FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Design and Prototypical Implementation of a High-Level Synchronization Component for Dynamic Updates of Task Run Queues in L4 Fiasco.OC/Genode

Gurusiddesha Chandrasekhara

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Design and Prototypical Implementation of a High-Level Synchronization Component for Dynamic Updates of Task Run Queues in L4 Fiasco.OC/Genode

Entwurf und Prototypische Implementierung einer High-Level Synchronisationskomponente fuer dynamische Updates von Task-Warteschlangen in L4 Fiasco.OC/Genode

Author:	Gurusiddesha Chandrasekhara
Supervisor:	Prof. Dr. Uwe Baumgarten
Advisors:	Daniel Krefft,M.Sc.; Sebastian Eckl,M.Sc.
Date:	October 15, 2016

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, October 15, 2016 Gurusiddesha Chandrasekhara

Acknowledgments

Abstract

Abstract abstract abstracting the abstractness

Contents

1 Introduction

The implemented high-level synchronization component in this master thesis is part of the KIA4SM project of the department of operating systems. The work aims to implement a method for dynamic update of tasks ready queues in L4 Fiasco.OC/Genode while providing a synchronized access to them.

1.1 Overview of KIA4SM project

KIA4SM (stands for Cooperative Integration Architecture for Future Smart Mobility Solutions) is a research project at the department of operating systems. Traditionally Cooperative Intelligent Transport Systems have been built on heterogeneous systems. KIA4SM aims to provide an architecture of having an homogeneous platform for heterogeneous systems.

KIA4SM focuses on developing systems for the interaction and coordination between computer-assisted vehicles, be it partially or fully autonomously functioning actors. KIA4SM aims to improve on the ad-hoc networking between vehicles

The final vision of the project is illustrate in figure 1.1. The goals of the project are,

- A common platform as foundation for device in- dependent (vehicles, mobile devices, traffic and transport architecture) provision and execution of software-based functionality
- Mechanisms that allow for online dynamic reconfig- uration, based on
  - en-/disabling and relocation/migration of software-based functionality
  - adaptive (data-centric) routing policy
  - flexible scheduling of tasks per ECU

In order to achieve the goals of the project a number of different methods have been applied. This has led to the application of Organic computing paradigm. igner and user. Organic Computing (OC) has the vision to address the challenges of complex distributed systems by making them more life-like (organic), i.e. endowing them with abilities such as self- organization, self-configuration, self-repair, or adaptation. In order to realize this, universally applicable Electronic Control Units (ECU) and a common run-time environment are used which provides Hardware/Software Plug-and-Play properties.

1.2 Motivation

There are number of micro controllers used for different calculations in a modern vehicle, this project aims to replace them with use more power-full and standardized hardware universally applicable ECUs.

OC approach proposes a Observer-Controller architecture similar to MAPE architecture(monitor, analyze, plan, execute) . An observer collects the data from the all the ECUs and computes and generates indicators where a controller takes a decision based on the indicator, generates an action.

One such action of the controller is to decide what tasks should be executed at what time in order to meet the aforementioned requirements of the safety critical systems. So the controller decides and produces a run/ready queue (RQ). There needs to be a method which allows to safely update the scheduler run queues of the system.

Therefore it is essential to be able to add threads and modify the execution order during operation time, A flexible thread handling is also required for example in case a ECU is malfunctioning. In this case it would be possible to swap the threads from the malfunctioning one to working ones. So it is important to generate new ready-queues based on the information we are receiving from the other ECUs in the grid, and then exchange them with the actual ready-queue the scheduler uses.

1.3 Thesis structure

The thesis is structured in a way that the reader understands the importance of the work carried out here and also the surrounding concepts before delving in to the specifics (inverted traingle).

The second chapter summarizes the related work on the state of the art algorithms for synchronization and different types of schedulers in use and at the end of the section an evaluation of the synchronization methods is provided inorder to chose the best possible approach for the existing project.

The third chapter explains the Genode and L4 Fiasco.OC details in brief, in order for the user to have an overview of the system.

The fourth chapter deals with the design considerations along with implementation details.

The fifth chapter is dedicated to explains testing method and the results obtained. And the final chapter concludes the thesis with the limitations and future work to be done.

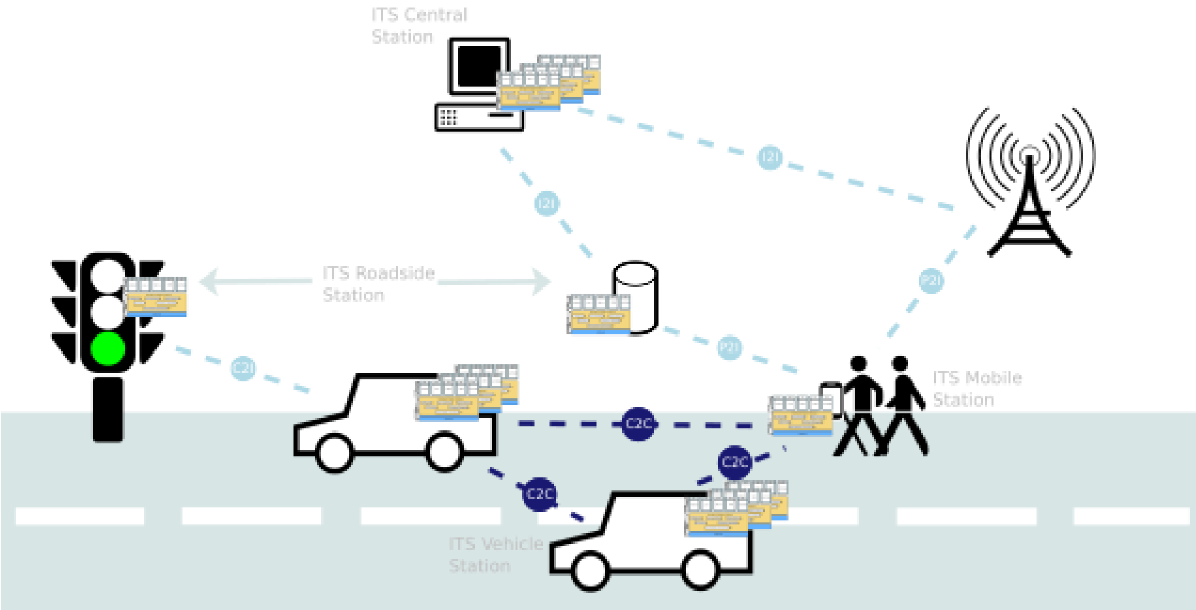

Figure 1.1: KIA4SM vision - homogeneous platform for heteroge- neous devices

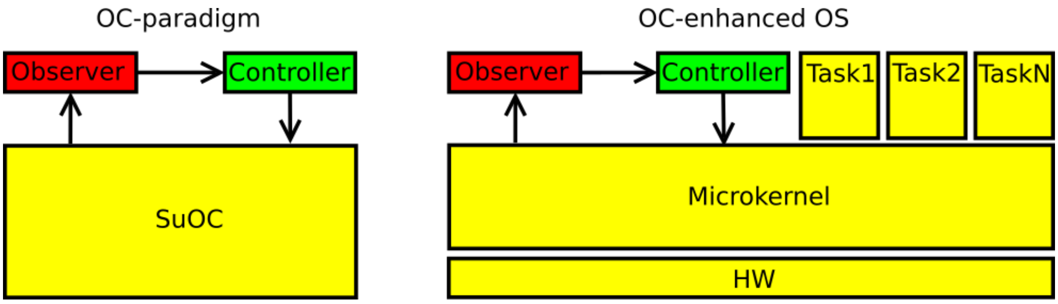

Figure 1.2: Organic Computing: Applying the Observer/Controller pattern to existing mi-crokernel architecture

2 Related Work

This chapter explains the previous work and concepts which led to the development of synchronization component to the KIA4SM project. It explains the synchronization algorithms available and makes a comparative study of these algorithms. The comparative study is based on the research work of many papers which are referred here. An attempt is made to pick the best choice algorithm for the work.

2.1 Synchronization of L4 fiasco tasks

The thesis is largely based on the work of Robert Häcker, who in his bachelor thesis *"Design of an OC-based Method for efficient Synchronization of L4 Fiasco.OC Mircrokernel Tasks"* [**?**], explains the design of a scheduler best suited scheduler for KIA4SM project and also gives comparison study of the different schedulers and synchronization methods suited for updating the task ready queue.

He suggests Modified-Maximum-Urgency-First algorithm as the best choice for KIA4SM project due to the importance of safety and security in embedded systems. After comparing the synchronization algorithms the sequential lock technique has been chosen for the better control it gives. He has suggested to verify the practical implication of Read-Copy Update(RCU). At the end he proposes a design for the existing system including aforementioned scheduler(MMUF) and sequential locks.

This work is an extension of Häcker's findings. However, the focus of the thesis it to develop a good synchronization method, the implementation of the scheduler is not carried out. As go forward I make reference to Robert Häcker's work.

Some of the ideas and code knowledge is taken from the Valentin Hauner's bachelor's thesis *"Extension of the Fiasco.OC microkernel with context-sensitive scheduling abilities for safety-critical applications in embedded systems"* [**?**]. In this thesis he added EDF scheduling strategy. Though his thesis concentrated on using it in Genode and L4RE environment, it helped in understanding the scheduler.

2.2 Synchronization Methods

To do the justification to many synchronization methods are studied and are explained in this section. The synchronization categories are divided based on the methods they apply. The figure shows the different types of synchronization methods along with examples.

2.2.1 Lock-based algorithms

2.2.2 Lock-Free algorithms

3 Genode OS framework and L4 Fiasco.OC

Genode operating system and L4/Fiasco is used as the preferred kernel in this project. Since the kernel itself acts a type-I Hypervisor which allows for partitioning of different software components via separated container, making them work on user mode leve. To understand the design and implementation of this project, reader has to be aware of few important concepts in both Genode operating system and L4 Fiasco.OC kernel. Many concepts here are the relevant parts explained from the Genode book according to author's understanding, but the reader can always refer to the book for detailed explanation.

3.1 Genode Operating System Framework

Genode Operating system framwork is

4 Design and Implementation

5 Testing and Results

6 Conclusion

6.1 Limitations

6.2 Future Work

List of Figures

List of Tables