AGENT BASED SIMULATION OF THE FLEXRAY PROTOCOL: A CASE STUDY OF THE CLOCK SYNCHRONIZATION AND MEDIA ACCESS CONTROL SERVICES

Sofiane Zaidi and Fateh Boutekkouk

Department of Mathematics and Computer Science, University of Oum El Bouaghi, Algeria
{sofiane_zaidi, fateh_boutekkouk}@yahoo.fr

ABSTRACT

This paper deals with agent based design and simulation of the FlexRay protocol for automotive distributed embedded systems. FlexRay supports both time driven and event driven communications. The proposed architecture is modeled as a multi-agent system and implemented in the JADE platform following mainly the so-called O-MaSE methodology. Using JADE, We developed an ontology that provides an automatic interpretation of frame fields. Furthermore, this paper provides a case study of the clock synchronization service and the control of access to media service in the static and dynamic segments of FlexRay.

1. INTRODUCTION

The increasing of safety in real time control applications in the automotive field led to the creation of the FlexRay Protocol [1]. FlexRay is a framework to develop distributed real time applications. It provides a logical bus connecting the host computers that implement the chosen application, and a set of services to guarantee high data rate, flexible and fault-tolerant communication. Each controller host computer and its communication controller are attached to the system through a FlexRay driver part; the combination of a controller and its driver parts is called a node. Nodes communicate over replicated shared media, called channels. The FlexRay protocol allows both time triggered and event triggered communication to guarantee a flexible and deterministic communication. Generally, FlexRay includes a set of basic services such as the wakeup, startup, clock synchronization, messages sending in the static and dynamic segments, reception and errors detection and correction. In this paper, we investigate the idea of using the agent paradigm to design and simulate the FlexRay services. Since there is a big similarity between multi-agent systems and distributed embedded systems, it seems very appropriate to model a distributed embedded system as a multi-agent system. The impetus behind this is to exploit the agent paradigm capabilities especially the complexity management, the intelligence, the planning, and the adaptability aspects. In order to accomplish our objective, we followed the O-MaSE methodology to design the multi-agent FlexRay system since it provides many models that cover many aspects such as organizational, functional, structural, and behavioral aspects. Of course we integrated the ontology aspect which is present in the O-MaSE methodology by the domain model. We used the JADE platform to implement and simulate our proposed system. In this paper, we have simulated the behavior of FlexRay nodes during the static and dynamic segments of the FlexRay protocol.

The paper is organized as follows: section two reviews briefly the related work. Section three puts the light on the FlexRay protocol. Sections four and five present the FlexRay clock synchronization and the media access control services respectively. Section six presents the conceptual models of the basic services using the O-MaSE methodology. Section seven presents our proposed architecture. Section eight discusses our implementation using JADE platform. Section nine presents a case study of the clock synchronization service and media access control services of the FlexRay protocol during the static and dynamic segments before the conclusion.

2. RELATED WORKS

In this section, we briefly present some pertinent works on FlexRay modeling and simulation. The work in [2] proposed a method for developing and verification of a FlexRay communication controller based on SystemC from analyzed FlexRay SDL descriptions which represent the core mechanism of the communication controller. This work is about the startup service. The author in [3] developed a FlexRay system model which is based on communication cycles with both static and dynamic segments using Matlab to evaluate the FlexRay network utilization. The result of simulation shows that the FlexRay network utilization is affected by various factors such as the length of message frames and the number of static slots and minislots. The author in [4] proposed a systemC simulation of dynamic segment of the FlexRay protocol. The simulator estimates messages delay which

is due to the interferences of higher priority messages. This work limited the values of message properties to certain range; for instance the length of the message is between the ranges of 2-15 minislots. The work in [5] proposed a based timing analysis simulation for FlexRay communication at the system level. The proposed simulation includes the communication within the static and dynamic segment, the calculation and application of offset and rate correction values of the synchronization process. According to the literature, we can state that there is a lack of works targeting the simulation of the FlexRay services using the agent paradigm although FlexRay is a complex protocol. For this reason, we try in this work to show how we can exploit efficiently the agent paradigm for FlexRay basic services modeling and simulation.

3. FLEXRAY ARCHITECTURE

FlexRay is composed of one or more clusters, each cluster is composed of a set of interconnected nodes via a replicate shared media, each node is an autonomous entity that executes a part of the distributed application and exchanges messages with others nodes. The communication between clusters is realized by the gateway. Each node generally has two parts: the controller part and the driver part [6]. The controller part consists of the host computer to run the application and a communication controller which is responsible for implementing the protocol aspects and services of the FlexRay communication system like: wakeup service, startup service, clock synchronization service, frames sending and reception service. The interface between the communication controller and the host part is the Controller Host Interface (CHI). The driver part consists of a bus driver which connects the controller to the communication bus and the bus guardian (optional) which protects the access to the communication bus.

4. THE CLOCK SYNCHRONIZATION SERVICE

The goal of the clock synchronization service is to maintain a global time-base that all nodes agree on. The FlexRay protocol utilizes the concept of microtick, macrotick and cycle to identify the time. Microticks correspond to the local oscillator ticks at each node, the macrotick consists of an integer number of microticks, and the cycle consists of an integer number of macroticks. Clock synchronization consists of two main concurrent processes, the calculation of offset and rate correction values using FTM (Fault-tolerant midpoint algorithm) process and the application of this corrections process. The offset correction value shall be calculated each communication cycle after static segment (measurement phase), and it's applied only during the NIT period of each odd communication cycle. The rate correction value shall be computed each even communication cycle after static segment (measurement phase), and it's applied during all segments of each communication cycle [1].

5. THE MEDIA ACCESS CONTROL SERVICE

The FlexRay protocol allows both time triggered and event triggered communication. In the time triggered communication, FlexRay uses a Time Division Multiple Access (TDMA) technique to control the access to the communication media in the static segment to enable collision-free bus allocation, the TDMA strategy permits to each node to periodically utilize the full transmission capacity of the bus for some fixed amount of time called a static slot. In the event triggered communication FlexRay uses a Flexible Time Division Multiple Access (FTDMA) technique to control the access to the communication media in the dynamic segment, the FTDMA strategy permits to transmit the messages nodes based on their priorities for some number amount of small time called a minislot [1].

6. CONCEPTUAL MODEL OF FLEXRAY SERVICES

FlexRay is a complex protocol for which the functional and oriented object methods are not able to deal with all aspects of this protocol. As mentioned before, we used the O-MaSE (Organization-based Multi-agent Systems Engineering) methodology [7] which is an extended version of MaSE that allows the design of the multi-agent organization. The main O-MaSE Models used in our MAS are, the goal model, the organization model, the domain model, the role model, the plan model and the protocol model.

6.1. Goal model

The main goal of FlexRay protocol is to guarantee a high performance communication for time triggered and event triggered safety critical real time automotive applications. This global goal is noted Goal0. The later can be divided into the communication between nodes (Goal1), the fault tolerate communication (Goal2), the deterministic communication (Goal3), and high data rate and flexible communication (Goal4). Goal2 can be divided into the sub goals: the node fault tolerance (Goal2.1) and the communication medium fault tolerance (Goal2.2).

6.2. Organization model

The aim of this model is to identify the system (the organization) interfaces with the external actors. There are two abstraction levels of the organization in our proposed simulation: the macro organization that represents the cluster; the external actor in this level is the main simulator and the micro organization that represents the node of the cluster; the external actors in this level are the communication medium and the main simulator.

6.3. Domain model

The aim of this model is to capture the object types, relationships and behaviors that define the domain in which agents will interact and reason about [8]. Domain model is similar to object class model of UML (Unified Modeling Language). In our design of FlexRay protocol, the domain model presents the different segments of FlexRay frame and the type of each field of each segment.

6.4. Role model

This model represents the roles that can be played by the system agents. Each leaf goal in the goal model must be assigned to one or more roles in the role model that can achieve it. A role may achieve multiple leafs goals. We identified thirteen roles: transmission service, send of static frame, reception of frame, and wakeup service, to achieve the Goal1. Startup service, clock synchronization and change of segments, control of communication controller states, and protection access to communication medium to achieve Goal2.1. Message event generation and control of FTDMA strategy to achieve Goal4. Arbitration and sending of dynamic frame to achieve Goal1 and Goal4. Redundancy mechanism to achieve Goal2.1 and Goal2.2. TDMA table Management to achieve Goal3. Each role has one or more capabilities representing the behavior that characterizes this role.

6.5. Agent class model

This model represents the system agent classes. Each role in the role model must be assigned to one or more agent classes that can play it. An agent class may play multiple roles. We identified three agent classes to play transmission service role: communication bus agent (we proposed that the medium communication is a bus), CHI interface agent and bus driver agent. Protocol operation control agent to play the roles: wakeup, startup, send of static frame, reception of frame, control of communication controller states, control of FTDMA strategy. Synchronizer and communication controller to play the roles: clock synchronization and change of segments, and TDMA Table management. Generator agent to play the message event generation role. Arbitrator agent to play the Arbitration and send of dynamic frame role. Bus guardian agent to play protection access to communication medium role. The redundancy mechanism role is played by instantiate two agents of the bus agent class.

6.6. Agent plan model

The plan agent represents an algorithm agent when he plays a specific role which requires a specific capability for achieving his specific goal. Again, because there are many different roles capabilities defined in the role class model, we have to develop at least thirteen roles; one for each capability.

6.7. Protocol model

This model defines the protocol in terms of messages passed between node agents when they play specific roles and the external node actors. We proposed three protocol models: the sending of the static frame, the sending of the dynamic frame and the reception protocol models. In the next of this paragraph we present only the sending of the static frame and dynamic frame protocol models. The sending of the static frame protocol begins when the synchronizer and communication controller agent which play TDMA table management role informs the protocol operation control which plays static frame sending role and the bus guardian agents that the current slot is assigned to their node to start the sending of a static frame. When the protocol operation control receives this message, it sends a request message to the CHI interface agent and waits for the received data. When this later is received, the protocol operation control integrates this data within the frame and sends it to the bus driver agent. This later in turns sends a request authorization message to the bus guardian agent and waits. Finally, when the authorization message is received, the bus driver agent sends a static frame to the communication bus agent.

7. THE PROPOSED ARCHITECTURE OF A FLEXRAY NODE

We assumed that each FlexRay node consists of seven agents; each agent has one ore many roles, and interact with others agents when they play specifics roles to achieve the different FlexRay services. Each node communicates with other nodes via the communication bus agent. Figure 1 shows the proposed architecture of a FlexRay node.

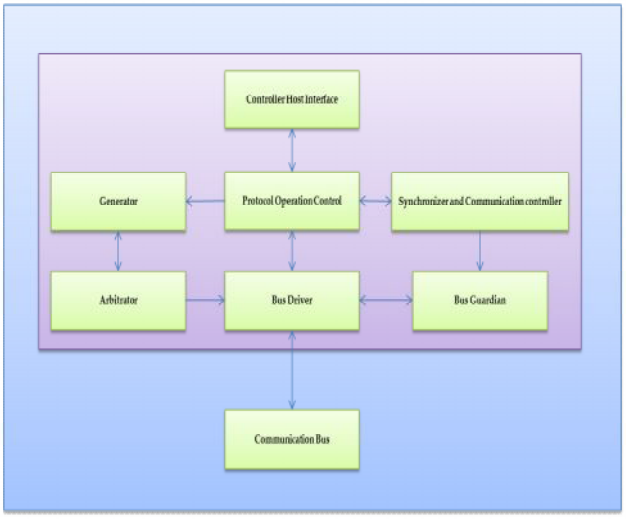

Figure 1. FlexRay node architecture

8. IMPLEMENTATION

We used JADE (Java Agent Development framework) [9] platform to implement our Multi-agent FlexRay protocol aspects and services. We choose JADE because it matches well O-MaSE methodology aspects such as the behavior of the agent, the possibility of creating new communication protocols, the possibility of defining a new ontology. We integrated the Jade with Java eclipse to create our graphical interface and exploit the JADE platform notation to implement the FlexRay cluster organization, and the container JADE to implement the node organization. We proposed to define a new ontology that provides an automatic interpretation of the frame fields and used the JADE language to automatically convert the format of exchanged messages between agents. Figure 2 shows an execution example of a FlexRay cluster consisting of three nodes each of them consists of five agents (CHI, protocol operation control, synchronizer-and-communication-controller, generator and arbitrator).

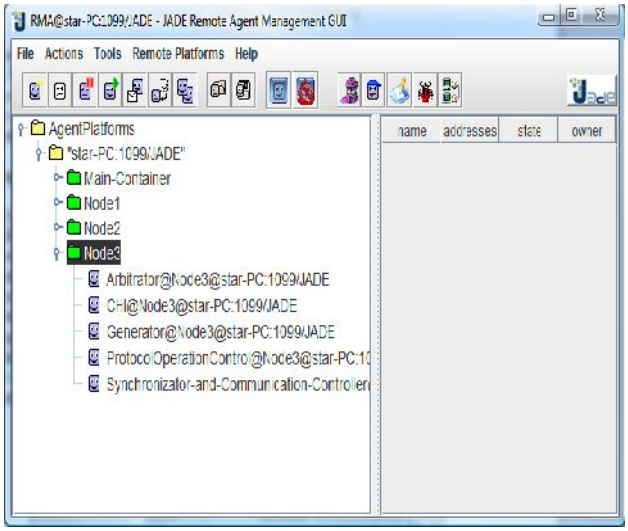

Figure 2. Running example of a cluster with three nodes

9. CASE STUDY

As a case study, we have coped with two FlexRay services that are the clock synchronization and the media access control services. We consider a single cluster with four nodes. Each node has an internal clock; node 1 is responsible of wakeup of all other cluster nodes; node 1 and node 2 are responsible of cluster startup. TABLE I summarizes the system configuration that we have used for the simulation experiment.

TABLE I. GENERAL SYSTEM CONFIGURATION

Number of nodes	4
Wakeup node	Node 1
Coldstart nodes	Node 1 and Node 2
Noncoldstart nodes	Node 3 and Node 4
Number of Microticks/Macrotick	5
Number of Macroticks/Slot	5
Period of static segment	20 Macroticks
Period of dynamic segment	20 Macroticks
Period of symbol window	2 Macroticks
Period of NIT	5 Macroticks
Clock rate of node 1	1 Microtick/230ms
Clock rate of node 2	1 Microtick/200ms
Clock rate of node 3	1 Microtick/240ms
Clock rate of node 4	1 Microtick/300ms
List of dynamic messages and their priorities	{Message1,2},{Message2,4}, {Message3,1},{Message4,7}

Figure 3 shows the evolution of produced macroticks by each of the four nodes before and after applying the synchronization algorithm of the FlexRay protocol. In the interval of time [T1,T2] (before applying the synchronization algorithm), node 2 (in the red color) generted a number of macroticks which is greather than the number of generated macroticks of node 1 (in the blue color), because the clock of node 2 has a rate higher than the clock of node 1. The same remark for node 3 (in the green color) and node 4 (in the purple color). In the interval of time [T3,T4] (after applying the synchronization algorithm), all the four nodes produced the same number of macroticks because the offset correction is applied during the NIT period of communication cycle and the rate correction is applied during all the cycle segments.

The result of media access control simulation during the static segment of communication cycle using TDMA strategy is commented as follows: in the slot0, node 1 sends a coldstart frame because it is a coldstart node; in the slot1, node 2 sends a coldstart node because it is a coldstart node; in the slot 2, node 3 sends a data frame because it is a noncoldstart node; in the slot 4, node 4 sends a data frame because it is a noncoldstart node. When each node sends a frame, all others nodes of the cluster receive this same frame.

In media access control simulation during dynamic segment of communication cycle, only non coldstart nodes can access to the communication bus during this segment. In cycle 15, node 3 generates message 2 with a priority value equals to 4, and node 4 generates message 3 with a priority value equals to 1. This tows nodes wanting to access the communication bus to send their dynamic frames at the same time. The node permitting to its message to be sent is node 4 because the priority of its message is higher than the priority of node 3 message. This rule of arbitration is applied to all cycles during dynamic segment. The TDMA and FTDMA strategies are applied perfectly because the synchronization service is applied during the startup phase (all nodes have the same view of global time).

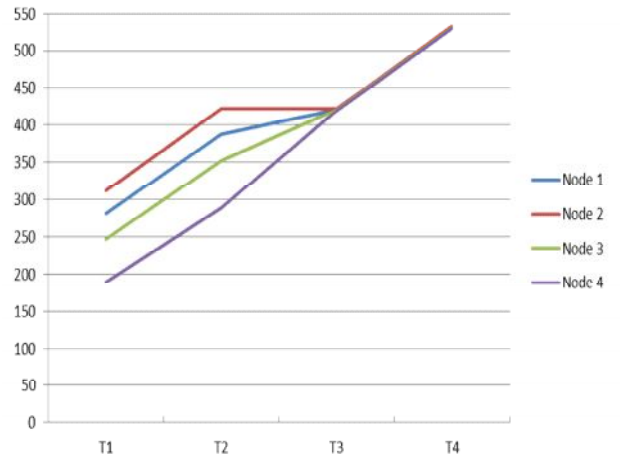

Figure 3. Evolution of produced macroticks by the four nodes before and after applied clock synchronization algorithm of the FlexRay protocol

10. CONCLUSION AND PERSPECTIVES

In this paper, we investigated the idea of applying the agent paradigm to model and simulate the basic services of the FlexRay protocol which allows both time and event triggered communication. In our case, we followed the O-MaSE methodology to build the goal, organization, domain, role, agent class, plan and the protocol models. We used the JADE platform to define the frame ontology and simulate our FlexRay multi-agent system. Our simulation of the FlexRay synchronization service provides good results: the cycle and macrotick counters values are equal for all nodes after applying the synchronization algorithm. FlexRay media access control service simulation provides the history of FlexRay nodes behavior during the static and dynamic segments of the communication cycle for sending and receiving static and dynamic frames. As a perspective, we plan to study the two services by increasing the range of simulation parameters like the number of microticks per macrotick, the number of macroticks per slot and the period of the cycle.

REFERENCES

[1] N. Navet, and F. Simonot-Lion, "Automotive embedded systems handbook,". CRC press, 2008.

[2] W. S. Kim, H. A. Kim, J.-H. Ahn, et B. Moon, "System-Level Development and Verification of the FlexRay Communication Controller Model Based on SystemC," in Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on, 2008, vol. 2, p. 124–127.

[3] F. Ren, Y. R. Zheng, J. Sarangapani, "FlexRay network utilization evaluations based on static and dynamic segments,". Proceedings of the 3rd Annual ISC Research Symposium. April 2009.

[4] V. R. K. R. Podduturi, "A SystemC simulator for the dynamic segment of the FlexRay protocol," Linköping, 2012.

[5] S. Buschmann, T. Steinbach, F. Korf, and T. C. Schmidt, "Simulation based Timing Analysis of FlexRay Communication at System Level," 2013.

[6] A. Zhao, "Reliable In-Vehicle FlexRay Network Scheduler Design," master of science thesis in Electrical Engineering, p. 17–23, 2011.

[7] D S. A. DeLoach, "Developing a multiagent conference management system using the O-MaSE process framework," in Agent-Oriented Software Engineering VIII, Springer, 2008, p. 168–181.

[8] S. A. DeLoach, and J. C. Garcia-Ojeda, "O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems," International Journal of Agent-Oriented Software Engineering, vol. 4, n° 3, p. 244–280, 2010.

[9] F. Bellifemine, G. Caire, T. Trucco, et G. Rimass, "JADE Programmer's guide,". JADE 2.44, September 2001.