Work-In-Progress: Toward Precomputation in Real-Time Mixed-Trust Scheduling

Dionisio de Niz[1], Bjorn Andersson[1], Hyoseung Kim[3], Mark Klein[1], and John Lehoczky[2]
dionisio@sei.cmu.edu, baandersson@sei.cmu.edu, hyoseung@ucr.edu, mk@sei.cmu.edu,jl16@andrew.cmu.edu
{[1]SEI, [2]Stat} – Carnegie Mellon University, [3]ECE – UC Riverside

I. Introduction

The Real-Time Mixed-Trust (RTMT) Framework [2] enables the use of untrusted components in safety-critical CPS functions (e.g., driving a car) by monitoring their actions with verified and trusted components (called *enforcers*) that correct unsafe actions to guarantee critical safety properties (e.g., brake to prevent a crash). The enforcers are run within a verified hypervisor that protects them from security attacks or bugs and the untrusted components are run in an unverified virtual machine (VM) on top of the hypervisor. The untrusted and trusted components are executed as a single coordinated sporadic real-time task, called a *mixed-trust task*, where the untrusted part is known as the guest task (GT, because it runs in the guest VM) and the trusted part running in the hypervisor (HV) is known as the hypertask (HT). The GT is run by a preemptive fixed-priority scheduler in the VM and the HT by a non-preemptive fixed-priority scheduler in the HV. The non-preemptive scheduler prevents interleavings and simplifies the logical verification [4], [5]. From a timing point of view, the HT monitors that the GT produces a valid output before the deadline, and if not, the HT itself produces a safe output before the deadline elapses. A new set of schedulability equations to evaluate their schedulability were presented in [2] along with a full discussion of the framework.

This paper builds on the RTMT framework and takes a step toward the ultimate goal of the development of full verification support of autonomous systems. Specifically, the paper develops support for predictive task sets. Our discussions with autonomous systems researchers have indicated a need for flexible runtime assurance mechanisms that take into account predictive computation. This paper offers a flexible mechanism from a timing point of view and formalizes the associated timing analysis.

The original RTMT framework had two drawbacks: (1) the schedulability equations assumed that the HT always runs enforcement although we expect in reality that the HT only runs in emergencies, and (2) it was assumed that the reaction to emergencies must be handled within only one invocation of the HT. To address these drawbacks, in this paper, we propose an enhancement that allows HTs to run alternative computations when they do not need to perform enforcement computations. The results of the alternative computations can be used for future enforcement when it needs to obtain a refined assessment of the situation (e.g., to verify that no obstacles are nearby and less than full braking is needed) or build a more refined reaction to create an escape trajectory to deal with a GT deadline miss. Hence, we identify such additional work as *predictive computation (or pre-computation)* and the HT jobs executing pre-computation as *predictive jobs*, which are executed only when the enforcer jobs of the HT do not execute. An example of a trajectory predictor is depicted in Figure 1. This figure captures the notional concept of some pre-computation that can be stored and used when the enforcement needs to take place.

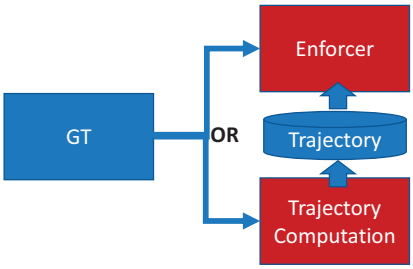

Fig. 1. Predictive trajectory computation and enforcement

In our model, the pre-computation jobs need to produce some amount of data to be used by the enforcer job. We encode this as I_i, which defines the minimum number of pre-computing jobs required in order for the enforcer to produce a better quality output. It is worth noting that safety does not depend on the value of I_i. However, if I_i pre-computation jobs are not completed, then the improved enforcement cannot be executed and the enforcement is executed as if no pre-computation jobs occurred (all or nothing semantics). In addition, we allow the enforcer and the pre-computing jobs to have different WCETs. In the following, we will discuss how we extend the original RTMT framework to use this parameter.

II. Mixed-Trust Scheduling Background

Let us start by presenting the original RTMT system scheduling followed by the extensions for predictive enforcement.

In the RTMT framework, a system is composed of a set of mixed-trust tasks $\Gamma = \{\mu_i | \mu_i = (T_i, D_i, \tau_i, \kappa_i)\}$ running on a single-core processor. Each mixed-trust task μ_i is modeled as two execution segments, τ_i and κ_i, with period T_i and

2576-3172/20/$31.00 ©2020 IEEE
DOI 10.1109/RTSS49844.2020.00051

deadline D_i. The segment τ_i is the untrusted component and runs in the untrusted OS kernel inside the VM. The segment κ_i is the trusted component and runs within the trusted HV in a higher priority band than all τ's. To represent the fact that these segments are handled by different schedulers, [2] considers them to be tasks and calls τ_i the *guest task* (*GT*) and κ_i the *hypertask* (*HT*). These tasks are defined by: $\tau_i = (T_i, E_i, C_i)$, $\kappa_i = (T_i, D_i, \kappa C_i)$, where T_i and D_i are the same as in μ_i, C_i is the WCET of τ_i, κC_i is the WCET of κ_i, and E_i is the deadline for τ_i.

We begin by showing the schedulability equations in [2]; this is useful because we will, later in this paper, make straightforward changes to the equations in [2] so that pre-computation jobs are included in schedulability analysis. Later on, we will also present a much better approach that is not based on straightforward changes.

The reasoning underlying the schedulability analysis in [2] is as follows. The response time of an HT κ_i (following [1]) is obtained by first calculating the maximum duration of a level-i active period (denoted t_i^κ) as the smallest solution of:

$$t_i^\kappa = \max_{j \in \kappa L_i} \kappa C_j + \left\lceil \frac{t_i^\kappa}{T_i} \right\rceil \kappa C_i + \sum_{j \in \kappa H_i} \left\lceil \frac{t_i^\kappa}{T_j} \right\rceil \kappa C_j, \quad (1)$$

where κL_i is the set of all HTs with lower priority than κ_i and κH_i is the set of tasks with higher priority than κ_i.

Then the latest start time of the q^{th} job of κ_i in the level-i active period ($w_{i,q}^\kappa$) is calculated as the smallest solution of:

$$w_{i,q}^\kappa = \max_{j \in \kappa L_i} \kappa C_j + (q-1)\kappa C_i + \sum_{j \in \kappa H_i} \left(\left\lfloor \frac{w_{i,q}^\kappa}{T_j} \right\rfloor + 1 \right) \kappa C_j, \quad (2)$$

and the response time of the HT κ_i by

$$R_i^\kappa = \max_{q \in \{1 \ldots \left\lceil \frac{t_i^\kappa}{T_i} \right\rceil\}} (w_{i,q}^\kappa + \kappa C_i - (q-1)T_i), \quad (3)$$

which basically adds the computation time to the latest start among all jobs in the active period.

Given this response time of an HT, the E_i timer which serves as the deadline of the corresponding GT is obtained as follows:

$$E_i = D_i - R_i^\kappa,$$

To calculate the response time of a GT τ_i, it is necessary to evaluate all the potential phasings of the interfering GTs (with higher priority) and HTs (all HTs except its own). To simplify this, [2] defines the request bound function (rbf) that captures the computation time of the mixed-trust task μ_i and all of the interfering tasks as in the equation below:

$$\text{rbf}_i^y(t, b) = \begin{cases} \left\lceil \frac{t-(T_i-E_i)}{T_i} \right\rceil^+ C_i b + \left\lceil \frac{t}{T_i} \right\rceil \kappa C_i & \text{if } y = E, \\ \left\lceil \frac{t}{T_i} \right\rceil C_i b + \left\lceil \frac{t-E_i}{T_i} \right\rceil^+ \kappa C_i & \text{if } y = A, \end{cases} \quad (4)$$

where $\lceil x \rceil^+ = \max(0, \lceil x \rceil)$, $y \in \{E, A\}$ indicates if the interfering task is aligned with the HT (E) or the GT (A), and $b \in \{0, 1\}$ indicates if the GT execution should be included in the rbf.

The rbf is then used to calculate the maximum level-i busy period that starts with an HT job ($x = E$) or the GT job ($x = A$) of the mixed-trust task μ_i:

$$t_i^{g,x} = \left(\sum_{j \in L_i} \text{rbf}_j^E(t_i^{g,x}, 0) \right) + \text{rbf}_i^x(t_i^{g,x}, 1) + \sum_{j \in H_i} \max_{y \in \{E,A\}} \text{rbf}_j^y(t_i^{g,x}, 1), \quad (5)$$

where L_i and H_i contain the tasks with lower and higher priority than τ_i, respectively.

Let $w_{i,q}^{g,x}$ denote the latest finishing time of the q^{th} job of the GT τ_i, relative to the start of the maximum level-i busy period, such that this level-i busy period starts with a job of the HT or the GT of μ_i. Then, the analysis in [2] computes $w_{i,q}^{g,x}$ as the smallest solution of:

$$w_{i,q}^{g,x} = \left(\sum_{j \in L_i} \text{rbf}_j^E(w_{i,q}^{g,x}, 0) \right) + qC_i + (q - 1 + \mathcal{B}_{(x=E)})\kappa C_i + \sum_{j \in H_i} \max_{y \in \{E,A\}} \text{rbf}_j^y(w_{i,q}^{g,x}, 1). \quad (6)$$

where $\mathcal{B}_\phi$ is an indicator function that returns 1 if ϕ is true and 0 otherwise.

The response time of a job for different phasings is computed by:

$$R_{i,q}^{g,x} = w_{i,q}^{g,x} - ((q-1)T_i + \mathcal{B}_{(x=E)}(T_i - E_i)). \quad (7)$$

Then the maximum response time among all the jobs in the busy period is calculated with

$$R_i^{g,x} = \max_{q \in \left\{1 \ldots \left\lceil \frac{t_i^{g,x} - I_{x=E}(T_i - E_i)}{T_i} \right\rceil\right\}} R_{i,q}^{g,x}. \quad (8)$$

Finally, the response time of a GT τ_i is given by:

$$R_i^g = \max_{x \in \{E,A\}} R_i^{g,x}. \quad (9)$$

With these equations, the original RTMT framework first calculates the response time of all the HTs, then their respective E_i intervals, and finally the response time of the GTs. A taskset is schedulable if after all the calculated response times of the HT are less than or equal to their deadline and all the response times of the GT are less than or equal to their respective E_i.

III. Predictive Mixed-Trust Scheduling

In our predictive mixed-trust model, we need to evaluate the schedulability of both the worst-case enforcement situation (that is basically the same as the original mixed-trust task) and the situation when the HTs do not need to enforce but perform some pre-computation.

A. System Model

Our new model for predictive scheduling can be captured as a modification to the real-time mixed-trust model as follows. First the I_i parameter is added to the mixed-trust task to capture the number of predictive jobs: $\mu_i = (T_i, D_i, I_i, \tau_i, \kappa_i)$. Secondly, we extend the definition of the HT to include the WCETs of the enforcement (κC_i^e) and the predictive (κC_i^p) jobs, redefining the HT as: $\kappa_i = (T_i, D_i, \kappa C_i^e, \kappa C_i^p)$. Finally, we define $\kappa C_i^m = \max(\kappa C_i^e, \kappa C_i^p)$. It is worth noting that our model is based on concepts of the multiframe model [3] accommodating the mixed-preemption, priority bands, and intermediate deadlines required by the mixed-trust framework.

In this work in progress, we limit our focus to analyze the case where, after I_i prediction/pre-computation jobs of the HT, there will be at least one enforcer job of the HT. This assumption will be lifted in our future work.

With this new system model, we need to calculate the response times of an HT in two different cases:

1) $R_i^{\kappa,e}$— the response time of the HT κ_i for the case that it performs enforcement.
2) $R_i^{\kappa,p}$— response time of the HT κ_i for the case that it performs prediction.

We need to compute these two versions of the response times considering that HTs are scheduled non-preemptively in the HV. Once we have computed these, we will use them to assign E_i for each GT τ_i, for example: $E_i = D_i - \max(R_i^{\kappa,e}, R_i^{\kappa,p})$.

B. Enforcement

To calculate the response time of the HT κ_i performing enforcement, $R_i^{\kappa,e}$, we focus on the existing analysis [2] where the HTs are always assumed to run enforcer jobs. This is basically the same as equation (3) but replacing κC_i with κC_i^m:

$$R_i^{\kappa,e} = \max_{q \in \{1 \ldots \left\lceil \frac{t_i^\kappa}{T_i} \right\rceil\}} (w_{i,q}^\kappa + \kappa C_i^m - (q-1)T_i), \tag{10}$$

C. Prediction

For the case that the HT performs prediction, we need to develop new equations. In this case, the deadline encodes the time by which we need to complete the pre-computation in order to obtain the benefit of the improved enforcement.

Let us now discuss computation of the response time for the case that the HT κ_i of a mixed-trust task μ_i performs prediction. Let $t_i^{\kappa,p}$ denote the maximum duration of a level-i active period. Following [1], we calculate $t_i^{\kappa,p}$ as the smallest solution of:

$$\begin{aligned} t_i^{\kappa,p} = & \max_{j \in \kappa L_i} \kappa C_j^m + \left\lceil \frac{t_i^{\kappa,p}}{T_i} \right\rceil \kappa C_i^p + CP(i, t_i^{\kappa,p}) \\ & + \sum_{j \in \kappa H_i} (\left\lceil \frac{t_i^\kappa}{T_j} \right\rceil \kappa C_j^p + CP(j, t_i^{\kappa,p})), \end{aligned} \tag{11}$$

where κL_i is the set of HTs with lower priority than κ_i, κH_i is the set of HTs with higher priority than κ_i, $\kappa C_j^m = \max(\kappa C_j^p, \kappa C_j^e)$, and

$$CP(j, t_i^{\kappa,p}) = \begin{cases} \left\lceil \frac{t_i^{\kappa,p}}{T_j \cdot\ I_j} \right\rceil (\kappa C_j^e - \kappa C_j^p) & \text{if } \kappa C_j^e > \kappa C_j^p \\ -\left\lfloor \frac{t_i^{\kappa,p}}{T_j \cdot\ I_j} \right\rfloor (\kappa C_j^p - \kappa C_j^e) & \text{otherwise.} \end{cases} \tag{12}$$

Given that a lower-priority HT may be running when a higher-priority HT arrives, (11) takes into account the maximum interference from one job of a lower-priority task. Note that we do not make any assumption about whether the WCET of a prediction job (κC_j^p) is larger than that of an enforcer job (κC_j^e). Instead, (11) first accounts for the cumulative execution time considering κC_j^p (the second term) and then makes a compensation by using the function CP, which returns a positive value if $\kappa C_j^e > \kappa C_j^p$, and negative otherwise.

Let $w_{i,q}^{\kappa,p}$ denote the latest starting time of the q^{th} prediction job $\kappa_{i,q}^p$ in the level-i active period. Then, from [1], we calculate $w_{i,q}^{\kappa,p}$ as the smallest solution of:

$$\begin{aligned} w_{i,q}^{\kappa,p} = & \max_{j \in \kappa L_i} \kappa C_j^m + (q-1)\kappa C_i^p + Q(i, q-1) \\ & + \sum_{j \in \kappa H_i} ((\left\lfloor \frac{w_{i,q}^\kappa}{T_j} \right\rfloor + 1)\kappa C_j^p + CP_w(j, w_{i,q}^{\kappa,p})), \end{aligned} \tag{13}$$

with

$$Q(i,q) = \begin{cases} \left\lceil \frac{q}{I_i} \right\rceil (\kappa C_i^e - \kappa C_i^p) & \text{if } C_i^e > \kappa C_i^p \\ -\left\lfloor \frac{q}{I_i} \right\rfloor (\kappa C_i^p - \kappa C_i^e) & \text{otherwise,} \end{cases} \tag{14}$$

and

$$CP_w(j, w_{i,q}^{\kappa,p}) = \begin{cases} (\left\lfloor \frac{w_{i,q}^{\kappa,p}}{T_j \cdot\ I_j} \right\rfloor + 1)(\kappa C_j^e - \kappa C_j^p) & \text{if } \kappa C_j^e > \kappa C_j^p \\ -\left\lfloor \frac{w_{i,q}^{\kappa,p}}{T_j \cdot\ I_j} \right\rfloor (\kappa C_j^p - \kappa C_j^e) & \text{otherwise.} \end{cases} \tag{15}$$

The response time can then be calculated as follows. For the jobs in the level-i active period, we can move the arrival times of the κ_i jobs to be as early as possible; this may change the schedule but neither the duration of the level-i active period nor the starting time of each κ_i job decreases. Hence, it holds that each $\kappa_{i,q}$ in the active period arrives $(q-1)T_i$ time units after the level-i active period starts.

For each job of κ_i, we can add κC_i to its starting time and then subtract the arrival time of this job, which yields the response time of the job. Then we upper-bound the response time of κ_i as:

$$R_i^{\kappa,p} = \max_{q \in \{1 \ldots \left\lceil \frac{t_i^{\kappa,p}}{T_i} \right\rceil\}} (w_{i,q}^{\kappa,p} + \kappa C_i^p - (q-1)T_i). \tag{16}$$

IV. Guest-Task Scheduling

It is worth remembering that safety does not depend on the GT. Hence, the GT scheduling guarantee is aimed at ensuring that the system is able to make progress (e.g., the car navigates the road and does not brake constantly) with the GT computation even if we cannot trust it with safety. Following

the same spirit, there are two ways to think about guest task scheduling: (i) considering the worst-case enforcement I_i interarrival, i.e., we will always use κC_i^m, and (ii) assume that the steady state is to have one enforcement job every I_i pre-computing jobs. Since we expect that for many tasksets $\kappa C_i^p > \kappa C_i^e$ using a fixed I_i can give us better schedulability.

In the next two subsections we present these two sets of schedulability equations.

A. Worst HT Execution

In this case the only difference will be the replacement of κC_i by κC_i^m in all the GT schedulability equations of Section II.

B. Fixed I_i parameter

This case involves modifying the interference equations as follows.

The re-definition of the request-bound function for our model is given by:

$$\text{rbf}_i^y(t,b) = \begin{cases} \left\lceil \frac{t-(T_i-E_i)}{T_i} \right\rceil^+ C_i b + \left\lceil \frac{t}{T_i} \right\rceil \kappa C_i^p + CP(i,t) & \text{if } y = E, \\ \left\lceil \frac{t}{T_i} \right\rceil C_i b + \left\lceil \frac{t-E_i}{T_i} \right\rceil^+ \kappa C_i^p + CP(i,(t-E_i)^+) & \text{if } y = A, \end{cases} \tag{17}$$

where $\lceil x \rceil^+ = \max(0, \lceil x \rceil)$ and $(x)^+ = \max(0,x)$.

Then, similar to (5), we compute $t_i^{g,x}$ as the smallest solution of:

$$t_i^{g,x} = \left(\sum_{j \in L_i} \text{rbf}_j^E(t_i^{g,x}, 0) \right) + \text{rbf}_i^x(t_i^{g,x}, 1) + \sum_{j \in H_i} \max_{y \in \{E,A\}} \text{rbf}_j^y(t_i^{g,x}, 1), \tag{18}$$

where L_i and H_i contain the tasks with lower and higher priority (respectively) than τ_i. Given τ_i and level-i busy period, we refer to job q as the q^{th} job with a GT arrival in the level-i busy period. For each τ_i, and $x \in \{E, A\}$, let $w_{i,q}^{g,x}$ denote the maximum finishing time of job q of task τ_i, relative to the start of the maximum level-i busy period, such that this level-i busy period starts with a job of the HT or the GT of τ_i arriving (x indicates which). Then, similar to (6), we compute $w_{i,q}^{g,x}$ as the smallest solution of:

$$w_{i,q}^{g,x} = \left(\sum_{j \in L_i} \text{rbf}_j^E(w_{i,q}^{g,x}, 0) \right) + qC_i + (q - 1 + \mathcal{B}_{(x=E)})\kappa C_i^p + Q(i, q - 1 + \mathcal{B}_{(x=E)}) + \sum_{j \in H_i} \max_{y \in \{E,A\}} \text{rbf}_j^y(w_{i,q}^{g,x}, 1), \tag{19}$$

where $\mathcal{B}_\phi$ is an indicator function that returns 1 if ϕ is true and 0 otherwise.

For each τ_i and for each $x \in \{E, A\}$, let $R_{i,q}^{g,x}$ denote the maximum response time of job q of τ_i such that this level-i busy period starts with the arrival of a job of the HT or the GT of τ_i (x indicates which). Then, similar to (3), we compute $R_{i,q}^{g,x}$ as:

$$R_{i,q}^{g,x} = w_{i,q}^{g,x} - ((q-1)T_i + \mathcal{B}_{(x=E)}(T_i - E_i)). \tag{20}$$

For each τ_i and for each $x \in \{E, A\}$, let $R_{i,q}^x$ denote the maximum response time of τ_i, such that this level-i busy period starts with the arrival of a job of HT or GT of τ_i (x indicates which). Then, similar to (8), we compute $R_i^{g,x}$ as:

$$R_i^{g,x} = \max_{q \in \left\{ 1 \ldots \left\lceil \frac{t_i^{g,x} - I_{x=E}(T_i - E_i)}{T_i} \right\rceil \right\}} R_{i,q}^{g,x}. \tag{21}$$

Finally, the response time of a GT is:

$$R_i^g = \max_{x \in \{E,A\}} R_i^{g,x}. \tag{22}$$

V. Concluding Remarks

This paper presents our work in progress on the extension of the real-time mixed-trust scheduling model to support predictive tasksets. Our discussions with autonomous systems researchers have indicated a need for flexible runtime assurance mechanisms that take into account predictive computation. This paper offers a flexible mechanism from a timing point of view and formalizes the associated timing analysis. Consequently, the paper represents an important step toward the development of full verification of autonomous systems.

VI. Acknowledgements

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
DM20-0898

References

[1] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 2007.

[2] D. de Niz, B. Andersson, M. Klein, J. Lehoczky, A. Vasudevan, H. Kim, and G. Moreno. Mixed-trust computing for real-time systems. In *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–11, Aug 2019.

[3] A. K. Mok and D. Chen. A multiframe model for real-time tasks. In *17th IEEE Real-Time Systems Symposium*, pages 22–29, 1996.

[4] A. Vasudevan, S. Chaki, L. Jia, J. M. McCune, J. Newsome, and A. Datta. Design, implementation and verification of an eXtensible and Modular Hypervisor Framework. In *2013 IEEE Symposium on Security and Privacy, SP*, 2013.

[5] A. Vasudevan, S. Chaki, P. Maniatis, L. Jia, and A. Datta. überspark: Enforcing verifiable object abstractions for automated compositional security analysis of a hypervisor. In *25th USENIX Security Symposium (USENIX Security 16)*, 2016.