Worst Case Response Time and Schedulability Analysis for Real-Time Software Transactional Memory-Lazy Conflict Detection (STM-LCD) *

Qiang Zhou†
Sch. of Elec. Info. Engi.
BeiHang University
Beijing, 100191, China
zhouqiang_ee@buaa.edu.cn

Yakun Li
Sch. of Elec. Info. Engi.
BeiHang University
Beijing, 100191, China
18813166248@163.com

Xingliang Zou
Dept. of Comp. Sci.
University of Houston
Houston, TX 77004, USA
xzou@uh.edu

Albert M. K. Cheng
Dept. of Comp. Sci.
University of Houston
Houston, TX 77004, USA
cheng@cs.uh.edu

Yu Jiang†
Sch. of Comp. Sci.&Tech.
Heilongjiang University
Harbin 150001, China
jiangyu@hlju.edu.cn

ABSTRACT

Software transactional memory (STM) is a transactional mechanism of controlling access to shared resources in memory. This transactional mechanism is similar to the abort-and-restart execution model in a functional reactive system (FRS). Due to its abort-and-restart nature, the execution semantics of STM are different from the classic preemptive or nonpreemptive model. Some research has strong constraints for its worst case response time (WCRT) analysis. In this paper, we research on worst case response time and schedulability analysis for real-time software transactional memory-lazy conflict detection (STM-LCD). Specifically, we introduce a parameter the remainder factor m, formally derive an exact WCRT for a 2-task set on STM systems using lazy conflict detection (LCD), propose an exact schedulability test for a 2-task set. Also, we present a near-exact WCRT for an n-task set on STM-LCD, and propose a new necessary condition and a new sufficient condition to schedule an n-task set. Finally, we show that experimental results are accordant with the aforementioned analysis.

Keywords

Software transactional memory, functional reactive systems, WCRT, real-time system, schedulability analysis

1. INTRODUCTION

Transactional memory, as a concurrency control mechanism, can perform better than previously used methods of lock-free/wait-free retry loops [1]. Software Transactional Memory (STM) has been implemented as language extension libraries for C/C++ and Java. Implementations of STM with real-time support are being actively studied. Schoeberl *et al.* [2] have proposed RTTM, an abstract model for implementing transactions with bounded response times in Java-based multiprocessor systems. Sarni *et al.* [3] propose a STM library with real-time support and improve algorithms for conflict detection. There are two ways to detect a conflict, i.e., eager conflict detection (ECD) and lazy conflict detection (LCD) [3, 4]. ECD detects a conflict early on, while LCD detects that at the time of committing. Software mechanisms to implement eager and lazy conflict detection schemes have been presented in [3] and [5]. In [2], Schoeberl *et al.* have pointed out that the implementation of eager conflict detection is more difficult than that of lazy conflict detection, and assume that conflict detection policy does not have an effect on real-time schedulability. Due to lack of studies, the effect of conflict detection policy on real-time schedulability is not very clear and can lead to incorrect implementation of STM in real-time and embedded systems. Since STM offers the advantage of preventing priority inversion without use of locks, it is very useful for developing safety critical and reliable embedded systems.

The transactional mechanism of controlling access to shared resources in software transactional memory is similar to the abort-and-restart (AR) execution model in a priority-based functional reactive programming (P-FRP) system which has been studied with regard to its temporal attributes including response time analysis [6, 7], schedulability analysis [8] and priority assignment [14]. Based on the aforementioned research, Belwal and Cheng [4] have compared the temporal

 This work is sponsored in part by the State Scholarship Fund of China under award Nos. 201303070189, 201308230034, by the Beijing Natural Science Foundation under award No. 4133089, and by the US National Science Foundation under award Nos. 0720856 and 1219082.

†Corresponding authors.

attributes of both ECD and LCD mechanisms. However, the work in [4] does not derive necessary and sufficient scheduling conditions under either of these two conflict detection policies. Though a transactional memory system can be implemented with real-time support, ascertaining the temporal characteristics of its execution model is challenging. This is because a transaction-based execution does not fit into the classic definitions of preemptive and nonpreemptive models, which have been the primary focus of real-time research over the past few decades. However, concurrency control has been an issue in the preemptive execution model, since it could lead to the serious problem of priority inversion [9]. To avoid this, methods like priority ceiling protocol (PCP) [9] or lock-free execution [1] were proposed. Previous works on STM [10] deal with response time analysis for uniprocessor and multiprocessor scheduling. Schedulability conditions for this execution model have not been presented yet.

In [11], some scheduling conditions for a 2-task STM using LCD is proposed. Although it is the first paper which formally derives a WCRT analysis on STM systems, there is a strong constraint in [11]. To overcome the limitation, [12] propose an updated WCRT analysis on STM-LCD. However, it still has a strong constraint in [12]. Both have the limitation of potentially misjudging the schedulability for a 2-task set, because neither is based on the exact WCRT of the tasks. It is hard to obtain a sufficient and necessary condition without the exact presentation of WCRT. Furthermore, for an n-task set on STM-LCD systems, [11] just advance a necessary scheduling condition by Lemma3 in [11] which is optimistic, and [12] present WCRT and advance a sufficient schedulability condition by Theorem 3 in [12] which is pessimistic.

Therefore, this paper solves the aforementioned issue, by introducing some basic concepts, notations, and the STM-LCD execution paradigm (in Sec. 2), by outlining the limitation of the WCRT analysis and the scheduling condition in [12] as a motivation (in Sec. 3), by proposing a parameter the remainder factor m, by deriving the exact WCRT analysis for a 2-task set (in Sec. 4). Also, we present a near-exact WCRT for an n-task set on STM-LCD, and propose a new necessary condition and a new sufficient condition to schedule an n-task set (in Sec. 5). Furthermore, we show that experimental results are accordant with the aforementioned analysis (in Sec. 6). Finally, Section 7 states our conclusions and future research areas.

2. NOTATIONS AND EXECUTION MODEL

2.1 Notations and Basic Concepts

Essential concepts in STM are tasks and their associated priority, their associated time period and the dual concept of arrival rate, and their execution time; the concept of a time interval and release offset therein. In our task model, all these are assumed to be known a *priori*. The notations and formal definitions for these concepts as well as a few others used in the paper are as follows:

• A **task set** $\Gamma_n = \{\tau_1, \tau_2, ..., \tau_n\}$ is a set of n periodic tasks. • the **priority** of $\tau_k \in \Gamma_n$ is the positive integer k, where a lower number implies a higher priority. • T_k is the **arrival time period** between two successive transactions (or jobs, instances) of τ_k and C_k is its **worst-case execution time** (WCET). • D_k is the **relative deadline** of τ_k, where $D_k = T_k$. • x_k is the **offset** (the **release time** of the 1^{st} transaction) of the τ_k, therefore $x_k + (j-1)T_k$ is the release time of the j^{th} transaction of the τ_k. Also, assuming the offset x_n of the 1^{st} transaction of τ_n which is the lowest priority task in Γ_n, is time 0. • tasks are said to be released **synchronously** when the release offsets of tasks are the same. If the release offsets are different tasks are said to be released **asynchronously**. • the **response time** $R_{k,j}$ is the time interval between the release time of the j-th transaction of τ_k and the time instant when it completes processing. R_k (the maximum of $R_{k,j}$) is the **worst case response time** (WCRT) of τ_k. • If the WCRT of all tasks is no more than their deadlines then it can be assured that **the task set will be schedulable in all scenarios**. Otherwise, the task set will be **unschedulable in a worst case** which is simplified as the term "unschedulable" in this paper). • **Critical instant** in STM-LCD is the time at which task releases lead to the WCRT of a lower priority task. Clearly, **a synchronous release of tasks is not guaranteed to lead to the worst case in STM** [13].

2.2 Execution Model

We consider an execution model of an STM system which runs on a uniprocessor system and implements lazy conflict detection (STM-LCD). In such real-time systems, tasks have a *normal* preemptive execution phase as well as an *update* (transactional) phase which has to be restarted upon preemption. Also, not all tasks share the same object, hence, preemption by a higher-priority task is not guaranteed to cause the *update* phase to restart. For this work, we make the assumption that a task is composed only of an *update* phase and that every task in the system shares an object. This is a **worst-case** assumption which allows us to focus on the transactional part of the execution and derive schedulability conditions that will work in every case. There are several *state-of-the-art* methods available for schedulability analysis in a preemptive model hence, accounting for preemptive execution is an unnecessary duplicity of work. The scheduling conditions derived by us for the transactional phase can be integrated with the conditions for the preemptive part of the execution. Thus, the overall temporal guarantees of the system can be derived by combining methods for the two phases.

Since we assume a task is composed only of an *update* phase, for a task τ_k, an *update* operation will take C_k time units to complete execution. If the task is preempted anytime before it has completed C_k times of execution, it will have to restart again. However in STM-LCD a data conflict is detected just before a task is ready to commit its results (and complete execution), therefore the task will still have to complete the remaining time units of execution before it starts a new *update* operation. Also, since all tasks make changes to the same object, if a higher-priority task preempts a lower-priority task the *update* operation of the lower-priority task will have to be restarted. Since, analysis of a transactional execution model is sufficiently complex in itself, overheads associated with preemption and conflict detection policies have been ignored in this paper.

Lazy Conflict Detection. For our STM model, in lazy conflict detection policy the task is assumed to execute for its WCET and then aborted. Its execution semantics is shown in Fig. 1 by using an example in [4].

3. RECENT WORK

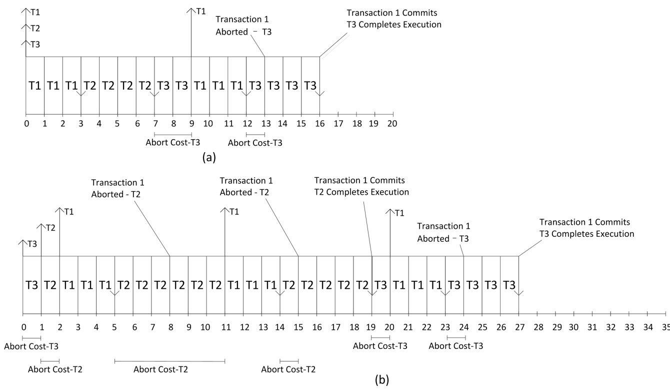

Figure 1: (a) Task execution with synchronous release in lazy conflict detection (STM-LCD) (b) Task execution under worst-case release scenario with STM-LCD. T1, T2 and T3 represent tasks $\tau_1(C_1 = 3, T_1 = 9)$, $\tau_2(C_2 = 4, T_2 = 28)$ and $\tau_3(C_3 = 3, T_3 = 30)$, respectively.

To acertain the hard real-time deadline guarantee, recent work [12] formally derive WCRT for a 2-task set on STM-LCD systems, and advance some scheduling conditions for a 2-task set. However, the sufficient (and necessary) schedulability condition proposed by Theorem 2 in [12] is pessimistic. In this section, we will demonstrate its limitation.

PROPOSITION 1. ([**12**], **theorem2**) *If tasks in a 2-task set $\Gamma_2 = \{\tau_1, \tau_2\}$ under STM-LCD are released asynchronously, a sufficient (and necessary) schedulability condition is: 1) $C_1 \leq T_1$; 2) $\left\lceil \frac{T_2-1}{T_1} \right\rceil (C_1 + C_2) + C_2 \leq T_2$.* ***Or the worst case response time of*** *τ_2 is $\left\lceil \frac{T_2-1}{T_1} \right\rceil (C_1 + C_2) + C_2 \leq T_2$.*

However,Proposition 1 may not hold, when $T_1 - C_1 - C_2 \geq C_2 - 1$. A counter-example can be found in Example 1 and Fig. 2.

Example 1: let $C_1 = 1$, $T_1 = 10$, $C_2 = 4$, and $T_2 = 12$.

Proposition 1 is assumed to proposes a sufficient and necessary condition because Proposition 1 presents the WCRT of τ_2 as $\left\lceil \frac{T_2-1}{T_1} \right\rceil (C_1 + C_2) + C_2$. According to Proposition 1, the worst case response time of τ_2 is $\left\lceil \frac{T_2-1}{T_1} \right\rceil (C_1 + C_2) + C_2 = \left\lceil \frac{12-1}{10} \right\rceil (4 + 1) + 4 = 14 > 12 = T_2$, so the task set Γ_2 is assumed to be unschedulable. However, we can feasibly schedule the task set, if we look into the actual timing analysis in Fig. 2, when the Critical Instant is considered. For the Critical Instant, x_1 can be any integer between $(0, C_2 - 1]$[1], here suppose $x_1 = 1$. So, when $T_1 \geq C_1 + C_2 + C_2 - 1 \geq C_1 + C_2 + C_2 - x_1$, the WCRT of τ_2 is $C_1 + 2C_2 = 9 < T_2 = 10$. Thus, Proposition 1 is incorrect when $T_1 - C_1 - C_2 = 10 - 1 - 4 = 5 \geq 3 = 4 - 1 = C_2 - 1$.

Therefore, there exists the misjudging of schedulability in Proposition 1, since the worst case response time derivation

[1]Here, suppose $C_2 > 1$. Note that the task τ_2 will never be preempted because of the atomicity when $C_2 = 1$, i.e., either τ_2 can start its execution at $t = 0$ and finish its execution at $t = 1$ in an asynchronous case, or τ_2 cannot be executed at $t = 0$ in a synchronous case, therefore there is no preemption for τ_2 at all, when $C_2 = 1$.

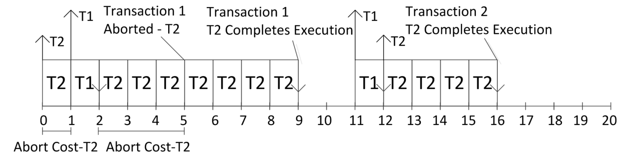

Figure 2: the WCRT timing analysis for τ_2 (when $T_1 - C_1 - C_2 \geq C_2 - 1$, i.e., $m \geq C_2 - 1$)

of τ_2 in Proposition 1 is not actually exact. Proposition 1 pessimistically calculates the WCRT of τ_2. In example 1, by Proposition 1, WCRT is assumed to be 14, while its exact value is 9.

Based on the actual timing analysis in Fig. 2, τ_2 can be scheduled at some time instant ($x_1 + C_1 + (C_2 - x_1) + C_2 = 9$) before the release of the ($\left\lceil \frac{T_2-1}{T_1} \right\rceil =)2^{nd}$ transaction of τ_1, where x_1 can be any integer between $(0, C_2 - 1 = 3]$. This means that the ($\left\lceil \frac{T_2-1}{T_1} \right\rceil =)2^{nd}$ transaction of τ_1 will never preempt the current transaction of τ_2, so no need to calculate its preempting and abort cost. While, Proposition 1 always pessimistically calculate the WCRT ($= 14$) of τ_2 until the time instant when the ($\left\lceil \frac{T_2-1}{T_1} \right\rceil =)2^{nd}$ transaction of τ_1 is released, which is not correct.

4. WCRT AND SCHEDULABILITY CONDITION FOR A 2-TASK SET

In this section, we propose an exact WCRT analysis for a 2-task set.

4.1 A Necessary Schedulability Condition for a 2-task Set

We formally state the schedulability characteristics of the STM-LCD execution model, which are required for the WCRT analysis and the necessary and sufficient schedulability tests derived in subsequent sections.

LEMMA 1. ([**4**, **11**]) *If a task set is schedulable in STM-LCD, a necessary condition of schedulability is that tasks will be able to complete execution (including the abort cost) between successive transactions of any other task present in the set. Or, for $\Gamma_n = \{\tau_1, \tau_2, ..., \tau_n\}; \forall \tau_i, \tau_j \in \Gamma_n : i \neq j, C_i + C_j \leq T_i$.*

However, for $\Gamma_2 = \{\tau_1, \tau_2\}$ in STM-LCD, Lemma 1 does not hold, when $0 = T_1 - C_1 - C_2$ and $C_2 > 1$.

Figure 3 shows this case as follows. Let the 1^{st} transaction of τ_1 be released at time x_1. For the worst case, x_1 can be any integer between $(0, C_2 - 1]$, here suppose $x_1 = 1$. So the 1^{st} transaction of τ_1 will preempt the 1^{st} transaction of τ_2, and start processing at time x_1. The 1^{st} transaction of τ_2 will resume execution after the 1^{st} transaction of τ_1, and will be aborted at time $x_1 + C_1 + (C_2 - x_1)$. The 1^{st} transaction of τ_1 will then restart at time $x_1 + C_1 + (C_2 - x_1)$ and cannot finish its execution because that the 2^{nd} transaction of τ_1 will preempt it again at time $x_1 + C_1 + C_2 = x_1 + T_1$ since $0 = T_1 - C_1 - C_2$. And the 1^{st} transaction of τ_2 will always be preempted by the following transactions of τ_1 and can never finish its execution. Thus, by considering the case of $0 = T_1 - C_1 - C_2$, the necessary condition in Lemma 2 can be strengthened, and we can derive the following lemma:

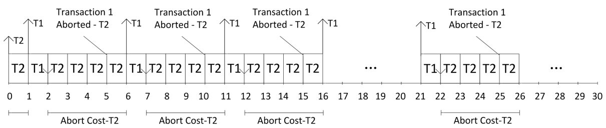

Figure 3: the WCRT timing analysis for τ_2 (when $T_1 - C_1 - C_2 = 0$, i.e., $m = 0$)

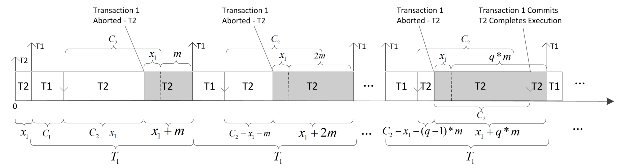

Figure 4: The WCRT timing analysis for τ_2

LEMMA 2. *If a task set $\Gamma_2 = \{\tau_1, \tau_2\}$ is schedulable in STM-LCD when the worst case is considered, a necessary condition of schedulability is that task τ_2 will be able to complete execution (including the abort cost) between successive transactions of τ_1, i.e. $1 \leq T_1 - C_1 - C_2$, where $C_2 > 1$.*

PROOF. By omitting the case of $0 = T_1 - C_1 - C_2$ in which the task set Γ_2 cannot be scheduled (this case can be shown in Fig. 3), we can directly derive this Lemma. □

4.2 WCRT and Schedualbility Test for a 2-task Set

In this subsection, we discuss the exact worst case response time for a 2-task set Γ_2. We first introduce a definition of the remainder factor m, then propose a lemma for m, after that derive the exact WCRT based on the definition of m.

Definition 1: For simplicity and without loss of generality, we introduce a parameter *the remainder factor*, denoted as m, which presents the reduction in the remainder of C_2 to be executed upon repeated pre-emptions/abortions by τ_1 in the interval of T_1, i.e., $m = T_1 - C_1 - C_2$.
For instance, in example 1, ($C_1 = 1$, $T_1 = 10$, and $C_2 = 4$), $m = T_1 - C_1 - C_2 = 5$.
By the remainder factor m and Lemma 2, we can get the following lemma.

LEMMA 3. *Task τ_2 in a 2-task set $\Gamma_2 = \{\tau_1, \tau_2\}$ under STM-LCD is unschedulable when the worst case is considered and the WCRT of τ_2 is infinite, when $C_2 > 1$ and $m \leq 0$.*

PROOF. This Lemma can be directly derived from Lemma 2. □

THEOREM 1. *The WCRT of τ_2 in a 2-task set $\Gamma_2 = \{\tau_1, \tau_2\}$ under STM-LCD is*

$$R_2 = \begin{cases} C_1 + C_2, & C_2 = 1, \\ \infty, & C_2 > 1; m \leq 0, \\ \lceil \frac{C_2 - 1}{m} \rceil * (C_1 + C_2) + C_2, & C_2 > 1; m > 0, \end{cases} \tag{1}$$

PROOF. 1) When $C_2 = 1$, the task τ_2 will never be preempted by τ_1 because of the atomicity, so the Critical Instant is the time instant when tasks τ_2 and τ_1 have the synchronous release, this means $R_2 = C_1 + C_2$. In this case, the abort-and-restart scheme will be not applicable.

2) When $C_2 > 1$ and $m \leq 0$, R_2 is infinite by Lemma 3.

3) When $C_2 > 1$ and $m > 0$,
To obtain the WCRT of τ_2, let us analyze the execution timing procedure of τ_2, when induced by τ_1. This can be described in Fig. 4. Suppose that the offset of the lowest priority task τ_2 is 0, that x_1 ($0 < x_1 < C_2$) is the offset of τ_1, therefore, the release time of the 1^{st}, 2^{nd}, $\cdots$, i^{th} transaction of τ_1, are x_1, $x_1 + T_1$, $\cdots$, $x_1 + (i-1) * T_1$, respectively.

At the beginning, the 1^{st} transaction of τ_2 releases at the time instant 0, and can be processed until the time instant x_1, at which the 1^{st} transaction of τ_1 releases and preempts **(for the 1^{st} time)** the 1^{st} transaction of τ_2, so in the time interval $[x_1, x_1 + T_1]$, the 1^{st} transaction of τ_1 can complete its processing and take up the time interval $[x_1, x_1 + C_1]$, then the remaining part of the 1^{st} transaction of τ_2 can be processed until the time instant $x_1 + C_1 + (C_2 - x_1)$, and can be (**for the 1^{st} time**) aborted and restarted by the LCD policy. After that there remains $x_1 + m$ time-slice (the shaded part in Fig. 4) taking up the interval $[x_1 + C_1 + (C_2 - x_1), x_1 + T_1]$, here m **is the remainder factor,** $m = T_1 - C_1 - C_2$, $m > 0$.

If the remaining $x_1 + m$ time-slice is no less than C_2, then the 1^{st} transaction of τ_2 can complete its processing, accordingly the response time of the 1^{st} transaction of τ_2 is $x_1 + C_1 + (C_2 - x_1) + C_2 = (C_1 + C_2) + C_2$. Otherwise, the 2^{nd} transaction of τ_1 will preempt **(for the 2^{nd} time)** the 1^{st} transaction of τ_2, and complete execution which takes up the time interval $[x_1 + T_1, x_1 + T_1 + C_1]$, then the remaining part of the 1^{st} transaction of τ_2 can be processed until the time instant $x_1 + T_1 + C_1 + (C_2 - x_1 - m)$, and can be (**for the 2^{nd} time**) aborted and restarted. After that there remains $x_1 + 2 * m$ time-slice (the shaded part in Fig. 4) taking up the interval $[x_1 + T_1 + C_1 + (C_2 - x_1 - m), x_1 + 2 * T_1]$, and so on.

Therefore, we can derive that there remains $x_1 + q * m$ time-slice (the shaded part in Fig. 4) taking up the interval $[x_1 + (q-1) * T_1 + C_1 + (C_2 - x_1 - (q-1) * m), x_1 + q * T_1]$. To search for the response time of the 1^{st} transaction of τ_2, is to find the minimal integer q_m satisfying: $x_1 + q_m * m \geq C_2$. This means $q_m = \lceil \frac{C_2 - x_1}{m} \rceil$. Note that q_m is just the abort number of the transaction of τ_2 induced by τ_1 before completing its execution.

Accordingly, the response time of the transaction of τ_2 is $x_1 + (q_m - 1) * T_1 + C_1 + (C_2 - x_1 - (q_m - 1) * m) + C_2 = (q_m - 1) * (T_1 - m) + C_1 + C_2 + C_2 = q_m * (C_1 + C_2) + C_2$.

To search for the worst case response time of τ_2 is to find the maximal q_m, denoted as q_{wc}, when considering any offset of τ_2 except the synchronous case. Because $q_m = \lceil \frac{C_2 - x_1}{m} \rceil$ is a non-increasing function of x_1, we can derive $q_{wc} = \lceil \frac{C_2 - 1}{m} \rceil$, when $x_1 = 1$, ($0 < x_1 < C_2$).

Thus, the worst case response time R_2 of τ_2 can be obtained and is $q_{wc} * (C_1 + C_2) + C_2 = \lceil \frac{C_2 - 1}{m} \rceil * (C_1 + C_2) + C_2$.

So Theorem 1 holds. □

Now, Theorem 1 gives an exact expression of the WCRT of τ_2 in a 2-task set $\Gamma_2 = \{\tau_1, \tau_2\}$ under STM-LCD.

Looking back into Example 1 ($C_1 = 1$, $T_1 = 10$, $C_2 = 4$, and $T_2 = 12$.) in previous section, where Proposition 1 does not hold, Theorem 1 can give a correct judgment as follows. Based on (1), since $m = T_1 - C_1 - C_2 = 10 - 1 - 4 = 5 > 0$

and $R_2 = \lceil \frac{C_2-1}{m} \rceil * (C_1+C_2) + C_2 = \lceil \frac{4-1}{5} \rceil * (4+1) + 4 = 9 < 12 = T_2$, the task τ_2 is schedulable. This follows the actual timing analysis in Fig. 2, when the worst case is concerned.

Thus, Theorem 1 corrects the schedulability misjudging from Proposition 1. Unlike Proposition 1, no limited condition between T_1 and T_2 is needed in Theorem 1. Therefore, Theorem 1 is more general.

After deriving the exact WCRT of the task, we will discuss the exact schedulability test for a 2-task set.

Theorem 2. *Tasks in a 2-task set $\Gamma_2 = \{\tau_1, \tau_2\}$ under STM-LCD is schedulable iff the following conditions are satisfied,*

1) $R_1 \leq T_1$, and

2) if $C_2 = 1$, then $C_1 + C_2 \leq T_2$ or

if $C_2 > 1$, then $m > 0$, and $\lceil \frac{C_2-1}{m} \rceil * (C_1+C_2) + C_2 \leq T_2$,

where $m = T_1 - C_1 - C_2$.

Proof. This Theorem can be proved by contradiction.(We omit the detailed proof because of the limited space.) □

Theorem 2 presents the sufficient and necessary condition to schedule a 2-task set. Based on Theorem 2, the exact schedulability test for a 2-task set can be efficiently implemented. In example 1, $m = T_1 - C_1 - C_2 = 10 - 1 - 4 = 5 > 0$, $R_2 = \lceil \frac{C_2-1}{m} \rceil * (C_1 + C_2) + C_2 = \lceil \frac{4-1}{5} \rceil * (4+1) + 4 = 9 < 12 = T_2$. Thus, $\Gamma_2 = \{\tau_1, \tau_2\}$ is schedulable.

5. WCRT AND SCHEDULABILITY CONDITIONS FOR AN N-TASK SET

For a n-task set $\Gamma_n = \{\tau_1, \tau_2 ..., \tau_n\}$, there is at least $(n-1)!$ abort combinations for n periodic tasks, all of which must be checked for the worst case to be found. Therefore, finding the *critical instant* for the abort-and-restart model with periodic and sporadic tasks is intractable [14]. Therefore, we present a near-exact WCRT expression as in Sec. 5.1 which is less pessimistic than Theorem 3 in [12]. After that, we derive one new necessary condition and one new sufficient condition for STM-LCD, respectively.

5.1 WCRT and a Sufficient Schedulability Condition for an N-task Set

Wen *et al.* [12] proposed a WCRT expression (Theorem 3 in [12]) for n-task sets, however it is pessimistic. Therefore, we improve it by proposing the lemma as follows.

Lemma 4. *In an n-task set $\Gamma_n = \{\tau_1, \tau_2 ..., \tau_n\}$, $n \geq 2$, the WCRT of τ_i is:*

$$R_i = C_i + \left\lceil \frac{R_i - x_{i-1}}{T_{i-1}} \right\rceil (C_i + C_{i-1}) + \left\lceil \frac{R_i - x_{i-2}}{T_{i-2}} \right\rceil (max\{C_i, C_{i-1}\} + C_{i-2}) + \cdots + \left\lceil \frac{R_i - x_1}{T_1} \right\rceil (max\{C_i, C_{i-1}, \cdots, C_2\} + C_1) \quad (3)$$

where $i = 1, 2, \cdots, n$; x_i is the offset for τ_i.

Proof. In $[0, R_i]$, there are $\left\lceil \frac{R_i - x_h}{T_h} \right\rceil$ transactions of τ_h, $h < i$. Since each transaction of a higher priority task can induce abort costs to only a single transaction of one lower priority task, to achieve the WCRT, each transaction of τ_h should preempt a task transaction with largest abort cost, or largest execution time, which is $max\{C_i, C_{i-1}, , C_{h+1}\}$. Then to finish all transactions of τ_h, the worst case execution time is $\left\lceil \frac{R_i - x_h}{T_h} \right\rceil (max\{C_i, C_{i-1}, \cdots, C_{h+1}\} + C_h)$, $h < i$. While, τ_i only needs C_i to execute itself, but it has to wait until tasks of higher priorities to finish. Summing them up proves the lemma. □

Then, we derive an improved sufficient schedulability condition.

Theorem 3. *In a n-task set $\Gamma_n = \{\tau_1, \tau_2 ..., \tau_n\}$, $n \geq 2$, suppose that R_i^* is a bounded resolvant of (3). The sufficient condition to feasibly schedule task τ_i is, $R_i^* \leq T_i$.*

Proof. It can be proved directly by Lemma 4. □

5.2 A Necessary Schedulability Condition

Belwal and Cheng [11] proposed a necessary schedulability condition (Theorem 2 in [11]) for n-task sets, here we strengthen it by propose the theorem as follows.

Theorem 4. *If a n-task set $\Gamma_n = \{\tau_1, \tau_2, \cdots, \tau_n\}$, $n \geq 2$, can be feasibly scheduled, the following condition should be satisfied: $2\sum_{j=1}^{n} C_j \leq \sum_{j=1}^{n} T_j - \frac{n}{2}$, where $C_k > 1$, $k = 2, 3, \cdots, n$.*

Proof. Main idea: from Lemmas 1 and 2, the theorem can be derived. (We omit the detailed proof because of the limited space.) □

6. EXPERIMENTAL ANALYSIS

6.1 Experimental Analysis for 2-task Sets

To check the sufficient and necessary condition for a 2-task set in STM-LCD, we generated 1000 2-task sets in two groups. The first group had utilization less than or equal to 0.5, and the second group had the utilization factor was in the range [0.1, 1]. Each task set in the same group was unique in the sense that at least one task was different between any two task sets. The arrival periods for all task sets were randomly selected from the range [10, 70], while their execution times were generated from the *UUniFast* algorithm [15].

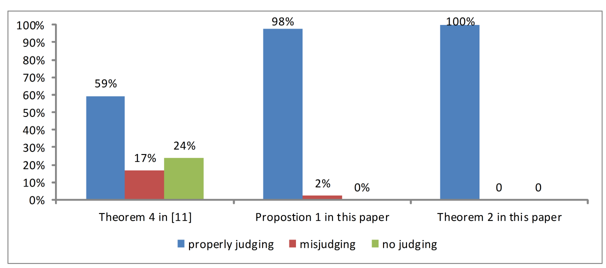

Figure 5: Comparisons of the schedulability test among Theorems

To sum up the results for the two groups of 1000 2-task sets, in term of the schedulability test for Theorem 4 in [11], Proposition 1 and Theorem 2 in this paper, all task sets can be divided into three categories, *properly judging*, *misjudging*, and *no judging*. Figure 5 showed comparisons of the schedulability test among Theorem 4 in [11], Proposition 1 and Theorem 2 in this paper. In Fig. 5 , Theorem 4 in [11], covering all three categories, has the lowest performance of schedulability test. While, Proposition 1, without *no judging* category, has better schedulability than Theorem 4 in [11]. Finally, Theorem 2, having 100% properly judging ratio, has the best schedulability of all these three theorems.

6.2 Experimental Analysis for N-task Sets

To check the sufficient condition of an n-task set in STM-LCD, we generated another 6 groups of 5000 n-task sets. All groups had the utilization factor in the range [0.1, 0.6]. The arrival periods for all task sets were randomly selected from the range [10, 70], while their execution times were generated from the *UUniFast* algorithm [15].

We experimentally tested the schedulability of each task set in 6 groups, and obtained the result. Figure 6 showed that the numbers in the blue bars were less than those in the red bars for all 6 groups, where the blue bars and the red bars represent the numbers of task sets which can satisfy the sufficient condition of Theorem 3 in [12] and the numbers of task sets which can satisfy the sufficient condition of **Theorem** 3 in this paper, respectively. **Theorem** 3 strengthened the sufficient condition of the former and had better schedulability test results: (1) task sets which can satisfy **Theorem** 3 definitely can satisfy the former; (2) While, there existed several feasibly scheduled task sets which can satisfy **Theorem** 3 but cannot satisfy the former, in Fig. 6, the numbers of this kind of task sets are $179(= 689-510)$, $86(= 146-60)$, $218(= 271 - 53)$, $470(= 572 - 102)$, $413(= 454 - 41)$ and $13(= 15 - 2)$ for $n = 2, 3, 4, 5, 6$ and 7, respectively.

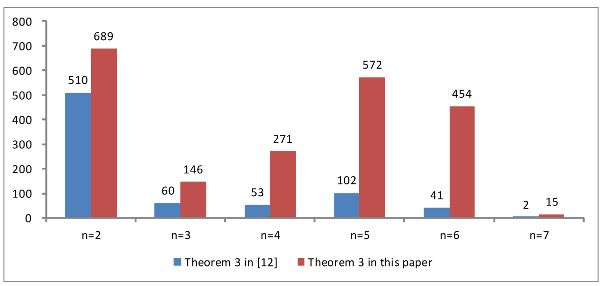

Figure 6: The schedulability test of Theorem 3

Also, we conducted experiments to test the necessary condition between Theorem 2 in [11] and Theorem 4 in this paper. And the result showed that Theorem 4 in this paper had better schedulability than Theorem 2 in [11]. (We omit the detailed content because of the limited space.)

7. CONCLUSIONS

In this paper, we introduced a definition of the remainder factor m, derived an exact WCRT analysis and schedulability test for a 2-task set. Also, we presented a near-exact WCRT for an n-task set on STM-LCD, and proposed an improved necessary condition and an improved sufficient condition to schedule an n-task set. Finally, we showed that experimental results are accordant with the aforementioned analysis.

Our future research will focus on searching a nearer-exact WCRT for an n-task set and more strengthening schedulability conditions for an n-task set in STM-LCM.

8. REFERENCES

[1] J. H. Anderson, S. Ramamurthy, and K. Jeffay, "Real-time computing with lock-free shared objects," *ACM Trans. Comput. Syst. 5(6), 388-395,* 1997

[2] M. Schoeberl, F. Brandner, and J. Vitek, "RTTM: Real-time transactional memory," *in Proc. ACM-SAC'10, Sierre, Switzerland,* 2010

[3] T. Sarni, A. Queudet, and P. Valduriez, "Real-time support for software transactional memory," *in Proc. RTCSA, Beijing, China, pp. 477-485,* 2009

[4] C. Belwal and A. M. K. Cheng, "Lazy vs. Eager conflict detection in software transactional memory: A real-time schedulability perspective," *IEEE Embed. Syst. Lett., vol. 3, no. 1, Mar.* 2011

[5] W. N. Scherer and M. L. Scott, "Advanced contention management for dynamic software transactional memory," *in Proc. PODC, Las Vegas, NV,* 2005

[6] J. Ras and A. M. K. Cheng, "Response Time Analysis for the Abort-and-Restart Event Handlers of the Priority-Based Functional Reactive Programming (P-FRP) Paradigm," *RTCSA, 2009. 15th IEEE International Conference on.* 2009

[7] Q. Zhou, X. Zou, A. M. K. Cheng, and Y. Jiang, "An Integrated Analysis of the Worst Case Response Time for P-FRP," *35th IEEE-CS RTSS WIP Session, Rome, Italy,* December 2014

[8] C. Belwal, and A. M. K. Cheng, "Schedulability Analysis of Transactions in Software Transactional Memory Using Timed Automata," *in TrustCom, 2011 IEEE 10th International Conference on.* 2011

[9] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real time synchronization," *Trans. Comput., vol. 39, no. 9, pp. 1175-1185,* 1990

[10] S. F. Fahmy, B. Ravindran, and E. D. Jensen, "Response time analysis of software transactional memory-based distributed real-time systems," *in Proc. ACM SAC Operat. Syst., Honolulu, HI,* 2009

[11] C. Belwal and A. Cheng, "Scheduling conditions for real-time software transactional memory," *IEEE Embed. Syst. Lett., vol. 3, no. 3, pp. 93-96, sept.*2011

[12] Y. Wen, A. M. K. Cheng, and C. Belwal, "Worst Case Response Time for Real-Time Software Transactional Memory", *in Research in Applied Computation Symposium, 2012. Proceedings of the 2012 ACM, pp. 459-460, Oct.* 2012

[13] C. Belwal, and A.M.K. Cheng, "A Sufficient Schedulability Test for Real-Time Software Transactional Memory," *in TrustCom, 2011 IEEE 10th International Conference on.* 2011

[14] H. C. Wong, and A. Burns, "Schedulability Analysis for the Abort-and-Restart (AR) Model," *in Proc. RTNS. 2014, ACM,pp. 119-128,* 2014

[15] E. Bini and G.C. Buttazzo, "Measuring the Performance of Schedulability Tests," *Real-Time Syst., 30(1-2): p. 129-154,* 2005