Semester papers Summary

deepwaterooo

December 18, 2023

Contents

1 Goals Statement

- **The overall idea of this project** is to design a **priority-driven isolation mechanism** where multiple isolated "Zones" (i.e., real-time applications) can run in parallel and meet their timing/safety guarantees. We want to design this solution for **ARM-based microcontrollers WITHOUT hardware-supported security extensions ("legacy systems").**
  - Some isolation techniques exist for general-purpose (i.e., non-real-time) systems.
  - Our goal is to **make the MultiZone framework "real-time" aware** (i.e., enable a **priority-scheduling mechanism for "Zones".** where each Zone will be assigned **user-defined priority and timing constraints (deadlines).**
  - We want to extend MultiZone with **a priority-driven scheduler** that **allocates processor times for Zones** based on **priority and ensures Zone allocation time meets deadline constraints.**
- The mentioned literature mainly focused on:
  - understand real-time scheduling techniques
  - how to integrate **priority scheduling algorithms** and **security solutions** for similar platforms (such as real-time containers and virtual machines) implemented by other researchers.

2 Security Solutions

- **Isolation to programs** in embedded devices: These approaches generally **have limitations in achieving both real-time performance and memory isolation.**

- **Automatic sensitive instruction identification:** To preserve real-time properties, it is not always acceptable to escalate execution privilege for all sensitive instructions, as those instructions may be executed very frequently in real-time MCS.
- **MINION**: which automatically reduces memory spaces of real-time MCS while meeting real- time constraints. The key idea behind MINION is an efficient memory view switching mechanism for memory isolation, which avoids frequent privilege mode switching. The prototype of MINION was able to meet the real-time constraints while significantly reducing memory attack surfaces.

3 Designs:

3.1 Abstracted Simple Zones: MMU facilitated private process isolated memory space (zones)

- To design and implement a simple zones system, to limit the cross zone communication overhead, and meet real-time requirements.
- **Process memory isolation and Kernel memory isolation**: they are the 2 major forms of memory protection to escape from being attacked. However, it is challenging for real-time MCS to support such memory isolation in practice, due to hardware and performance constraints.
- I believe the fundamental implementation of **multi-isolated zones, are still zoned-memory regions, thus isolated memory space.**
- **MINION: The underlying View Switcher**, to my current understanding, is still process memory space isolation. Compared with MultiZone-SDK well-managed continuous memory regions for each zone, MINION per process has no contiguous memory blocks. I consider it as simple as dynamic process private process memory allocation, though here MINION limited per process memory space as small as possible to make both goals achievable at same time.
- Based on these fundamentals, for **realtime MCS**, it is fairly easy to design **logically simple yet secure memory management (down from MMU hardware/firmware fundamental)** for zones, considering zones as private/isolated tasks, with achieving "real-time" awareness.
- **from scratch ARM MCS simple Secure system:**
- Two simple Modes: **Kernal Mode and User mode**
  - User mode, all zones/tasks isolated/private;
  - Kernel mode shared memory space among all tasks.
  - But User mode applications can only enter kernel mode through ARM system interruptions – prioritized scheduling-rescheduling.
- Switching Modes, or even switching tasks, due to context switch overhead, would be the major factor limiting ARM real-time responsiveness for multiple tasks here.

3.2 Priority Scheduling: Prioritized schedulable real-time scheduler

- According to problem statement, each zone/application may have I/O requirements, or resource requirements etc blocking factors, **we may need to sublet each zone/application tasks: prioritized with ranked/numbered priorities, and they are preemptive.** When tasks need to acquire certain resources, tasks could be preempted out by CPU during waiting for resources waiting time. **Thus a task may be executed by CPU during multiple execution time slots, though these time slots may not necessarily be equal, but rather dynamic.**

- To ensure MCS responsiveness and meeting deadlines, we may design and implement dynamic prioritized scheduling. Here I mean during the parallel execution of different tasks, the priority of each task is not static, but may change during execution according to environment tasks priorities.

  - To ensure a task is able to meet its deadline with certain bounds, a task may increase its priority to get CPU time to meet the deadline. And as it is preemptive, this task could be switched out by CPU for offering execution time to other higher prioritized tasks as well.

- The ARM Uni-Processor board is currently the highest priority task.

- For MCS to meet deadlines and make tasks responsive, the scheduler supports auto-task switching whenever there is any higher priority task in readyQueue.

- The scheduler switching tasks will produce **context switch overhead CPU time usage**, which is **the main factor affecting "real-time" aware achievement**. Therefore, there is a balance between **longest possible continuous execution time slots for current task** to limit context switch overhead, and **dynamic priority adjustments frequency** to be "real-time" aware and meet task deadlines.

- Scheduler components:

  - Workloads: N (zones/applications/tasks);
  - Resource Model: Not very clearly understanding the importance in this problem, but some meaning of total II time C time units available for each application.
  - Algorithms: propose earliest deadline first, but preemptive to dynamically increased priorities to meet deadline, or try to acquire necessary resources at relatively earlier stage before deadline. Prioritized (EDF) to ensure real time tasks meeting deadlines.

4 IEEE article latex

- There is solutions for using **emacs org-mode** related configurations so that **IEEE articles pdfs can be developed from and exported directly from emacs org-mode.** But I underestimated the difficulty, and quite a few posts and git-repositories available configuring it. But usually geeks do not share as detailed configurations as I would need. There are missing configurations or modules here or there. It is right now slightly challenging for me to configure it immediately, but I will spend about 1 week, or even 2-3 weeks during the winter holiday to configure it ready for use, for my spring Semester.

- This semester this summary, I hope the advisor could help telorant it as a plain pdf file for 1 summary only.