater than the length of r_{max}.

If $\left\lceil \frac{T_i}{T_h} \right\rceil \to \infty$, then:

$$\frac{s_{max}}{r_{max}} \to \frac{1}{2-\Delta}$$

Thus, Δ is still in $[0, 2[$, and if $\Delta \to 0$, $\therefore \frac{s_{max}}{r_{max}} \to 0.5$. But if $\Delta \to 2$, $\therefore \frac{s_{max}}{r_{max}} \to \infty$. So, in case of high conflict among tasks, the minimum upper bound for the length of s_{max} is half the length of r_{max}. By proper choice of Δ, the length of s_{max} can be much greater than that of r_{max}.

Thus, we see that, in G-EDF/LCM and by the proper choice of Δ (consequently, by the proper choice of α_{min} and α_{max}), the length of s_{max} ranges from half the length of r_{max} to much higher multiples of r_{max}. On the other hand, in ECM , the length of s_{max} ranges only from half the length of r_{max} to the full length of r_{max}. Claim follows. □

7.4. G-RMA/LCM versus lock-free synchronization

CLAIM 16. *While the minimum upper bound on s_{max}/r_{max} in RCM is 0.5 , G-RMA/LCM achieves a higher minimum upper bound of $1/(1+\alpha_{max})$ on s_{max}/r_{max} by proper choice of α_{max}. This helps to achieve a better or equal schedulability than the retry-loop lock-free algorithm. Similarly to RCM, G-RMA/LCM can increase the length of s_{max} over r_{max} in some cases, which happens when abortion of lower priority tasks effect is reduced and α_{min} is increased.*

PROOF. $RC(T_i)$ for G-RMA/LCM is upper bounded by (37), and $RL(T_i)$ for retry-loop lock-free algorithm is upper bounded by (46), which is re-written here as:

$$RL(T_i) = \sum_{\tau_v \in \gamma_i} \left(\left\lceil \frac{T_i}{T_v} \right\rceil + 1 \right) \beta_{iv} r_{max} \qquad (47)$$

The set of tasks $\tau_v \in \gamma_i$ are composed of tasks with higher priority than τ_i, and are denoted as τ_j, to be consistent with G-RMA/LCM's index notation, and tasks with lower priority than τ_i, which are denoted as τ_h. Thus, (47) can be re-written as:

$$RL(T_i) = \sum_{\forall \tau_j^*} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) \beta_{ij} r_{max} + \sum_{\forall \bar{\tau}_h} \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_{ih} r_{max} \tag{48}$$

Assume that $\lambda_3(i,j) = \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1 \right) \beta_{ij}$ and $\chi_3(i,h) = \left(\left\lceil \frac{T_i}{T_h} \right\rceil + 1 \right) \beta_{ih}$. To determine when the schedulability of G-RMA/LCM will be better or equal to that of retry-loop lock-free, we compare the total utilization of both:

$$\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} \Big(\lambda_3(i,j)(1+\alpha_{max}) len(s_{max}) \Big)}{T_i} + \sum_{\forall \tau_i} \frac{\sum_{\forall \bar{\tau}_h} \Big(\chi_3(i,h)(1-\alpha_{min}) len(s_{max}) \Big)}{T_i}$$
$$\leq r_{max} \sum_{\forall \tau_i} \frac{\sum_{\tau_j^*} \lambda_3(i,j) + \sum_{\bar{\tau}_h} \chi_3(i,h)}{T_i}$$

$$\therefore \frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\tau_j^*} \lambda_3(i,j) + \sum_{\bar{\tau}_h} \chi_3(i,h)}{T_i}}{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} (\lambda_3(i,j)(1+\alpha_{max})) + \sum_{\forall \bar{\tau}_h} (\chi_3(i,h)(1-\alpha_{min}))}{T_i}}$$

Since task relative deadline equals task period, and τ_h is of lower priority than τ_i, $T_i \leq T_h$. Therefore, $\left\lceil \frac{T_i}{T_h} \right\rceil = 1$ for any τ_i and τ_h. Thus:

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} \lambda_3(i,j) + 2\sum_{\forall \bar{\tau}_h} \beta_{ih}}{T_i}}{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} (\lambda_3(i,j)(1+\alpha_{max})) + 2\sum_{\forall \bar{\tau}_h} ((1-\alpha_{min})\beta_{ih})}{T_i}} \tag{49}$$

If the following inequality holds:

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} \lambda_3(i,j) + 2\sum_{\forall \bar{\tau}_h} \beta_{ih}}{T_i}}{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} (\lambda_3(i,j)(1+\alpha_{max})) + 2\sum_{\forall \bar{\tau}_h} ((1+\alpha_{max})\beta_{ih})}{T_i}} \tag{50}$$

where $1 - \alpha_{min}$ in the denominator of the right hand side of (49) is replaced by $1 + \alpha_{max}$, then (49) holds, because the right hand side of (50) is lower or equal to the right hand side of (49). Thus,

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} \lambda_3(i,j) + 2\sum_{\forall \bar{\tau}_h} \beta_{ih}}{T_i}}{(1+\alpha_{max}) \sum_{\forall \tau_i} \frac{\sum_{\forall \tau_j^*} \lambda_3(i,j) + 2\sum_{\forall \bar{\tau}_h} \beta_{ih}}{T_i}} \tag{51}$$

$$\therefore \frac{s_{max}}{r_{max}} \leq \frac{1}{1+\alpha_{max}} \tag{52}$$

Since $0 \leq \alpha_{max} \leq 1$, and by proper choice of α_{max}, the minimum upper bound on $\frac{s_{max}}{r_{max}}$ is higher than 0.5. Thus, the length of s_{max} can be kept higher than half the length of r_{max}, which is higher than that achieved in the specific cases of RCM. (In fact, $\frac{s_{max}}{r_{max}}$ is higher than $\frac{1}{1+\alpha_{max}}$, because the right hand side of (49) is the actual upper limit to $\frac{s_{max}}{r_{max}}$, not the right hand side of (51)). (52) represents the general case for comparison between G-RMA/LCM and retry-loop lock-free. (52) yields the higher upper limit on $\frac{s_{max}}{r_{max}}$ as 1, which means that the maximum allowed length for s_{max}, for G-RMA/LCM's schedulability to be better or equal to that of RCM, is the length of r_{max}. However, considering the special cases in Section 7.2, this can be changed as follows.

From (49), it appears that $\frac{s_{max}}{r_{max}}$ depends on β_{ij}, β_{ih}, $\left\lceil \frac{T_i}{T_j} \right\rceil$, α_{max}, and α_{min}. The minimum value for β_{ij} and β_{ih} is 1, as there must be at least one shared resource between τ_i, τ_j and τ_i, τ_h. $\beta_{ij} \rightarrow 0$, which means that the value of β_{ij} is very small compared to that of β_{ih}, and $\beta_{ih} \rightarrow 0$, which means that the value of β_{ih} is very small compared to that of β_{ij}. Also, if $\beta_{ij} \rightarrow \infty$ ($\beta_{ih} \rightarrow \infty$), then the value of β_{ij} (β_{ih}) is very large compared to that of β_{ih} (β_{ij}). If $\beta_{ij} \rightarrow \infty$ or $\beta_{ih} \rightarrow 0$ (which can happen if higher priority tasks tend to access shared resources more frequently than lower priority tasks), then (49) $\rightarrow \frac{s_{max}}{r_{max}} \leq \frac{1}{1+\alpha_{max}}$, which is the same as the general boundary from (52).

If $\beta_{ij} \rightarrow 0$ or $\beta_{ih} \rightarrow \infty$, which happens if higher priority tasks tend to access shared resources rarely compared with lower priority tasks, then (49) $\rightarrow \frac{s_{max}}{r_{max}} \leq \frac{1}{1-\alpha_{min}}$. If $\alpha_{min} \rightarrow 0$, then the length of s_{max} can be at most as that of r_{max}.

The physical interpretation of these parameters (i.e., $\beta_{ij} \rightarrow 0$ or $\beta_{ih} \rightarrow \infty$ and $\alpha_{min} \rightarrow 0$) is that the effect of blocking in G-RMA/LCM is very large compared to that of interference, and it is even increased by making $\alpha_{min} \rightarrow 0$. This means that an atomic section $s_j^l(\theta)$, belonging to a higher priority task T_j, would suffer the maximum blocking time from any atomic section $s_i^k(\theta)$ belonging to a lower priority task T_i, because $s_j^l(\theta)$ will have to wait for almost the whole length of $s_i^k(\theta)$ in the worst case of blocking.

Since the retry-loop lock-free method also suffers blocking from lower priority tasks whose effect is approximately the same as lower priority tasks in G-RMA/LCM (as $\alpha_{min} \rightarrow 0$), this will cause the lengths of s_{max} and r_{max} to be approximately the same. If $\alpha_{min} \rightarrow 1$, then $\frac{s_{max}}{r_{max}} \leq \infty$, which means that the length of s_{max} can be much greater than that of r_{max}.

The physical interpretation for these parameters (i.e., $\beta_{ij} \rightarrow 0$ or $\beta_{ih} \rightarrow \infty$ and $\alpha_{min} \rightarrow 1$) is that the effect of blocking in G-RMA/LCM is very large compared to the effect of interference. If $\alpha_{min} \rightarrow 1$, then an atomic section $s_j^l(\theta)$ will be blocked by another one $s_i^k(\theta)$ for a very short time length (almost 0). This will significantly reduce the effect of lower priority tasks, in contrast to the retry-loop lock-free algorithm, where lower priority tasks still affect the retry cost. Thus, the length of s_{max} can be much greater than that of r_{max} to achieve higher schedulability than the lock-free method.

If $\left\lceil \frac{T_i}{T_j} \right\rceil \rightarrow 1$, then (49) becomes:

$$\frac{s_{max}}{r_{max}} \leq \frac{\sum_{\forall \tau_i} \frac{\sum_{\tau_j^*} \beta_{ij} + \sum_{\bar{\tau}_h} \beta_{ih}}{T_i}}{\sum_{\forall \tau_i} \frac{(1+\alpha_{max}) \sum_{\tau_j^*} \beta_{ij} + (1-\alpha_{min}) \sum_{\bar{\tau}_h} \beta_{ih}}{T_i}}$$

This means that, the relative schedulability of G-RMA/LCM and the retry-loop lock-free method depends on β_{ij}, β_{ih}, α_{max}, and α_{min}, as explained above. If $\left\lceil \frac{T_i}{T_j} \right\rceil \rightarrow \infty$, then (49)$\rightarrow \frac{s_{max}}{r_{max}} \leq \frac{1}{1+\alpha_{max}}$, which is the same as the general boundary derived in (52).

The claim follows from the analysis for the general and special cases. $\square$

8. CONCLUSIONS

Under both ECM and RCM, a task incurs $2.s_{max}$ retry cost for each of its atomic section 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. 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 abort and retry due to lower priority tasks, while in LCM, this happens.

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 with a task τ_i. 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. Schedulability of G-EDF/LCM and G-RMA/LCM can be better or equal to ECM and RCM, respectively, by proper choices for α_{min} and α_{max}. When LCM was compared with lock-free, the minimum upper bound on the length of s_{max} could be more than half the length of r_{max}, which is higher than that achieved in ECM and RCM, and it can even be much larger than r_{max} in both G-EDF/LCM and G-RMA/LCM. While achieving these high values for s_{max} compared to r_{max} is possible in RCM, it is not possible in ECM, as the maximum value for s_{max}'s length is r_{max}.

Our work has only further scratched the surface of real-time STM. The questions that we ask (see Section 1) are fundamentally analytical in nature, and hence, our results are analytical. However, significant insights can be gained by experimental work on a broad range of embedded software, which is outside our work's scope. 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.

REFERENCES

ANDERSON, J., RAMAMURTHY, S., AND JEFFAY, K. 1995. Real-time computing with lock-free shared objects. In *RTSS*. 28–37.

BARROS, A. AND PINHO, L. 2011. Managing contention of software transactional memory in real-time systems. In *IEEE RTSS, Work-In-Progress*.

BERTOGNA, M. AND CIRINEI, M. 2007. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *RTSS*. 149–160.

BRANDENBURG, B. B. ET AL. 2008. Real-time synchronization on multiprocessors: To block or not to block, to suspend or spin? In *RTAS*. 342–353.

DEVI, U. C., LEONTYEV, H., AND ANDERSON, J. H. 2006. Efficient synchronization under global EDF scheduling on multiprocessors. In *ECRTS*. 75–84.

FAHMY, S., RAVINDRAN, B., AND JENSEN, E. D. 2009. On bounding response times under software transactional memory in distributed multiprocessor real-time systems. In *DATE*. 688–693.

FAHMY, S. F. 2010. Collaborative scheduling and synchronization of distributable real-time threads. Ph.D. thesis, Virginia Tech.

GUERRAOUI, R., HERLIHY, M., AND POCHON, B. 2005. Toward a theory of transactional contention managers. In *PODC*. 258–264.

HARRIS, T., MARLOW, S., ET AL. 2005. Composable memory transactions. In *PPoPP*. 48–60.

HERLIHY, M. 2006. The art of multiprocessor programming. In *PODC*. 1–2.

MANSON, J., BAKER, J., ET AL. 2006. Preemptible atomic regions for real-time Java. In *RTSS*. 10–71.

MCDONALD, A. 2009. Architectures for Transactional Memory. Ph.D. thesis, Stanford University.

SAHA, B., ADL-TABATABAI, A.-R., ET AL. 2006. McRT-STM: a high performance software transactional memory system for a multi-core runtime. In *PPoPP*. 187–197.

SARNI, T., QUEUDET, A., AND VALDURIEZ, P. 2009. Real-time support for software transactional memory. In *RTCSA*. 477–485.

SCHOEBERL, M., BRANDNER, F., AND VITEK, J. 2010. RTTM: Real-time transactional memory. In *ACM SAC*. 326–333.

SHAVIT, N. AND TOUITOU, D. 1995. Software transactional memory. In *PODC*. 204–213.