An Architecture for Collaborative Driving Systems

Shou-pon Lin
Department of Electrical Engineering
Columbia University
New York, USA
shouponlin@ee.columbia.edu

Nicholas F. Maxemchuk
Department of Electrical Engineering
Columbia University
New York, USA
nick@ee.columbia.edu

I. Introduction

Vehicle automation has progressed from systems that monitor the operation of a vehicle and assist a driver, with functions such as antilock braking and cruise control, to systems that detect the operation of adjacent vehicles, to implement emergency braking and intelligent cruise control. The next generation of systems will share sensor readings and collaborate to control braking operations by looking several cars ahead or by creating safe gaps for merging vehicles. The rules that control the interaction between automobiles are protocols.

It is not sufficient to demonstrate that collaborative driving systems work. Before we allow these systems on public highways we must prove that these systems will do no harm even when rare, multiple failures occur. The failures will include loss of communications, failures or inaccuracies of sensors, mechanical failures of the automobile, aggressive drivers who are not participating in the protocol, and unusual obstacles or events on the roadways. To verify that such systems not only work but are also safe, we apply the techniques developed for communications protocols.

The problem with the protocols used in collaborative driving systems is that:

1) They are much more complex than communications protocols. Communications protocols interact with the physical world in a single way, through the communications channel, while collaborative driving systems interact with physical world in several ways, including the communications channel, a collection of sensors that detect conditions surrounding the vehicle, and the computers that monitor and control the operation of the vehicle. Each interaction with the physical world requires its own control and monitoring structure, and presents its own failure mechanisms;
2) They may have more than two parties participating in the protocol. Simple communications protocols have two parties. When the protocol that operates on each machine is modeled as a finite state machine, the verification process examines the composite state of all of the machines. The number of states increases exponentially with the number of parties that are participating.

To assist the design of such protocol as well as other intelligent automobile application, we need a framework on which we can build vehicular application. In [1] we have proposed a modular architecture that contains components that interface and control different hardware in the vehicles and the communication channel. The application we built can interact with these components without interfacing with the physical world directly or taking care of the implementation details of other component.

The modular architecture enabled us to simplify the design of a three party collaborative merging protocol that assists a driver who wants to merge between two cars in an adjacent lane. The protocol begins operation when the driver signals his intent to merge. The automatic cruise control systems in the cars creates a gap between two cars, lines the merge car up with the gap, and informs the driver when it is safe to move into the new lane. To verify that the protocol is safe, we applied probabilistic verification [2] to the extended finite-state machine of the protocol. We have verified that the protocol will not cause a potential accident by informing the driver when it isn't safe to merge for any combination of up to three rare events including communications failures, sensor failures, hardware failures in the vehicles, or changes in road conditions.

In this poster we extend the modular architecture in the previous work and propose a multiple stack architecture. The architecture divides the functionalities in an intelligent vehicle into layers that reside in three different stacks which represent multiple hardware platforms. In the layered architecture of communications protocols, protocol complexity is reduced by using the services provided by a lower layer. Similarly, the multiple stack architecture aims to reduce the complexity of application protocols that rely on the services provided by different stacks, as opposed to communications protocols that only need the service of a lower layer protocols. This architecture can be used for many collaborative driving applications.

II. Architecture Overview

The structure of the multiple stack architecture that we envision for collaborative driving system as well as other automobile applications is shown in Fig.1. There are three layered stacks that serve as the base, upon which several other components such as collaborative merging protocol and intelligent cruise control operate. The layered stack structure of the base mimics the layered architecture employed in communications protocols, while multiple stacks are a must to provide the application with access to various hardware platforms of a vehicle. The complex problem of collaborative vehicle control is thus partitioned into smaller components that can be implemented and tested individually, and each component could be modified separately without affecting the overall structure of the system.

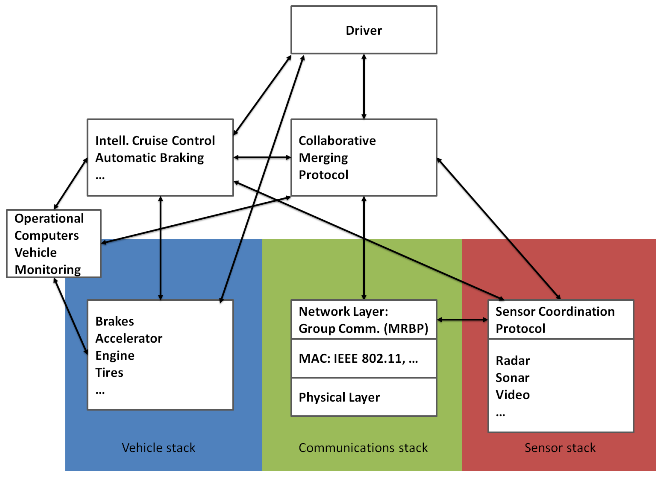

Fig. 1. Multiple stack layered architecture for collaborative driving system

The stacks have layered structure as in the communications protocols. Through well-defined interfaces between layers, the implementation details of each layer are abstracted out and could be designed individually. We can design a protocol by assuming that the lower layer functions correctly and provides the defined services. For instance, we write a distributed database without concern for manipulating the bits to access a communication channel. The complexity of a protocol could thus be reduced, and it would be easier to verify the correctness and the safety of a protocol.

However, a collaborative merging protocol or any collaborative driving application operates over multiple physical platforms. The functionalities in a system of intelligent vehicle include vehicle control, communications, and sensors. The single layered stack structure of communications protocols assumes that a higher layer only needs the services provided by a lower layer, which is not the case in automobile application. Therefore, we need an architecture with multiple stacks that address different types of interaction with the physical world.

1) The vehicle stack represents the kinematic aspects of a vehicle and consists of engine, tires, brakes, accelerator, etc. There could be a operational computers for vehicle monitoring in the stack, but we decided to separate it from the stack. The reason is that an application protocol like intelligent cruise control may access the vehicle stack either directly or through the operational computer.
2) The communications stack provides the automobile with communication capability. It retains common layers that we see in communications protocols. The highest layer is the network layer that provides the components a way to communicate with the components that reside on other vehicle. The network layer operates over MAC layer, which in turn operates over physical layer.
3) The sensor stack has the sensor coordination protocol as the protocol in its highest layer. The sensor coordination protocol exchanges the sensor readings with protocols on other vehicles through communications stack, then merges the sensor readings from different vehicles so as to make sure that all participants use the same set of information. The sensor coordination protocol relies on the sensor readings obtained by its lower layer, which could be any kind of sensors that are able to obtain measurement from the physical world.

Unlike communications protocols, in which the information is being passed between adjacent layers in a single stack. Our architecture also allows information to be passed between stacks. The sensor coordination protocol that uses communications is such example.

On top of these stacks the collaborative driving application could be defined. For instance, the intelligent cruise control system accesses the vehicle stack and controls the mechanical components to accelerate or apply brake, according to the information such as speed and distance to the car in front fed from the sensor stack. The collaborative merging protocol also relies on more than one hardware platform. It communicates with the protocols on collaborating vehicles through the communications stack, makes decision on the unified sensor readings, and sends orders to the intelligent cruise control system or automatic braking system.

Under this architecture, the complicated problem of collaborative driving is partitioned into smaller, manageable components. A component is analogous to subroutines in programming instead of duplicating code, and it enables intelligent vehicle function to be reused. These components have well-defined interface that determine what and how the information is passed between components. Each component can be designed and implemented in any ways as long as it conforms with the interface specified. The network layer in the communications stack could be either a point-to-point reliable retransmission protocol, or a group transmission protocol such as M-RBP [3] with delay guarantee. Similarly, the sensors on different vehicles can also be implemented in different ways. A vehicle may use radar, while another may use sonar, but both offer distance measurement. The modular structure allows each component to be independently modified and improved while not affecting the rest of the system.

References

[1] B. H. J. Kim and N. F. Maxemchuk, “A safe driver assisted merge protocol,” in *Systems Conference (SysCon), 2012 IEEE International*, pp. 1 –4, Mar. 2012.
[2] N. F. Maxemchuk and K. Sabnani, “Probabilistic verification of communication protocols,” *Distributed Computing*, vol. 3, no. 3, pp. 118–129, 1989.
[3] T. L. Willke and N. F. Maxemchuk, “Coordinated interaction using reliable broadcast in mobile wireless networks,” *Computer Networks*, vol. 51, pp. 1052–1059, Mar. 2007.