electronics

MDPI

Article

Hybrid Inference Based Scheduling Mechanism for Efficient Real Time Task and Resource Management in Smart Cars for Safe Driving

Sehrish Malik [1], Shabir Ahmad [1], Bong Wan Kim [2], Dong Hwan Park [2] and DoHyeun Kim [1,*]

[1] Computer Engineering Department, Jeju National University, Jeju-si 63243, Korea; serrym29@gmail.com (S.M.); shabir@jejunu.ac.kr (S.A.)

[2] Electronics and Telecommunications Research Institute, Daejeon-si 34129, Korea; kimbw@etri.re.kr (B.W.K.); dhpark@etri.re.kr (D.H.P.)

* Correspondence: kimdh@jejunu.ac.kr; Tel.: +82-64-754-3658

Received: 4 March 2019; Accepted: 18 March 2019; Published: 21 March 2019

Abstract: In recent years, the focus of the smart transportation industry has been shifting towards the research and development of smart cars with autonomous control. Smart cars are considered to be a smart investment, as they promote safe driving while focusing on an alternate transportation fuel resource, making them eco-friendly too. Safe driving is one of the crucial concerns in autonomous smart cars. The major issue for the better provision of safe driving is real time tasks management and an efficient inference system for autonomous control. Real time task management is of huge significance in smart cars control systems. An optimal control system consists of a knowledge base and a control unit; where the knowledge base contains the data and thresholds for rules and the control unit contains the functionality for smart vehicle autonomous control. In this work, we propose a hybrid of an inference engine and a real time task scheduler for an efficient task management and resource consumption. Our proposed hybrid inference engine and task scheduler mechanism provides an efficient way of controlling smart cars in different scenarios such as heavy rainfall, obstacle detection, driver's focus diversion etc., while ensuring the practices of safe driving. For the performance analysis of our proposed hybrid inference based scheduling mechanism, we have simulated a non-hybrid version with the same system constraints and a basic implementation of inference engine. For performance evaluation, CPU time utilization, tasks' missing rate, average response time are used as performance metrics.

Keywords: real-time tasks; inference engine; task scheduling; smart cars; smart control systems; periodic tasks; event-driven tasks

1. Introduction

With the advancements towards autonomous driving vehicles, the focus of the smart transportation industry is being shifted towards the research and development of smart cars with autonomous control. Smart cars are considered to be a smart investment as they promote safe driving while focusing on an alternate transportation fuel resource, making them eco-friendly too. Driving a vehicle requires the utmost focus of the driver, as it is an extremely intricate task. Safe driving is tremendously important and must not be taken for granted as it could be a matter of life and death. Due to increasing number of vehicles on roads, traffic sometimes becomes extremely congested; requiring a complete focus and full awareness of the driver of their surroundings without being distracted by anything like mobile calls or radio, etc. According to a survey, around 3 million people suffer injuries and around 40 thousand people lose their lives due to car accidents in the United States

every year [1]. Therefore, ensuring the safety of everyone is the most essential element while driving a vehicle. The major issue for better provision of safe driving is real time tasks management and an efficient inference system for autonomous control. Real time task management is of huge significance in smart cars control systems.

As driving is a multitasking job, a driver must simultaneously control the speed, avoid hazards, set up directions and do strategic planning; an extremely experienced human being can also get distracted by environmental conditions or by any other elements. Hence, the need for some kind of automation in the transportation industry emerged, which can ensure the safety of vehicles and share responsibility with the drivers. Car companies responded to this need by vigorously supporting car safety features.

With the evolution of time and the development of smart technologies, automobiles or electrical vehicles being a predominant mean of transportation witnessed a boost in the emergence of advanced and innovative car models. The necessity of safe driving has resulted in the potential need for substitute technologies in automobiles such as smart cars. Internet of things (IoT) and the power of computer vision are the primary actors that enabled the existence of smart cars. These smart cars are autonomous vehicles; an autonomous vehicle takes over the driver's responsibilities by sensing the environment, and navigating on its own.

Design and implementation of the precise control systems in the smart cars is very crucial. The advanced control system collects data from different sensors and actuators; and interprets the information in order to accurately identify suitable navigation paths, hazards, and appropriate signage. The control system of a smart car usually manages the decisions based on the inputs from different sensors e.g., GPS, rear camera, and radar sensors etc., and enables the actuators like distance control module, steering control, wipers control etc., to work accordingly (Figure 1).

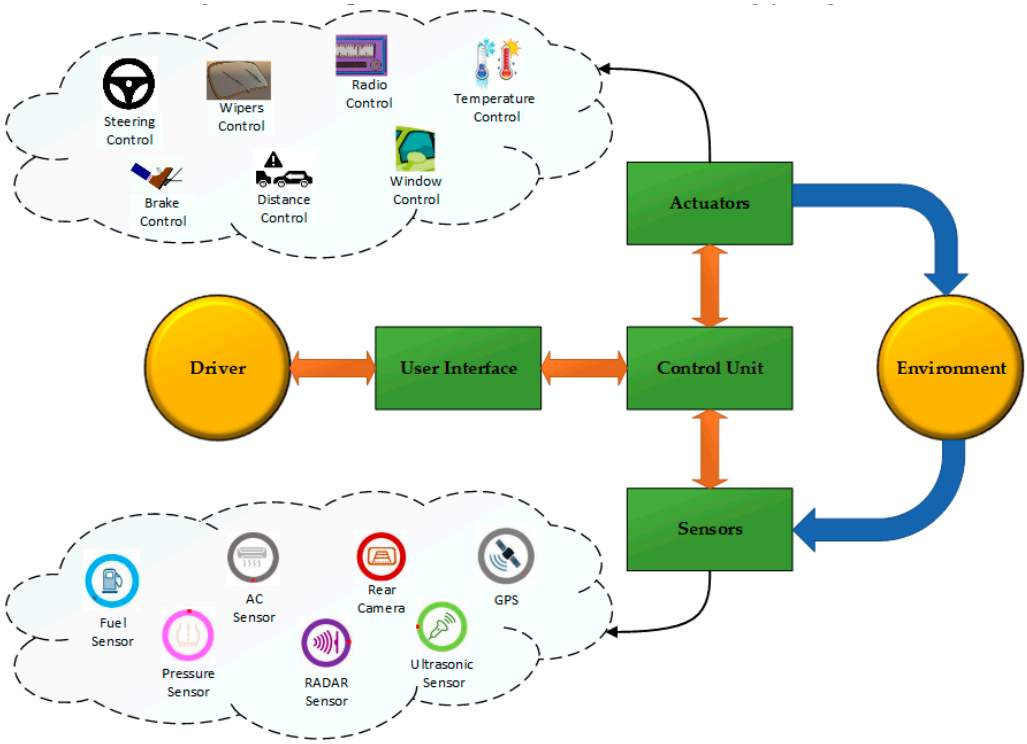

Figure 1. General safe driving conceptual model for smart cars.

The overall flow of the system for a smart car is shown in Figure 1; the basic components are sensors, actuators and a control unit. The control unit works according to the environmental conditions, the sensors collect data from the environment e.g., temperature, location etc. and this information are passed on to the control unit to make control decisions and the controller implements the control

commands to actuate different functionalities e.g., controlling speed, steering wheel etc. The driver gets alerts through a user interface. This implies that an interface bridges between the real world and the system, and thus an inference system must be proficient enough to plan and produce control commands. An inference engine is a component of the inference system that applies logical rules to the knowledge base to deduce new information. However, only a tiny percentage of cars on the road today have these driver alert systems and it will take a decade for this new technology to be commonplace in most cars across the globe.

Efficient control systems demand a precise interaction between lower level nodes and a reliable and robust data sharing between the sensing and control units for decision making with fast control loop update rates. Since every action taken by an autonomous vehicle system results in a new scenario, the action of each control function must be precisely synchronized. Previously, the inference engine for smart control and the scheduler has been separated in design, resulting in the use of more resources. In this work, we propose a hybrid of inference engine and task scheduler for efficient task management and resource consumption. Our proposed combined mechanism provides an efficient way of controlling smart cars in different scenarios and making them self-sufficient. Our proposed hybrid system provides complex task management in embedded IoT systems; giving flexible parameter setting options to consider all the system constraints for embedded IoT systems of different nature, and maximizing the performance.

The rest of the paper is divided as in the following way: Section 2 presents the related works, while Section 3 presents the proposed inference based scheduling mechanism. In Section 4, we provide the simulation of the proposed system. Simulation visualization is presented in Section 5, performance analysis is presented in Section 6 while Section 7 concludes the paper with discussions.

2. Related Work

Smart cars have been a hot topic in the past decade; we can observe that many researchers either from academia or from industry have been focusing their research efforts on smart cars. Inference engines, a primary and core measure of smart cars systems, are also being part of researcher efforts. In reference [2], an inference engine is being proposed that works for rule based expert systems. This inference engine is proved to be producing ideal, accurate and general rules according to time. The reasons for its accurate results are: the engine is being invoked when there is a change in any feature, conditions or rules are tested when values are assigned to features, every time a condition is altered, and the corresponding rule is examined, and actions will be performed once its rule is invoked. The predominant reason for producing optimal results is it does only the tasks which it is asked to perform, it does not do stuff which is not required. This means until and unless a features value is changed, and the rules are observed only if the corresponding conditions meet the criteria. Smart cars follow a complex mechanism, there might occur many complications and challenges if there happens to be malfunction or any failure; implying that inference engines must be capable of dealing with such failures. Some researchers have also focused on this area for proposing different tools or applications that can help avoid such circumstances.

In another study [3], guiding a vehicle on the limited access highways, is explored and the applicability of expert or inference systems is studied. The vehicle considered here is a smart car that takes input form sensors that are able to detect the surrounding traffic situations, signs, and road conditions etc. It has a control system and actuators. The proposed system is rule based and implements backward chaining inference engine for rules inference, and these inferred rules are then processed by a knowledge based compiler that performs reasoning. A simulator is also implemented to simulate a vehicle's behavior on limited access highways.

Researchers in reference [4] proposed Automated Car Failure Diagnosis Assistance (ACFDA); an agent-based inference engine for cars failure diagnosis expert system. The agent here tries to maximize the productivity and competence of the complete routine of the ACFDA system by completing a number of inferencing tasks and by tweaking the inferencing logical flow. This system or

toolkit focuses on finding the reason for the failure. This helps the driver to know the exact source of the failure and act accordingly.

A smart car is a comprehensive integration of many different sensors, control modules, actuators, and so on. Many systems to support the purpose of smart cars were proposed e.g., a driver assistant system from BMW that assists in lane change warnings and parking [5], an intelligent driver's system proposed by Mercedes-Benz to monitor the car's surroundings using stereo cameras and radar sensors [6], a driver's assistant system based on traffic situation and conditions is proposed in reference [7], and a system focusing on progressive safety expertise with installations of a pre-collision system, dynamic driving, automated brake assistance and parking assistance is proposed in reference [8].

A general view of smart cars is presented in reference [9], focusing on context awareness; as a model focusing on the complex environment of driving is proposed that can learn the context in a hierarchal manner. In order to ensure safe driving, many studies have been done on the driver's distraction. The term driver's distraction can be defined as diversion of the driver's attention towards non-driving activities [10]. The distractions can be classified into four categories: visual distraction, auditory distraction, biochemical distraction and cognitive distraction [11]. The literature work from references [12–21] present driver's distraction detection systems based on different measures such as driving behavior, driver's physiological or the hybrid of both approaches.

The work in reference [22] surveys a cyber-physical vehicle system (CPVS) for cyber and physical schemes for resource consumption, time varying patterns, scheduling, vehicle control and task and motion planning. In reference [23], the safety level of roundabouts is evaluated for autonomous vehicles by simulating the roundabouts win combination to the conventional vehicles. The simulation highlights the differences in traffic parameters as queue length, average speed and delay in stopping. Safety is analyzed based on the potential conflicts at the roundabouts. A platform for safe autonomous driving is proposed in reference [24], with flexible architecture capable of integrating a wide variety of actuators and sensors for testing purposes. The work provides a complete navigation system for test scenario and two flexible routing algorithms are also proposed. The research work in reference [25] aims to solve the speed planning problems for autonomous vehicles. After summarizing the existing constraints in the speed planning, the work proposes a mathematical model for a general speed planning of autonomous vehicles based on the summarized constraints.

3. Inference based Scheduling Mechanism

In this section, we present our proposed inference based scheduling mechanism for the autonomous control of smart cars. Figure 2 below shows the conceptual model of the inference based scheduling mechanism in an optimal scheduling inference system for smart control of EVs. The main components of the model are a smart car sensing data unit, task modeling unit, scheduling unit, inference system unit, control system unit, and optimal output.

The smart car sensing data unit collects the input data from the smart car sensing environment. This data can be various kinds, e.g., environmental data, road state data, traffic state data, driver health state data, smart car physical self-state data etc. Task modeling unit models the smart car input data into periodic, event and complex nature tasks along with scheduling required parameters such as arrival time, execution time, deadline, priority and periodic/event tags. Scheduling unit implements a scheduling policy as per which the arrived tasks are to be executed at the CPU. The inference system unit implements the inference engine, where all the rules are to be followed according to different scenarios and threshold values are defined. The control system unit implements the control functions for the smart vehicle; which are triggered after firing the rules at inference engine. Optimal output generates the list of activated control output tasks and notifications/alert output tasks.

Our proposed hybrid inference based scheduling mechanism aims to meet all possible scenarios in a smart car environment with safety measures taken into account too. The proposed system satisfies the basic requirements of ISO26262, which is an international standard for functional safety of road

vehicles [26], as explained in sub-sections below. In Section 3.1, the task model is given for the smart cars' task classification based on severity of the task or event. In Section 3.2, we give the scheduling policies, where we propose a custom made scheduler to best meet the scheduling of different priority tasks of different scenarios, ensuring that no chance is taken in critical scenarios in order to implement safe driving. In Section 3.3, we propose our inference engine's design, which defines the driving rules for controlling smart cars, along with exceptions to deal with unexpected events in best possible way. In Section 3.4, we present our hybrid agent which bridges the scheduling module with inference engine module.

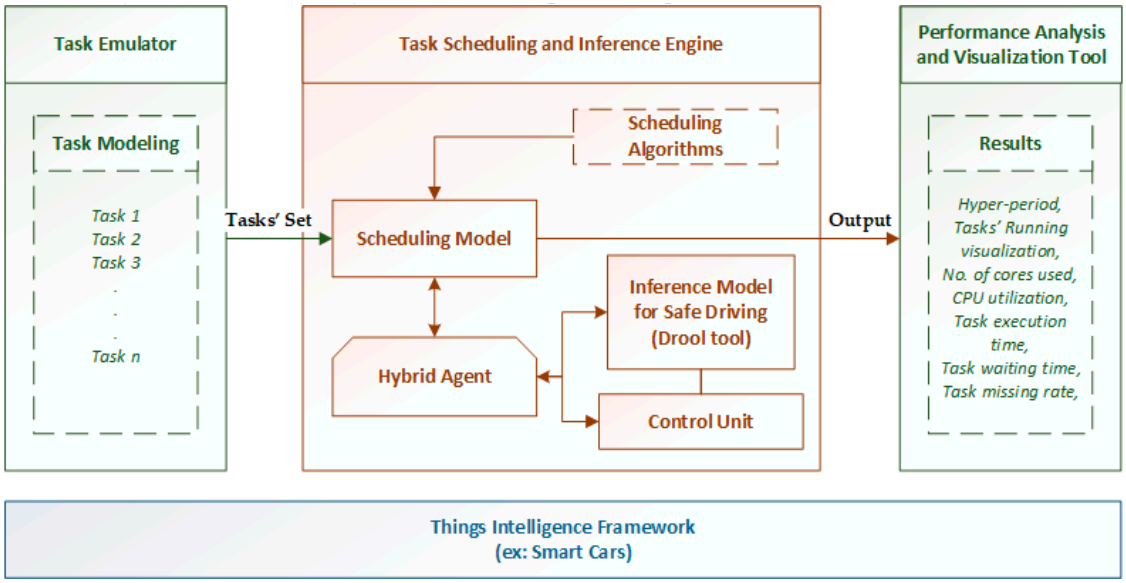

Figure 2. Intelligence framework model for hybrid task scheduler and inference engine.

3.1. Task Model

There are two main types of tasks as periodic tasks and event-driven tasks. Each task must have a start time, execution time, deadline, priority/urgency value and period (if periodic task).

A periodic task and its nth periodic execution are denoted by PTi (AT, ET, D, P, PB) and PTin, respectively. An event driven task and its nth execution are denoted by ETi (e, AT, ET, D, U) and ETin, respectively. Table 1 below presents the task set parameters for periodic and event driven tasks.

Table 1. Task set parameters for periodic and event driven tasks.

Parameters	Detail
i	Identifier of a task.
AT	Arrival time of a task.
ET	Execution time of a task.
D	Deadline of a task.
P	Period of a periodic task.
PB	Priority bit for a periodic task. - PB = 1 indicates task has priority over other periodic task with PB = 0 - PB = 0 indicates task is a normal periodic task
e	The event that triggers an event driven task.
U	Urgency factor of an event driven task - U = 1 indicates task is urgent and should be executed ASAP - U = 0 indicates non-urgent event driven tasks

3.2. Scheduling Model

In this sub-section, we elaborate the scheduling model for smart car safety driving (Figure 3). The scheduling model has sensor tasks, system tasks and actuator tasks as input. Sensor tasks are the tasks received from the sensor with sensing data readings, system tasks are the tasks in the inference engine to generate action commands in response, and actuator tasks are the response tasks which are generated at the inference engine in order to control the smart car. The tasks are either periodic tasks or event driven tasks. In our scheduling model design with efficient task execution for smart cars, we aim to add three scheduling algorithms: fair emergency first, fixed priority preemptive and earliest deadline first scheduling.

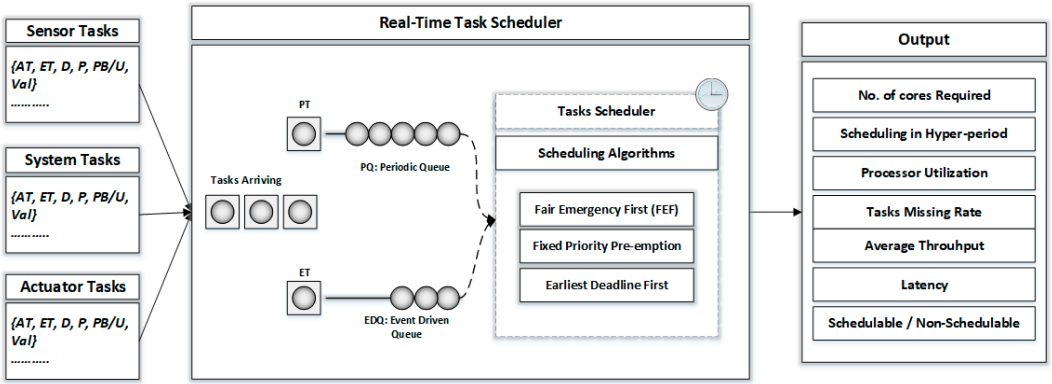

Figure 3. Scheduling model.

3.2.1. Fair Emergency First Scheduling Policy

We have designed a customized scheduling algorithm for our rule based smart control inference system for smart cars. We have improvised the priority based scheduling algorithm with the basic design goal to run high priority tasks (emergency tasks) first, and additional design goals being to minimize the task starvation rate and fair allocation of CPU resources.

The tasks in fair emergency first scheduling policy are classified into four categories: high urgency event driven tasks, normal event driven tasks, high priority periodic tasks and normal periodic tasks. A default high priority is given to the event driven tasks over the periodic tasks. Urgent event driven tasks have priority over normal event driven tasks and priority periodic tasks have priority over normal periodic tasks.

The designed priority policy aims to run the high priority and urgent tasks first while making sure that no other tasks are starved unnecessarily, consuming the CPU time as efficiently as possible in the given input scenarios. The task starvation is addressed by consuming the free CPU slots for starving tasks, and by pushing the execution of normal event driven tasks and periodic tasks to make room for possibly missed tasks without compromising the performance of the system.

3.2.2. Priority Based Scheduling Policy

Priority based scheduling is a scheduling system commonly used in real-time systems. With a priority associated with all given task, the scheduler ensures that at any given time, the processor executes the highest priority task of all those tasks that are currently ready to execute. Since priority is given to higher-priority tasks, the lower-priority tasks could wait an indefinite amount of time before being implemented, leading to a higher starvation rate [27].

3.2.3. Earliest Deadline First Scheduling Policy

Earliest deadline first (EDF) or least time to go is a dynamic priority scheduling algorithm used in real-time operating systems to place processes in a priority queue. Whenever a scheduling event

occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline. This process is the next one to be scheduled for execution [28].

3.3. Inference Engine Model for Safe Driving

In this sub-section, we present an inference model for safe driving in a smart car. The inference engine model has the inference engine and the control unit.

The inference engine is the component where rules are defined for the autonomous control of smart cars. The inference engine is where all the safe driving rules are listed along with the thresholds and if-else conditionings. A rule is based on three elements: arriving task type, condition associated with the task and the contextual scenario of the task (Figure 4). In the task type we have four task types: urgent event, normal event, priority periodic and normal periodic type tasks. Each condition associated with a task to fire rule will have threshold and value ranges to be met. Each scenario will have some associated dependencies and described relations.

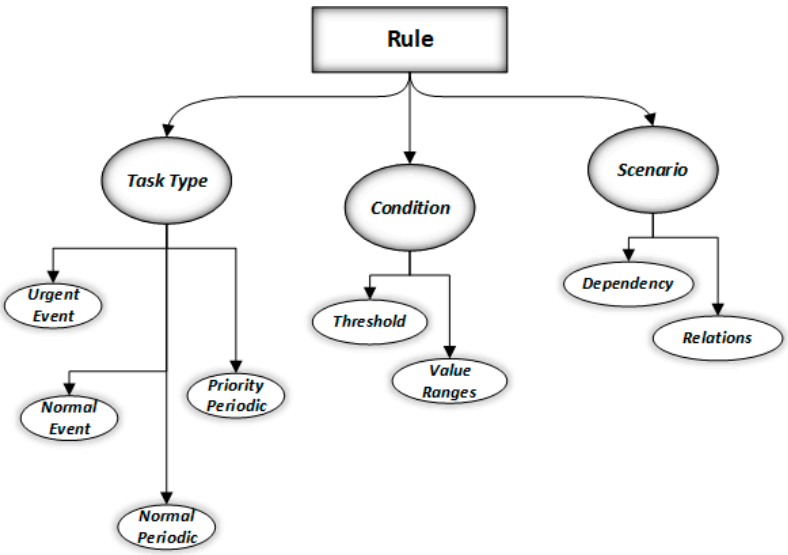

Figure 4. Defining of a rule in Inference Engine.

In case of an inference engine, contexts and scenarios are considered to follow rules and generate further tasks. The combination of different sensor readings makes a context for smart vehicle scenarios. In the case of an event, different contexts data can be combined in order to make more sense of the situation and scenario to make better control decisions for the smart cars.

A contextual scenario is based on contextual data and events are generated from the scenario context and the data available. For example from the context scenario such as the driver being detected will have input contextual data dependency on factors such as camera and the pressure sensor. In response, it will have an event control task "perform seat adjustment" generated to adjust the car's performance according to the driver's requirements. Similarly other context scenarios can be pedestrian detected, heavy rainfall, strong wind, speed jump detected etc. with each depending on different contextual data collection and generating different tasks in response of firing rules. The inference engine model also considers a set of exceptions for the decision making and rule firing process depending on the scenarios for better provision of safe driving for smart cars. Exceptions are listed and mapped when such scenarios occur where a special step needs to be taken (Figure 5).

In order to make safe driving inference engine system for smart cars, we have to design scenarios against each input sensing data values. In our case, we have six input environment sensors, one human data input sensor, and four smart car data input sensors. In environment sensors we have rainfall sensor, noise sensor, wind sensor, blurriness sensor and temperature sensor. In human data sensor, we have cameras. In smart car sensors, we have tire check, speed check, brake check and distance

from nearby cars as input sensing data. Table 2 below explains the environment sensing data scenario classifications, Table 3 explains the human sensing data scenario classifications and Table 4 explains the sensing data scenario.

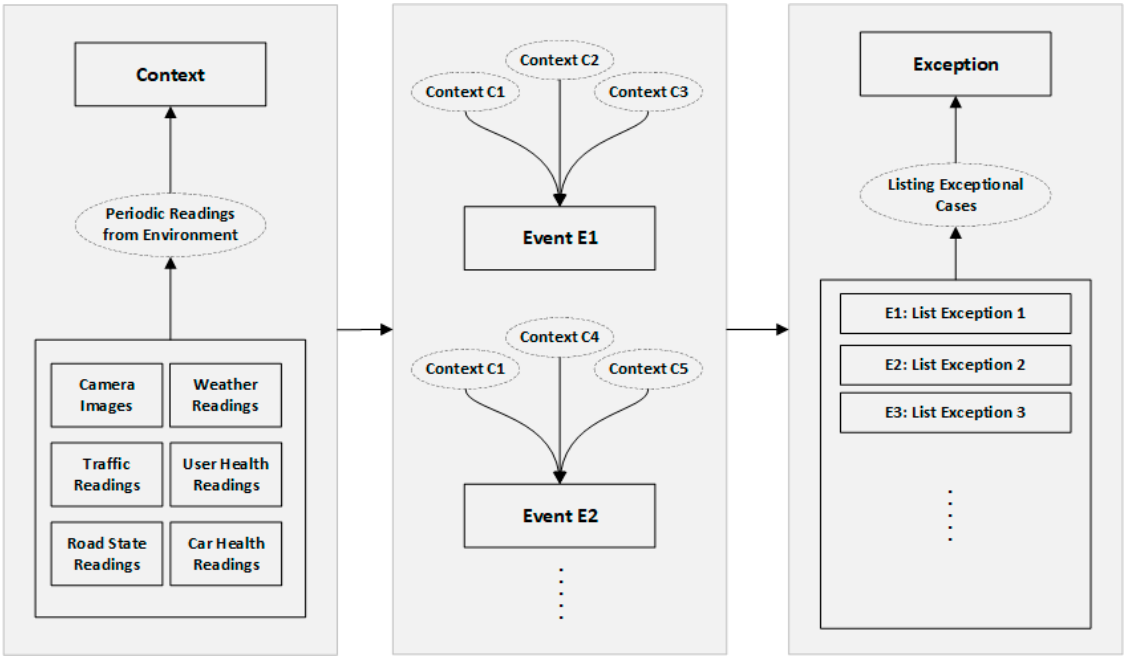

Figure 5. Event building from context in inference engine.

Table 2. Environment sensing data scenario classification for building rules.

Sensing Data	Scenarios
Rainfall	Light Showers
	Medium Showers
	Heavy Showers
Noise	Low Noise
	Medium Noise
	Loud Noise
Wind	No wind
	Light wind
	Medium wind
	Heavy wind
Blurriness	No Blurriness
	Light Blurriness
	Medium Blurriness
	Heavy Blurriness
Temperature	Low Temperature
	Medium Temperature
	High Temperature
Light	Low Light
	Medium Light
	Sharp Light

Table 3. Human sensing data scenario classification for building rules.

Sensing Data	Scenarios
Camera	Looking Right
	Looking Left
	Looking Straight

Table 4. Smart car sensing data scenario classification for building rules.

Sensing Data	Scenarios
Tire Check	Safe
	Warning
	High Alert
Speed Check	Safe
	Warning
	High Alert
Brake Check	Safe
	Warning
	High Alert
Car Distance	Safe
	Warning
	High Alert

In order to ensure safe driving conditions and environments, we define rules for each sensing value. The rules are built by first classifying the sensor value into some range, setting thresholds and determining an action opposite to each class range being true or false. As described in Tables 2–4 above, each sensing data has its own scenario classes corresponding to the sensor value readings. After classifying the incoming data based on threshold values, the next step for an inference engine is to determine the actions against the thresholds and fire rules accordingly (Figure 6).

The control unit has all the control tasks' logic for the smart vehicle's autonomous control. The actions to be performed in safe driving case fall into two types. First are the control functions to keep the car controlled in every scenario and maintain the safe driving environment. Second is to generate appropriate notifications and warnings for the user and system. Figure 7 below elaborates the control functions involved in the safe driving of a smart car. The smart car controller functionality includes window control, radio control, wipers control, lights control, distance control, brakes control, speed control, steering control, fan control and temperature control.

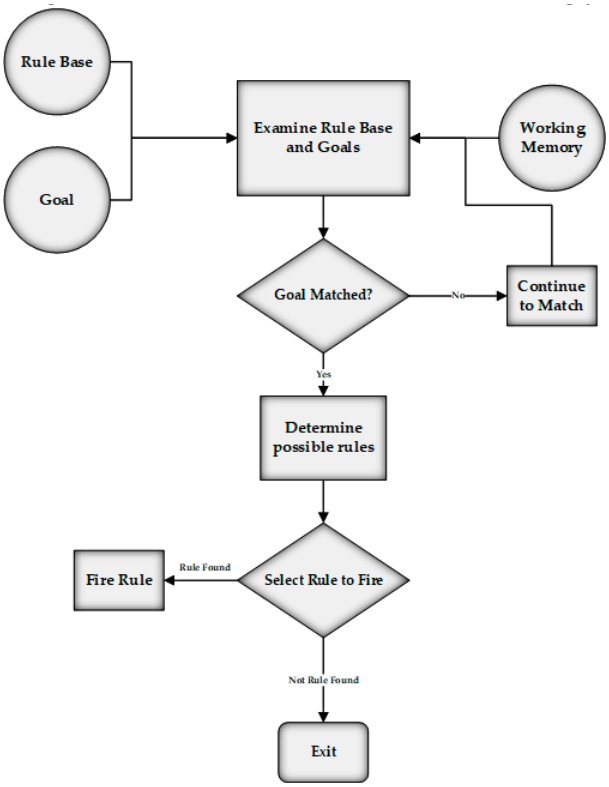

Figure 6. Workflow of firing a rule in inference engine.

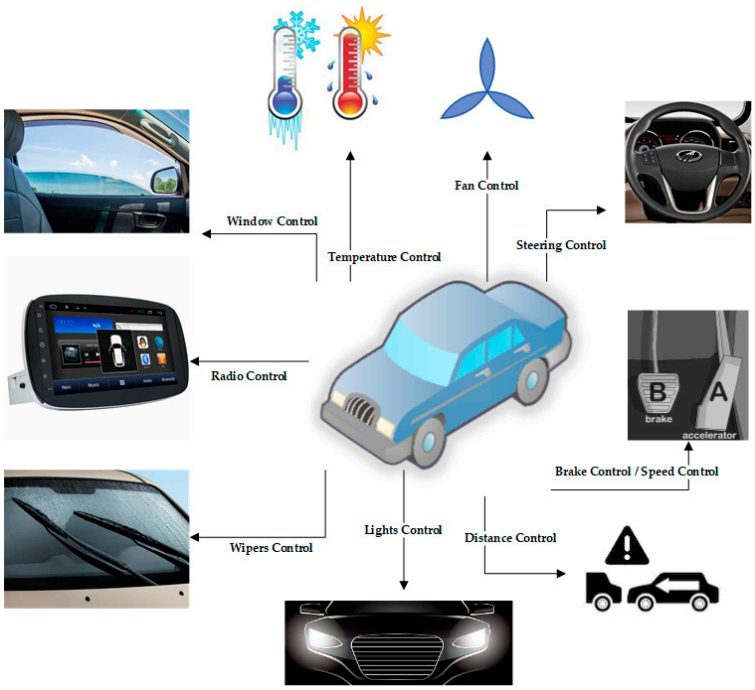

Figure 7. Smart car control operations for safe driving.

3.4. Hybrid Agent

The hybrid agent provides a flexible and robust interaction platform between the scheduler and the inference engine in order to have a more reliable system which executes the tasks of the overall system more efficiently with minimum resources being used.

Figure 8 shows the detailed system modules interaction with inclusion of the hybrid agent. The hybrid agent sends and receives the messages containing tasks data between scheduling module, inference engine module and the control module. It performs the configurations for the arriving tasks, depending on their types, scenario dependencies and priority tags. It parses the message in order to apply specific configurations and resource management actions. The hybrid agent maintains online information of the complete tasks flow among all system modules to manage the resources and keep the resources available for any high priority tasks.

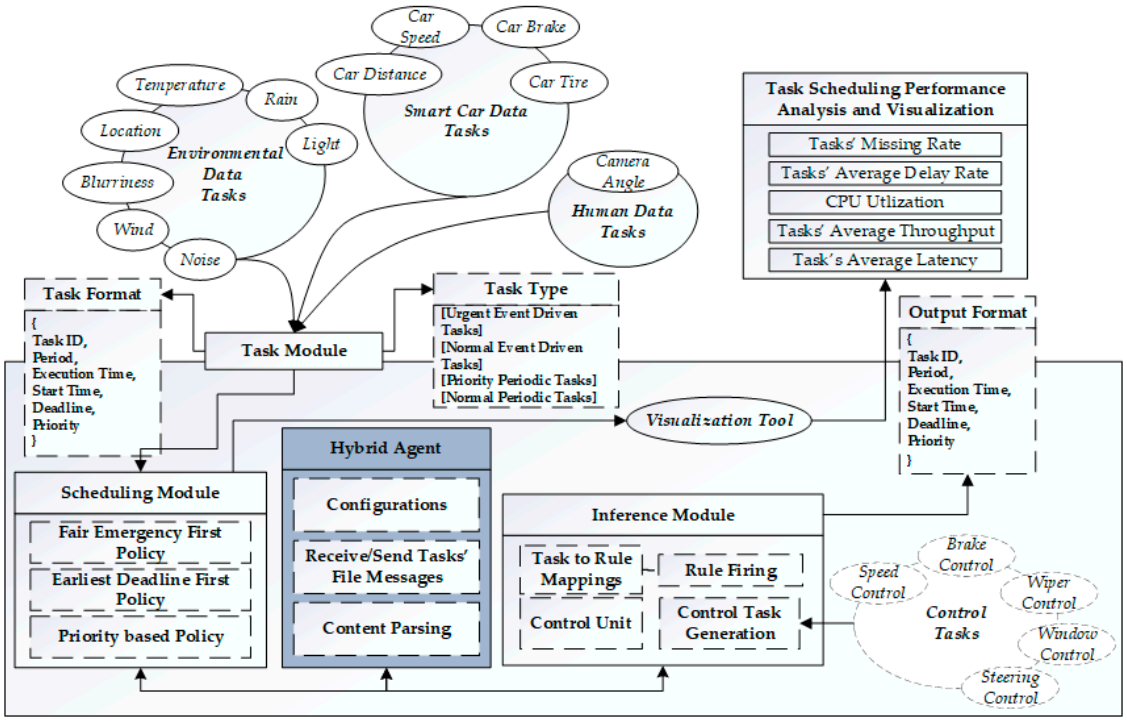

Figure 8. Hybrid agent's based scheduling and inference mechanism.

The sensing input data, after modeling into task format and task type, is passed onto the scheduler. At scheduler, upon receiving the list of tasks, the scheduling policy is applied to schedule the execution order of the tasks. For extracting the execution rules attached with each/any task, the tasks are forwarded to the inference engine via hybrid agent. The task ID mappings to match the rules are done. After mappings, the rule is extracted from the inference engine and sent to the control unit, where the control task opposite to the fired rule is generated. The control task is modeled along with its start time, execution time, deadline and other parameters and sent back to the scheduler via hybrid agent. The scheduler then executes the control task along with other running tasks based on its selected scheduling policy.

The hybrid agent has the vital role, as it bridges between the sub-modules in the system and also copes with the resource management based on the tasks load.

4. Simulation of Inference based Scheduling Mechanism for Safe Driving in Smart Cars

In this section we present our data set of smart cars and the simulation setup.

4.1. Task Data

We have used smart cars data collected after intervals of 0.1 s, 0.5 s, 1 s, 5 s and 10 s. The collected data is divided into three input task categories as environmental data, vehicular data and human data (Figure 9).

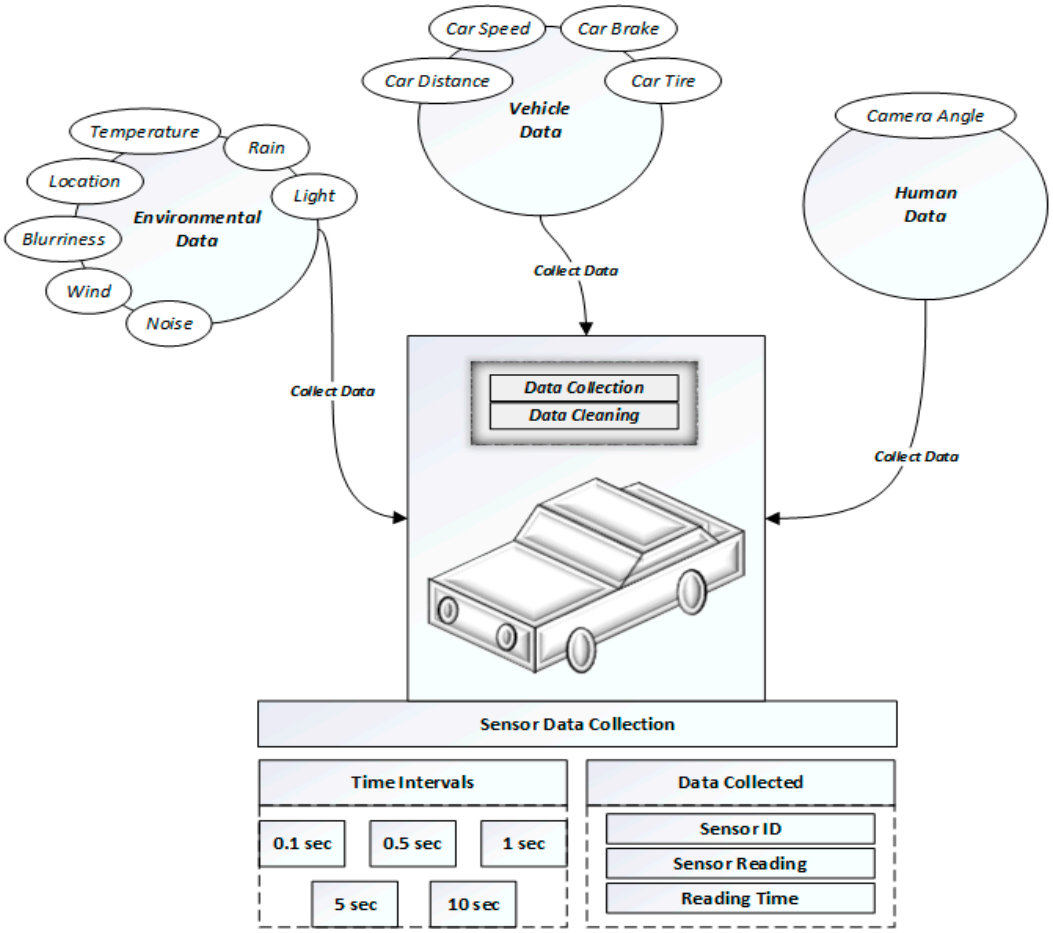

Figure 9. Tasks data.

The environment sensing data tasks have light, rain, temperature, location, blurriness, wind and noise sensing data. The human sensing data task has camera angle for the driver's head angle. The vehicular sensing data has car tire, car brake, and car speed and car distance sensing data. Data pre-processing is performed in order to remove any noise from the collected data and make the data understandable for the system.

4.2. Implementation Environment

We have used Python for implementing the core programming logic of the task scheduling algorithms. Python is a very popular general purpose programming language that is widely used for developing desktop based and web based applications. In order to develop our inference engine, we have used Drools libraries. Drools is a Business Rules Management System (BRMS) solution. It provides a core Business Rules Engine (BRE). The development environment for the system is shown in Table 5 below and in the simulation and testing phase; we have tested our built task scheduler on an embedded IoT testbed (Table 6). Raspberry-pi is used in simulation and testing scenarios as it best suited our system requirements.

Table 5. Development environment.

System Component	Value
Operating System	Windows (10.0.17134 Build 17134, Microsoft, Redmond, WA, USA)
CPU	Intel® Core™ i5-4570 CPU at 3.20 GHz (Santa Clara, CA, USA)
Primary Memory	8 GB
Platform	Eclipse Java Photon
Libraries	Drools
Programming Language (Scheduler)	Python 3

Table 6. Simulation and testing environment.

System Component	Value
Hardware	Raspberry Pi 3 Model B (Raspberry Pi Foundation, UK)
Operating System	Raspbian (November 2018, Raspberry Pi Foundation, UK)
Memory	1 GB
Server	Flask Webserver (Pallets, USA)
Libraries	GPIO (Version 0.6.5), CSVReader, Jinja Template (Pocoo), Bootstrap (Version 3), HTML 5/CSS3 (W3C)

4.3. Hybrid Scheduling and Inference Engine Model

In this sub-section we present the implementation of our hybrid approach of task scheduler and inference engine. Sensing input tasks as explained above in Section 4.1 are given input to the scheduler.

Initially a set of sensing input tasks arrive at the scheduling model, which has scheduling logic implemented at one end with different scheduling algorithms and hybrid agent interactions at the other end. The input tasks are periodic sensor readings, collected after a set interval of time. The sensing tasks are initially scheduled and forwarded to the hybrid agent accordingly; the hybrid agent then passes them onto the inference engine based on the priorities and levels of urgency. The process tasks are executed at the inference engine, resulting into generation of output tasks which are smart control tasks or the notifications/alerts for the smart car. Hybrid agent gets the output tasks and manages the execution depending on the existing resources and priority orders. Our inference engine is drools based, and a set of rules to be fired is defined in a drl file, in accordance with all input data scenarios and smart car's safety control functionality (Figure 10).

In Table 7 below, we consider an example scenario for smart car input data as rainfall, noise and temperature. The table explains the total number of tasks to be executed along with each task's name, its required CPU consumption demand and the priority of the task at the scheduler.

The simulation system in this example scenario will have rainfall sensing data, noise sensing data and temperature sensing data as periodic sensing data input tasks. The system task involved will be rule firing task, for each time a rule goal is matched and a rule is followed. The actuator tasks involved in the given scenario are window control task, wiper control task, radio control task, temperature control task, speed control task and steering control task.

The Figure 11 above shows the simulation execution flow from the input sensor readings to the scheduler. At the scheduler, the sensing tasks are executed based on priorities and sent to the hybrid agent for further processing. The hybrid agent based on the incoming tasks, sets the configurations and forwards the parsed content to the inference engine. The inference engine classifies the incoming data sensing values, matches the rules and follows the rules when the goal is met. The followed rules contain the response control tasks to be executed for the safe driving of a smart car. The fired control tasks are lined at control unit and communicated back to scheduler via hybrid agent and executed at control unit when scheduled to execute.

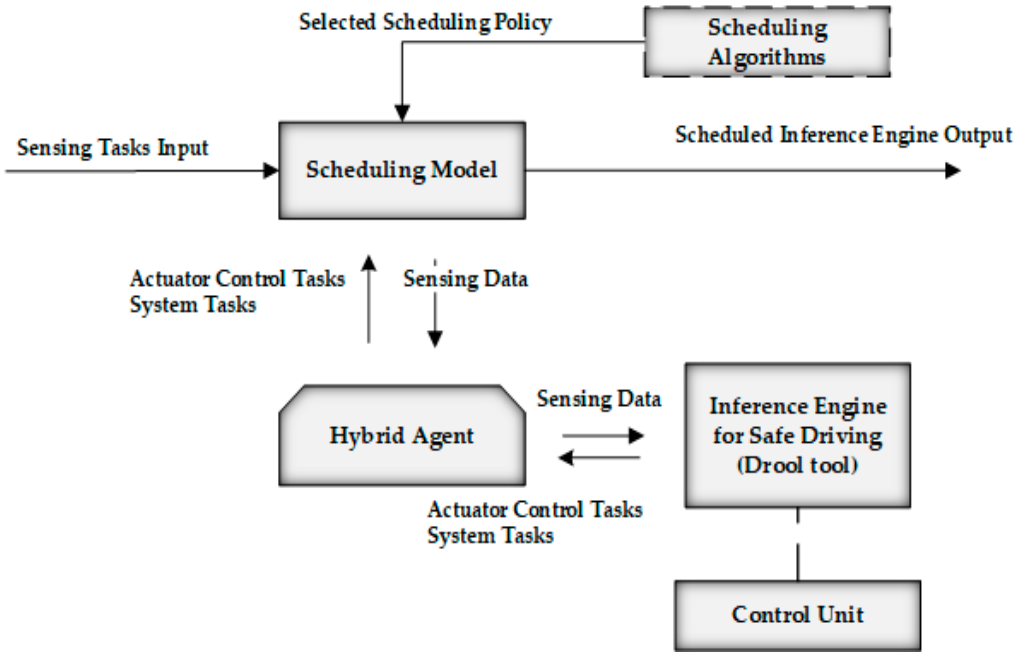

Figure 10. Implementation flow of scheduling and inference engine.

Table 7. Example derived scenario for rainfall, noise and temperature.

Task	CPU Required	Priority
Sensing Tasks		
Sensing Rainfall Data	20 ms	Normal Periodic Task
Sensing Noise Data	20 ms	Normal Periodic Task
Sensing Temperature Data	20 ms	Normal Periodic Task
System Tasks		
Fire Rule	300 ms	Priority Periodic Task
Actuator Tasks		
Window Control Task	520 ms	Normal Event Driven
Wiper Control Task	520 ms	Urgent Event Driven
Radio Control Task	520 ms	Normal Event Driven
Temp Control Task	520 ms	Normal Event Driven
Speed Control Task	520 ms	Urgent Event Driven
Steering Control Task	520 ms	Urgent Event Driven

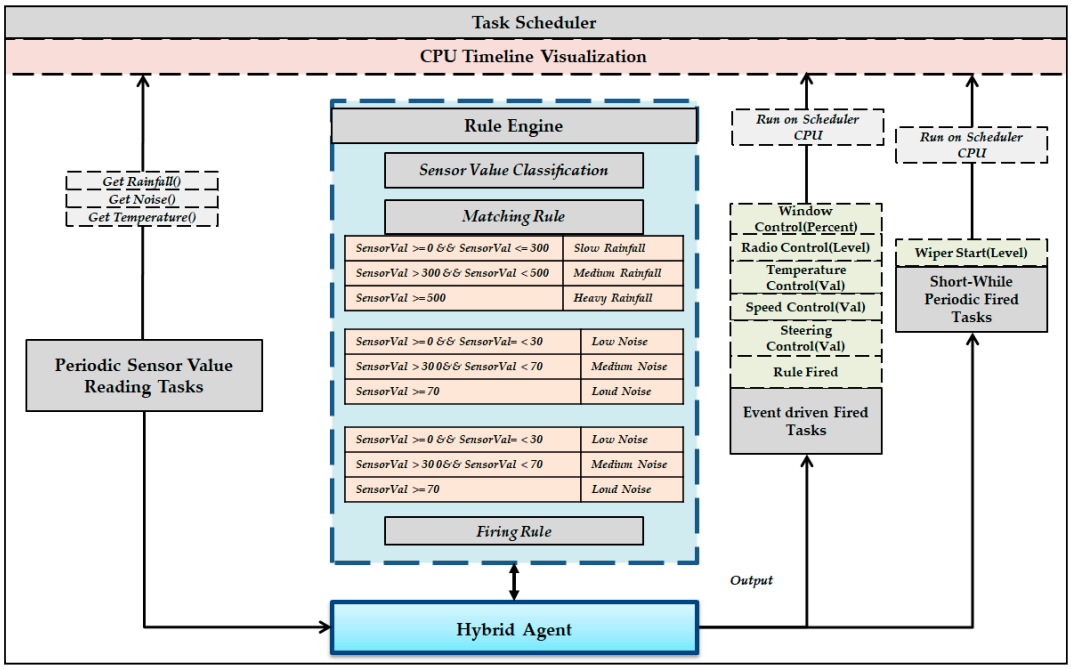

Figure 11. Simulation execution of the task scheduling and inference module for the example scenario.

5. Simulation Visualization

We have developed a web-based task simulation visualization tool using flask, which is an MVC based framework. As part of visualization we implemented FEF, Priority based and EDF scheduling policies for all different scenarios and presented several interfaces in a very neat and clear way. Similarly to all other algorithms, we have certain tabs like General Summary, Tasks Timeline, Performance Visualization, Overall Metrics Visualization and CPU Timeline.

The Figure 12 below shows the CPU timeline of scheduled tasks. Various color codes represent various categories of the tasks. For instance, Task 1 with pink color is normal periodic task which is received at time 0. At this particular moment, no other high priority task has arrived yet, so it gets CPU eventually. For next couple of cycles the CPU is free, which is indicated by dark green color. Clock cycle 3 to 9 is assigned to event-driven task 1.1 and this goes on till the hyper period amount of time. After this the same pattern gets executed.

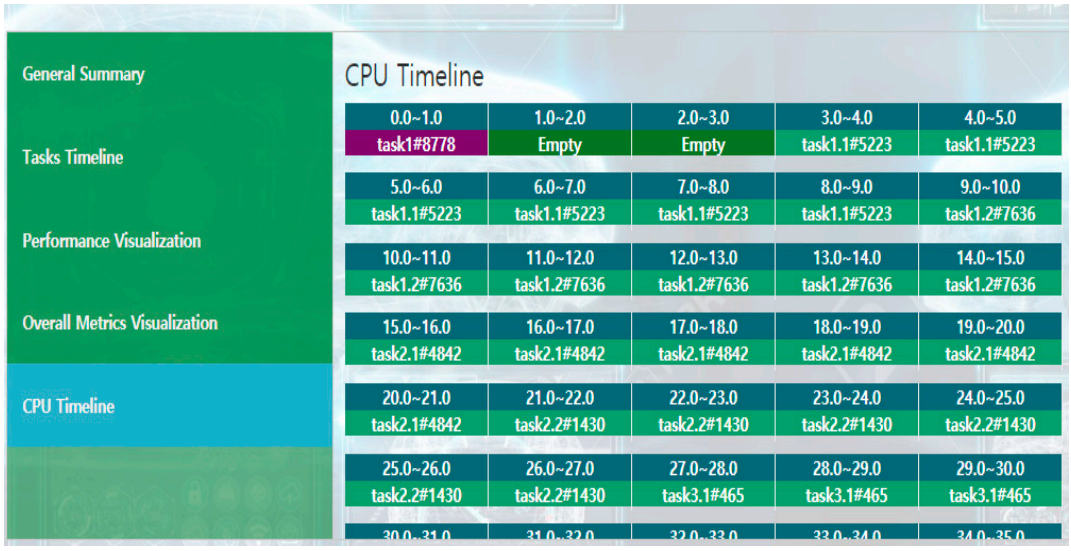

Figure 12. CPU timeline visualization.

6. Performance Analysis

In this section we present a performance analysis of our proposed rule based scheduling for smart cars autonomous control. In Section 6.1, we perform comparisons analysis of our proposed system with non-hybrid rule engine and basic rule engine. In Section 6.2, we perform the implemented scheduling schemes comparisons at the proposed system.

For the purpose of performance evaluations, we have considered 9 scenarios compromising of different input data combinations. The Table 8 below shows the list of our considered input data combinations' scenarios. Each scenario is a combination of many sub-tasks based on the scenario dependencies.

Table 8. Input data scenarios.

Scenarios	Input Data Combination
Scenario 1	Rainfall + Noise + Temperature
Scenario 2	Blurriness + Light + Rainfall
Scenario 3	Camera Angle + Speed Check + Brake Check
Scenario 4	Light + Blurriness + Wind
Scenario 5	Location + Rainfall
Scenario 6	Noise + Wind + Rainfall
Scenario 7	Rainfall +Tire Check + Brake Check
Scenario 8	Blurriness + Speed Check + Distance
Scenario 9	Camera Angle + Speed Check + Distance

6.1. Comparison Analysis between Hybrid, Non-Hybrid and Basic Rule Engine Approaches

In this sub-section, we present the performance analysis comparisons for the proposed hybrid system of inference engine based scheduling mechanism with a non-hybrid system and a basic rule engine. We have simulated the non-hybrid approach with the same environment constraints as our proposed hybrid approach. The non-hybrid system does not have the implementation features of the hybrid agent, which in a hybrid approach system are better resource management and a robust execution of tasks execution. We have also simulated a basic rule engine with FIFO scheduling and priority rule firing for detailed comparison purposes. In the comparisons and graphs, for the sake of ease, we will refer to our proposed inference engine based scheduling mechanism as a hybrid system.

6.1.1. CPU Time Utilization

CPU time utilization (or process time) is the amount of time for which a central processing unit (CPU) was used for processing the tasks on its core/cores. In this context, we refer to CPU time Utilization as usage of CPU time slots in the most efficient manner; high CPU utilization translates into more tasks being handled and less CPU slots being wasted.

Figure 13 above shows the average CPU time utilization for the smart car's tasks set at y-axis with varying sampling intervals on x-axis. The proposed hybrid system performs better than the non-hybrid system and basic inference engine as the hybrid system target to utilize the free CPU slots by priority shifting in order to increase throughput. The non-hybrid system though lacks the efficient resource allocation and thus results in more free slots, which would eventually result in a high tasks' missing rate. The basic inference engine has the minimum CPU slots consumption as it follows the FIFO scheme with single queue and basic priority scheme in comparison to the non-hybrid system which has a fully featured multi-priority scheduling end implemented.

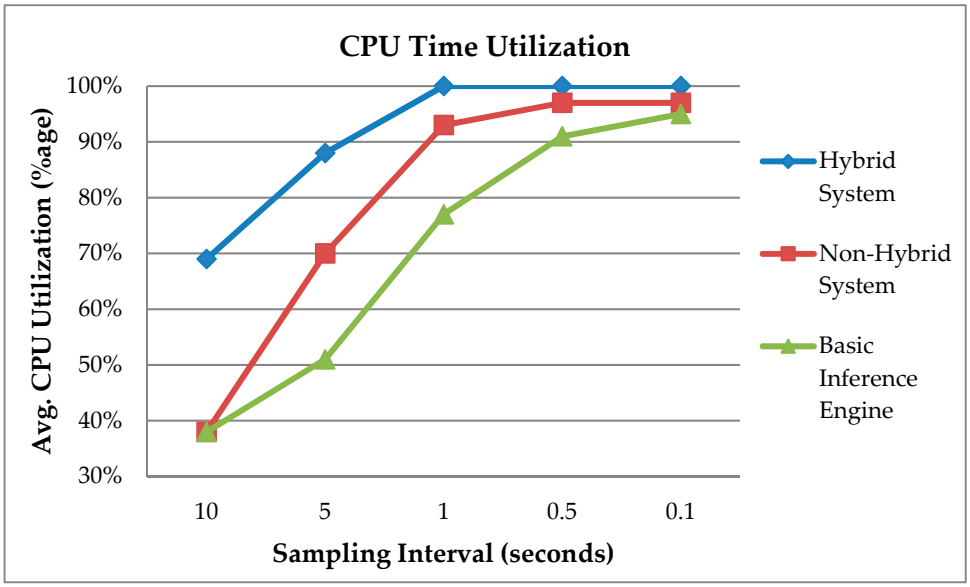

Figure 13. Average CPU time utilization.

6.1.2. Missing Rate (Percentage)

Missing rate is the percentage of tasks which missed their deadline and are executed after their deadline. As in Section 6.1.1, we have observed that more free CPU slots are utilized by a hybrid system in comparison to a non-hybrid system and basic inference engine system. An efficient and optimized usage of free CPU slots would definitely result into a higher throughput and lower missing rate for the tasks. Hence the average missing rate is lowest for the hybrid system, followed by the non-hybrid system and basic inference engine having the maximum tasks' missing rate. The missing rate for the tasks increases with the decrease in the sampling interval, as the tasks load on the system increase in smaller sampling intervals. In the case of a hybrid approach, the tasks' missing rate is 0% for sampling interval of 10 and 5 but it increases to around 7%, 13% and 19% for the sampling intervals 1, 0.5 and 0.1 s respectively (Figure 14).

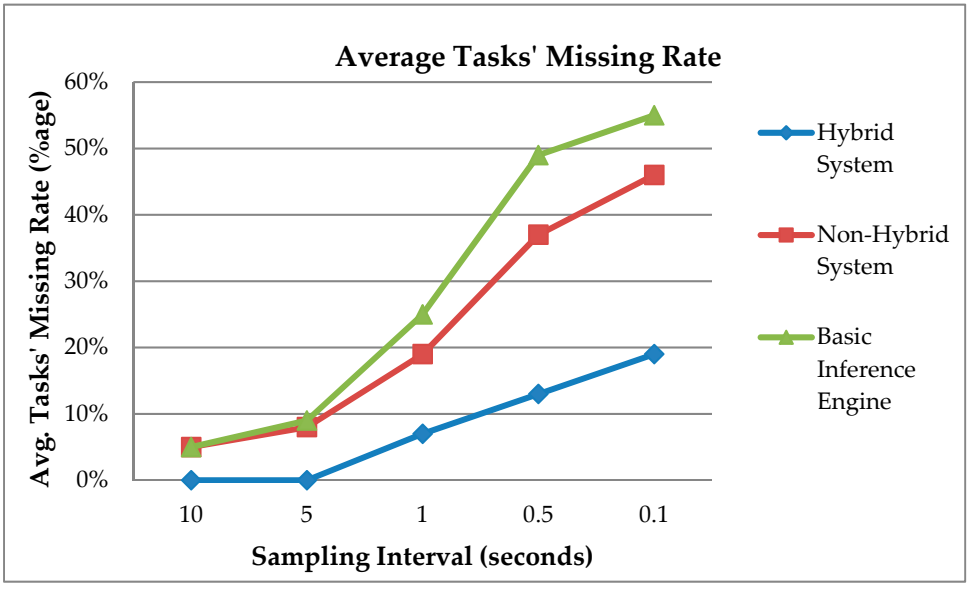

Figure 14. Average tasks missing rate.

An essential concern in the Figure 14 above is whether the inevitable missing tasks with the high task load are of a critical nature or a non-critical nature. In the scenarios where the total tasks load becomes higher than the total available capacity, the hybrid agent in the proposed system manages an efficient trade-off between the critical tasks and non-critical tasks. The Figure 15 below shows the missing tasks rate for varying load in a basic inference engine system while Figure 16 shows the missing tasks rate for varying load in hybrid system. By comparing Figures 15 and 16, we can conclude that the non-hybrid system might miss a higher number of critical tasks as compared to the hybrid system. The combination of hybrid agent and FEF scheduling mechanism modifications allows the hybrid system to reduce the tasks starvation rate. In this way, the only critical tasks missing at the hybrid system would be those which fall into the lower priority at the ongoing CPU time slots and running those would mean causing bigger damage to the system eventually.

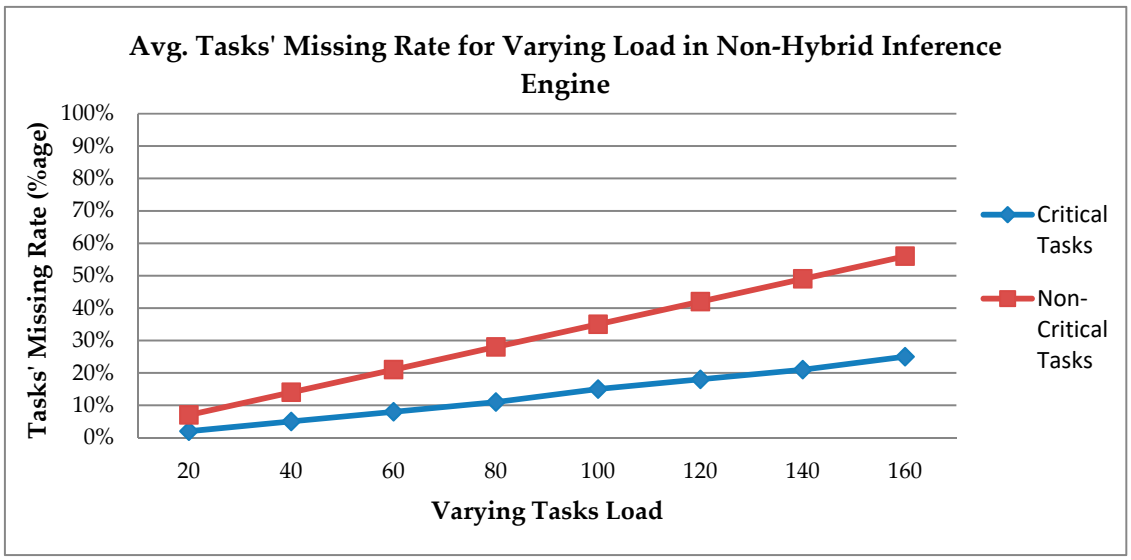

Figure 15. Non-hybrid inference engine.

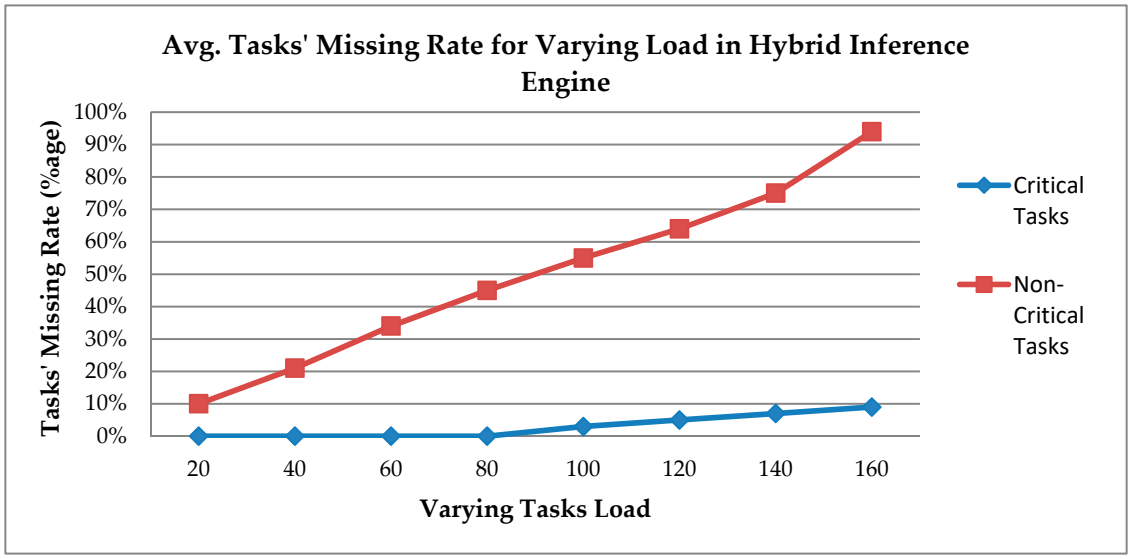

Figure 16. Hybrid inference engine.

6.1.3. Response Time

The response time is the time taken from the release of a task to its first execution. In this context we refer to the response time as the first sub-task execution return of a larger task scenario (Table 7).

In the Figure 17 below, average response time for varying sampling intervals is shown. The non-hybrid system has a slightly better response time than the basic inference engine system whereas the hybrid approach significantly improves the response time. The main difference between the hybrid and non-hybrid systems is the implementation layer for the hybrid agent with better resource allocation, fast tasks processing, minimizing delay and avoiding unnecessary system tasks.

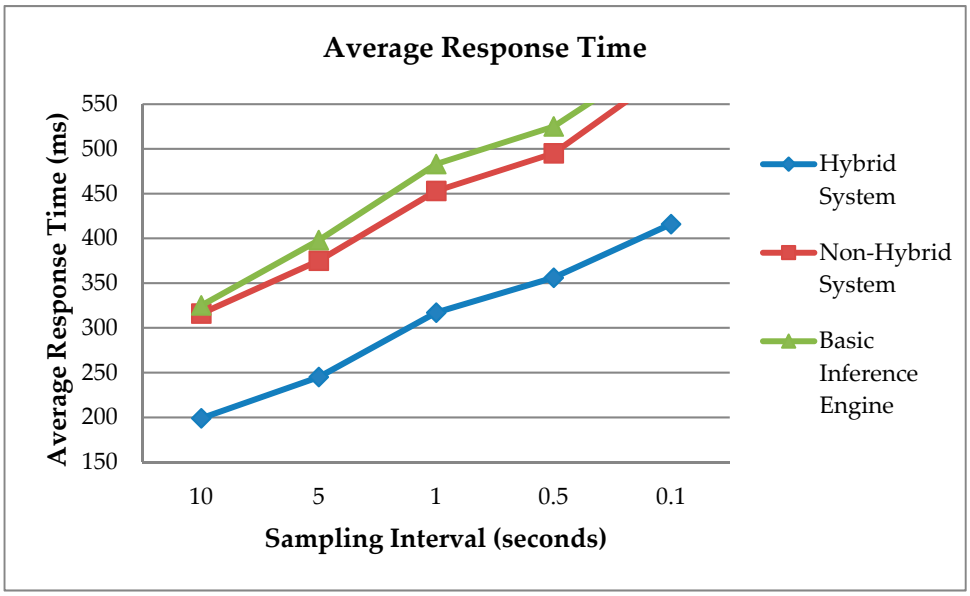

Figure 17. Average response time.

6.2. Comparisons between FEF, EDF and Priority based Scheduling Algorithms

In this sub-section we provide the comparisons between the implementation of hybrid system with three different scheduling policies: FEF (Fair Emergency First) scheduling algorithm, EDF (Earliest Deadline First) scheduling algorithm and Priority based scheduling algorithm.

In EDF the task scheduler directly executes the task with nearest deadline first while in the Priority based scheduling algorithm, the task scheduler executes the tasks based on their priority (i.e. highest priority first). The FEF scheduling algorithm takes into account the system and user priorities and also other factors such as current missing rate and delay rates along with the total tasks load on the system. Hence the hybrid system with FEF scheduler implementation shows more promising results in comparison to the priority based on EDF based implementation.

Figure 18 above shows the response time comparison of the three scheduling algorithms and their implementation of the proposed hybrid system for the nine scenarios tasks set. The figure clearly indicates that hybrid FEF implementation outperforms the priority based and EDF based implementations.

Now, we consider other crucial scenarios in the real-time scheduling, e.g., over flow of tasks at the scheduler, low-priority tasks not being able to get executed due to high flow of high priority tasks and large number of unexpected interrupts. Our tailored FEF algorithm makes best use of its resources and capacity to deal with such scenarios. The urgency and PB factors, introduced in task modeling (Section 3.1), help the scheduler to take the best scheduling decision. The PB factor is ideally introduced to invert the priority to the low priority tasks when their executed can no longer be delayed. Figure 19 below elaborates a scenario where the real-time system is overflown with its tasks load, and a low priority task has to wait forever for its turn. In such cases the PB for a low priority task is set to 1 if it's obligatory to run the low priority task after a definite time period, otherwise it might cause a superior loss to the system.

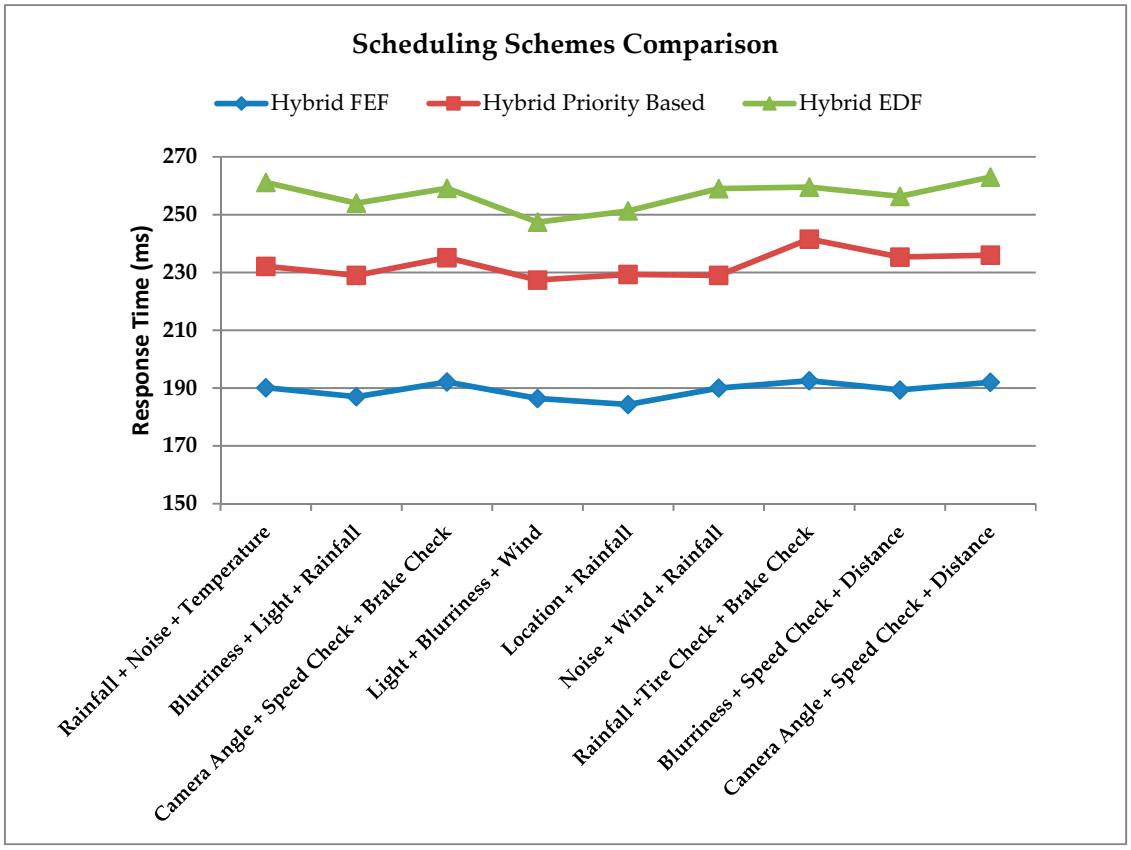

Figure 18. Response time.

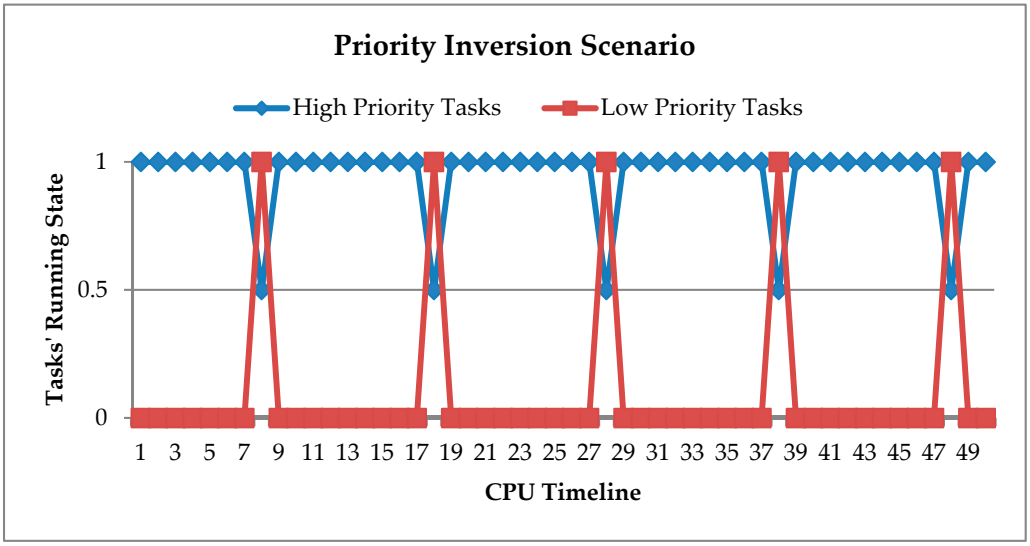

Figure 19. Increasing tasks' load and priority inversion scenario.

7. Discussions

In this work, we have studied the importance of safe driving in smart cars and the importance of a robust inference engine for smart cars' control. Owing to advances in transportation industry and IoT, smart cars have become intelligent enough to deliver numerous expedient functions for drivers. With convenience, there comes complexity as a side effect; smart cars are embedded with smart modules performing complex tasks. These complex functions have led to an increase in the complexity of controls. For the smart cars to be more accurate and situation aware, it requires a task scheduler

for the controller. Previously, smart cars were using a separate task scheduler which prioritizes the tasks for the controller; and brings an additional level of complexity in the system. In order for the smart cars to be more efficient, safe and lightweight, complexity needs to be reduced. Thus in this paper, we introduced a hybrid module which combines the task scheduler with the inference engine via hybrid agent. The hybrid agent aims to facilitate the real-time task management and the resource management for provision of an efficient inference engine for smart cars. An improvised version of a priority based scheduler is introduced to be installed at task scheduler level named as an FEF scheduler. For the purpose of performance analysis, a non-hybrid version of the system with priority scheduling is simulated and a basic inference engine with priority rule firing and FIFO scheduler is also implemented. The comparisons are made between the three implementation based on performance metrics of CPU Usage, Tasks' Missing Rate and Response Time. At the scheduler level for the proposed hybrid system, three scheduling techniques as FEF, Priority based and EDF are implemented to draw the performance comparisons between them. All three scheduling techniques (FEF, Fixed priority and EDF) come under the definition of preemptive scheduling in OSEK/VDX task scheduling. In OSEK/VDX preemptive scheduling, the running task might be rescheduled at the occurrence of any pre-set event or condition [29]. In FEF, pre-set conditions are based on task urgencies and PB conditions. In EDF the pre-set condition is based on another task arriving with an earlier deadline than the currently running task and in fixed priority the pre-set condition would be if another task arrives with a higher priority than the currently running task.

It can be clearly observed from the results that our proposed hybrid approach outperformed the non-hybrid and basic inference engine systems. We observed a significant reduction of 25% to 75% in the number of task instances missed and the number of high priority tasks missed. As the combination of better resource management at hybrid agent and improvised FEF scheduler results in a significant reduction in the task starvation rate and maximum CPU time slots utilization; as low priority tasks are saved from unnecessary starvation and CPU unit times are used wisely. To the best of our knowledge, our proposed hybrid system is first of its kind. Hence, our proposed hybrid inference based scheduling mechanism is recommended for smart cars in safe driving environments; where constraints and requirements vary depending on the scenarios. This is because it is flexible, adaptive and allows setting system constraints based on independent priorities when the input is of a diverse nature.

Author Contributions: Data curation, S.M.; Formal analysis, S.M.; Funding acquisition, D.K.; Investigation, S.A.; Methodology, S.M.; Resources, D.K.; Software, S.A.; Supervision, D.K.; Validation, S.M.; Visualization, S.M. and S.A.; Project management, B.W.K. and D.H.P.; Writing—original draft, S.M.; Writing—review & editing, S.M. and D.K.

Funding: This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) and ITRC (Information Technology Research Center) support program supervised by the IITP.

Acknowledgments: This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00526, Development of Intelligent IoT System capable of cooperation and learning between things in movement-free environment), and this research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00313) supervised by the IITP (Institute for Information & communications Technology Promotion). Any correspondence related to this paper should be addressed to DoHyeun Kim.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Car Accident Statistics 2013. Available online: https://www.fishertalwar.com/car-accident-statistics/ (accessed on 20 December 2018).
2. Griffin, N.L.; Lewis, F.D. A rule-based inference engine which is optimal and VLSI implementable. Tools for Artificial Intelligence. In Proceedings of the IEEE International Workshop on Architectures, Languages and Algorithms, Fairfax, VA, USA, 23–25 October 1989.

3. Axel, N.; Stengel, R.F. An expert system for automated highway driving. *IEEE Control Syst.* **1991**, *11*, 53–61.
4. Wang, F.-Y.; Zeng, D.; Yang, L. Smart cars on smart roads: an IEEE intelligent transportation systems society update. *IEEE Pervasive Comput.* **2006**, *4*, 68–69. [CrossRef]
5. Mostafa, S.A.; Mustapha, A.; Hazeem, A.A.; Khaleefah, S.H.; Mohammed, M.A. An Agent-Based Inference Engine for Efficient and Reliable Automated Car Failure Diagnosis Assistance. *IEEE Access* **2018**, *6*, 8322–8331. [CrossRef]
6. Hoch, S.; Althoff, F.; Rigoll, G. *The Connected Drive Context Server-Flexible Software Architecture for a Context Aware Vehicle. Advanced Microsystems for Automotive Applications*; Springer: Berlin/Heidelberg, Germany, 2007.
7. Benz. Mercedes-Benz to Present Driver Assistance Systems of Tomorrow at the ITS World Congress 2008. 2007. (accessed on 20 December 2018).
8. Volvo. Volvo Cars Focuses on Preventive Safety. 2007. Available online: http://www.volvocars.com/intl/corporation/NewsEvents/News/Pages/default.aspx?item=33 (accessed on 20 December 2018).
9. Lexus. Lexus Outlines Advanced Active Safety Technologies for All-New LS. 2007. Available online: https://newsroom.lexus.eu/lexus-outlines-advanced-active-safety-technologies-for-all-new-ls/ (accessed on 20 December 2018).
10. Sun, J.; Wu, Z.; Pan, G. Context-aware smart car: from model to prototype. *J. Zhejiang Univ.-Sci. A* **2009**, *10*, 1049–1059. [CrossRef]
11. Lee, J.D.; Young, K.L.; Regan, M.A. Defining driver distraction. In *Driver Distraction: Theory, Effects, and Mitigation*; CRC: Boca Raton, FL, USA, 2008; p. 31.
12. Koesdwiady, A.; Soua, R.; Karray, F.; Kamel, M.S. Recent trends in driver safety monitoring systems: State of the art and challenges. *IEEE Trans. Veh. Technol.* **2017**, *66*, 4550–4563. [CrossRef]
13. Jo, J.; Lee, S.J.; Jung, H.G.; Park, K.R.; Kim, J. Vision-based method for detecting driver drowsiness and distraction in driver monitoring system. *Opt. Eng.* **2011**, *50*, 1272022-1–1272022-4. [CrossRef]
14. Wollmer, M.; Blaschke, C.; Schindl, T.; Schuller, B.; Farber, B.; Mayer, S.; Trefflich, B. Online driver distraction detection using long short-term memory. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 574–582. [CrossRef]
15. Hirayama, T.; Mase, K.; Takeda, K. Detection of driver distraction based on temporal relationship between eye-gaze and peripheral vehicle behavior. In Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, AK, USA, 16–19 September 2012; pp. 870–875.
16. Gallahan, S.L.; Golzar, G.F.; Jain, A.P.; Samay, A.E.; Trerotola, T.J.; Weisskopf, J.G.; Lau, N. Detecting and mitigating driver distraction with motion capture technology: Distracted driving warning system. In Proceedings of the 2013 IEEE Systems and Information Engineering Design Symposium, Charlottesville, VA, USA, 26 April 2013; pp. 76–81.
17. Craye, C. A Framework for Context-Aware Driver Status Assessment Systems. Master's Thesis, University of Waterloo, Waterloo, ON, Canada, 2013.
18. Tango, F.; Botta, M. Real-time detection system of driver distraction using machine learning. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 894–905. [CrossRef]
19. Liang, Y.; Lee, J.D. A hybrid Bayesian network approach to detect driver cognitive distraction. *Transp. Res. C Emerg. Technol.* **2014**, *38*, 146–155. [CrossRef]
20. Ragab, A.; Craye, C.; Kamel, M.S.; Karray, F.A.; Craye, C.; Kamel, M.S.; Karray, F. A visual-based driver distraction recognition and detection using random forest. In *Image Analysis and Recognition*; Springer: New York, NY, USA, 2014; pp. 256–265.
21. Craye, A.; Rashwan, M.; Kamel, S.; Karray, F. A multi-modal driver fatigue and distraction assessment system. *Int. J. Intell. Transp. Syst. Res.* **2016**, *14*, 173–194. [CrossRef]
22. Bradley, J.; Atkins, E. Optimization and control of cyber-physical vehicle systems. *Sensors* **2015**, *15*, 23020–23049. [CrossRef] [PubMed]
23. Deluka, T.A.; Giuffrè, T.; Surdonja, S.; Trubia, S. Introduction of Autonomous Vehicles: Roundabouts design and safety performance evaluation. *Sustainability* **2018**, *10*, 1060. [CrossRef]
24. Borraz, R.; Navarro, P.; Fernández, C.; Alcover, P. Cloud Incubator Car: A Reliable Platform for Autonomous Driving. *Appl. Sci.* **2018**, *8*, 303. [CrossRef]
25. Zhang, Y.; Chen, H.; Waslander, S.; Yang, T.; Zhang, S.; Xiong, G.; Liu, K. Toward a more complete, flexible, and safer speed planning for autonomous driving via convex optimization. *Sensors* **2018**, *18*, 2185. [CrossRef]
26. ISO 26262. Available online: https://en.wikipedia.org/wiki/ISO_26262 (accessed on 14 March 2019).

27. Fixed-priority_Pre-emptive_Scheduling. Available online: https://en.wikipedia.org/wiki/Fixed-priority_pre-emptive_scheduling (accessed on 20 December 2018).
28. Earliest_Deadline_First_Scheduling. Available online: https://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling (accessed on 20 December 2018).
29. OSEK/VDX Operating System. Available online: http://www.irisa.fr/alf/downloads/puaut/TPNXT/images/os223.pdf (accessed on 14 March 2019).