A Safe Driver Assisted Merge Protocol

Bo Hyun J. Kim
Department of Electrical Engineering
Columbia University of New York City
New York, USA
bk2410@columbia.edu

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

***Abstract*—we present a cooperative driving protocol that assists drivers in merging with other vehicles. We develop a modular architecture that is based upon the multiple hardware and communications platforms that must participate in this type of system. The architecture isolates the lower level functions related to the hardware from the application, and simplifies writing the merge protocol. We specify the protocol as an Extended Finite State Machine and validate it using Probabilistic Verification [1]. We demonstrate that for a wide range of hardware failures, communications failures, and unresponsive drivers, the protocol will not cause an accident. We also show that there may be situations with a very low probability, where the protocol will leave a vehicle in a state where it cannot participate in future merge operations without being reset.**

Keywords-system architecture, layered architecture, system validation, probablistic verification, merge protocol, assisted driving

I. INTRODUCTION

Our objective is to design a safe three party protocol to assist a driver who is merging between two other vehicles. We assume that all three vehicles are equipped with sensors that can detect the distance between vehicles, a communications system that will be used for collaborative maneuvering, and an advanced cruise control system that can adjust the distance between vehicles. The driver will signal his intent to merge, the vehicles will adjust their speed to create a safe gap, and notify the driver when it is safe to merge. The driver then performs the lane changing maneuver so that it is a driver assisted system rather than a robotic driving system. Our merge protocol differs from many existing solutions of automated car driven merging [1]–[6], because we present a driver assisted system, that ensures the safety of the protocol.

We prove that, with very high probability, there are no combinations of hardware or communications failures that will lead to an accident, or that will leave any of the vehicles in a state that will make them incapable of participating in future collaborations. We model the control portion of the protocol as an extended finite state machine, EFSM, shown in figure 3 and use a probabilistic verification technique, that was first used for communications protocols [7], and is described in section V, to prove that the composite machine for the three participants will not reach a failure state when there are less than a specified number of errors or failures.

The merge protocol, which is our application protocol, is built on top of other protocols that interface and control the hardware in the automobiles and the communications channel, rather than interfacing the physical world directly. Our architecture serves the same function as the layered architectures that are used in computer-communications systems, but is unique to automotive systems and has a richer structure. We believe that this architecture will be useful in a wide range of cyber-physical systems that are used for collaborative driving. Our architecture is described in section III.

The objectives of the architecture are to simplify writing collaborative driving applications and to also simplify the effort needed to verify these protocols. Layered architectures simplify writing applications by creating a higher level language. For instance, if a lower layer has a reliable transmission protocol, the application can transmit a message and assume that it will be received or fail, rather than implementing a retry procedure for each message. Layered architectures also simplify the verification process by reducing the complexity of the EFSM that describes the application. When we perform the verification of the application, it is based on the assumption that the services provided by the lower layer protocol have been proven, and are correct.

In section III, we present the details of our architecture. The difference between our architecture and the layered architectures in communications networks is that the automobile operates over several hardware platforms that have their own protocol architectures and can interact. The communications stack uses a Reliable Neighborcast Protocol [8] that provides a unique set of services that are useful for collaboration. The automobile's sensor stack uses the communications stack to coordinate the sensors in the different automobiles so that the protocols operating in the three vehicles are using the same data. The modular architecture we present differs from the hierarchical control architecture that was illustrated in [10], [11], [12], which are a vertical structure.

Our paper is organized in the following order. In section II, we describe the three generations of our merge protocol. Then our Driver Assist Architecture is illustrated in section III. In section IV, we describe the Extended Finite State Machine (EFSM) of our protocol. Using our EFSM, we performed the probabilistic verification, which is explained in section V. We conclude our paper with our findings in section VI.

978-1-4673-0750-5/12/$31.00 ©2012 IEEE

II. THREE GENERATIONS OF PROTOCOLS

The merge protocol and architecture that we present is the result of three generations of protocols. Our protocol is designed and implemented for the following scenario. Typically, there are three vehicles involved in a merge process with one vehicle that tries to merge when there is a safe space made between the front and the back vehicle in the other lane. Prior to beginning a merge process, the merge protocol will enforce that all three cars agree to be locked to maintain the safety of the protocol. Once locked into a merge process, cars can no longer be engaged in another merge process and other lock requests will be declined. Only upon some external or internal warnings, this lock can be released and cars can engage in a future merge process. All cars are tagged with unique RFIDs and this information is critical during the merge process.

The first generation operated directly over a communications channel that only provided best effort delivery. The application was responsible for recovering lost messages and coordinating the processes in three vehicles. The protocol was extremely complex, and the communications functions were duplicated in many places.

We built the second generation over a communications layer that implemented an Automatic Repeat Request (ARQ) protocol on a point-to-point channel with each of the other vehicles. It had two separate state machines for the sender that commands the merge and the receiver that responds to the request. This approach was also taken in [10] where they had two state machines but theirs did not include timers. Our application protocol used multiple timers to coordinate the vehicles and to detect the loss of a communications channel. When we applied probabilistic verification to the composite machine, we found combinations of two failures, which were caused by having multiple timers that may cause accidents. While we could correct the protocol, we decided to use a reliable broadcast protocol that provides a set of services which significantly simplifies the application.

Figure 1 illustrates the third generation that uses the Reliable Neighborcast Protocol (RNP). RNP is a timed token passing protocol that changes the broadcast group as vehicles move with respect to one another [8]. The protocol can recover lost messages and also notifies all of the group members when not all of the members recovered a message by a deadline. By using RNP, we significantly reduced the number of timers to a single, which is used to detect an unresponsive driver.

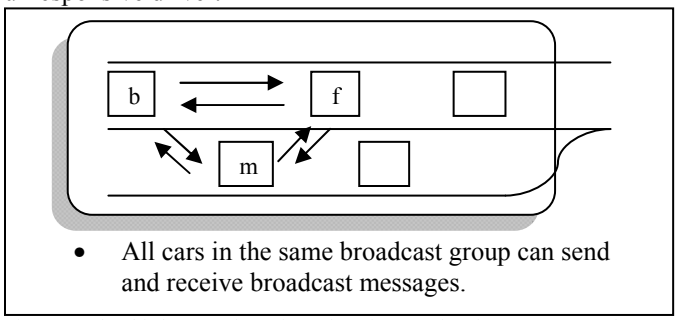

Figure 1. The third generation of a Safe Driver Assisted Merge Protocol

III. DRIVER ASSIST ARCHITECTURE

We present our modular architecture in Figure 2. The merge application operates over other applications, which are an automated cruise control system, hardware platform that includes sensors in all three vehicles, a communications channel, and an interface to the driver. The architecture is modular, rather than layered, which allows richer interaction between the components.

The driver signals when he would like to merge, and the Merge Protocol notifies him when it is safe, or warns him when a dangerous condition occurs, such as a non-cooperating vehicle entering the area. The Merge Protocol requests a safe merge separation from the intelligent Cruise Controls in the appropriate vehicles. The Cruise Controls uses the sensors to calculate the optimal separation.

The Merge Protocol does not interact directly with the sensors in a vehicle. Instead, the application interacts with a Coordinated Sensor Readings (CSR) module that merges all sensor results from three vehicles, using Reliable Neighborcast Protocol (RNP). It guarantees that all applications in all of the vehicles are presented with the same information.

The RNP, described in [8], operates over a standard layered architecture that may include an 802.11 device. It provides a broadcast service between the three vehicles and simultaneously notifies all the vehicles when a message has been delivered, or when it was not delivered by a deadline. It also notifies the vehicles when the communications group has changed, due to a loss of communications or a vehicle moving out of the area. Any erroneous message sent from RNP will immediately abort the Merge Protocol, preventing accidents or dangerous situations.

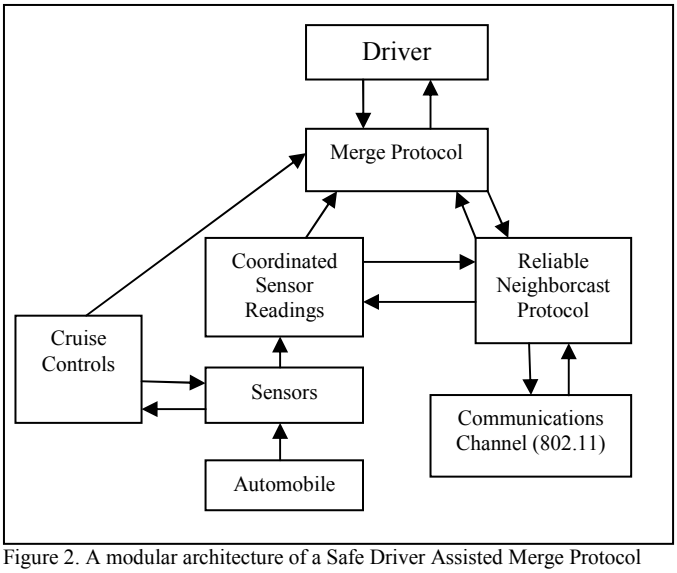

Figure 2. A modular architecture of a Safe Driver Assisted Merge Protocol

IV. EFSM

We model the merge application as an Extended Finite State Machine (EFSM) as shown in figure 3.

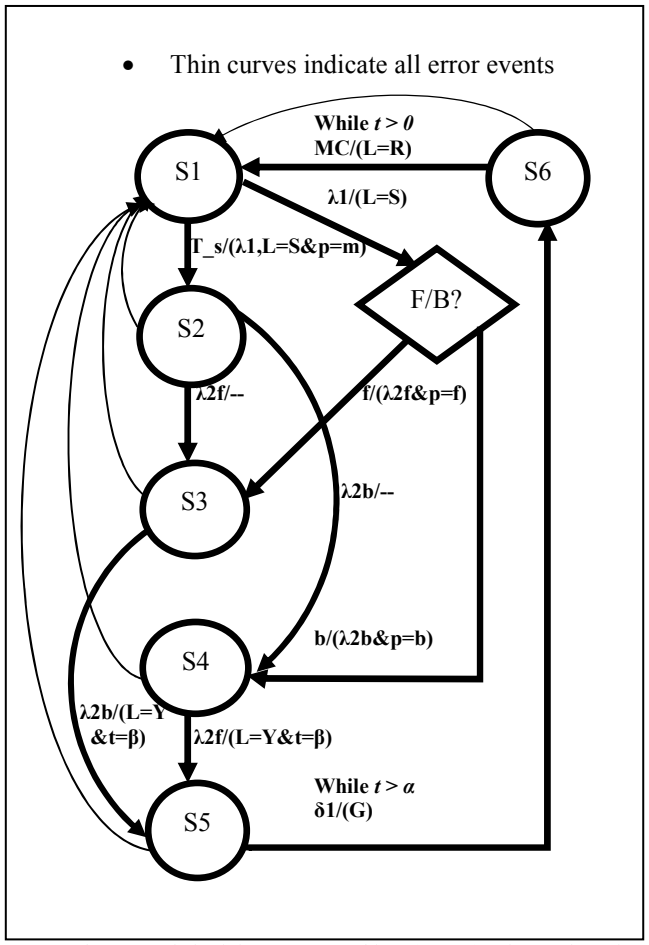

Figure 3. The Extended Finite State Machine (EFSM) of a Safe Driver Assisted Merge Protocol

The merge application can only be initialized when a driver signals his intent of merging. When the driver signals, it triggers the messages to be sent to other vehicles involved. It learns its position ('m' as merging, 'f' as front, 'b' as back car) by receiving different messages. Each car's identity is determined in state 1 by receiving either 'T_s' or 'λ1', which are a turning signal from the driver and a lock request, respectively. 'T_s' triggers the merge process which then sends out a message 'λ1' and signals the light.

The merge car will wait in S2 for lock grants from both the front and the back cars. In all states except S1, important messages such as the lock grant/deny will always have a parameter, 'p' (one of 'm', 'f', or 'b') to indicate the position of the car. Once determined as either the 'front' or the 'back' car in the comparator that branched out from S1, each car will respond with either a lock grant, 'λ2p' or a lock deny, 'λ3p'. In both S3 and S4, all vehicles will wait for the remaining lock grant messages to safely proceed to the next state. In S5, the timers are initialized as ($t = \beta$) for each car and while any one of those three timers remain in condition of ($t > \alpha$), cars will wait for the unified sensor reading, 'δ1', that implies the safe space has been made and that the merging can begin. Upon receiving 'δ1' from Coordinated Sensor Readings, a green light turns on for the driver and for the remaining period, which is for ($t > 0$), cars will wait for the driver to merge safely into the merge space made. Upon receiving 'MC' which implies that the merge is complete, the system will go back to the initial state, S1 and prepare for the next merge process. The timers in S5, which extend to S6 unless a timer (hardware) failure occurs, were implemented to prevent deadlock situation where cars might indefinitely wait for messages such as 'δ1' or the merge completion message, 'MC' in either of those states. We use probabilistic verification to validate our EFSM which is presented in the next section.

V. PROBABILISTIC VERIFICATION

Probabilistic verification [7] is a formally directed simulation technique that generates a portion of the composite machine in all three vehicles from the component machines. In [10], they also proposed a formal specification of the merge protocol and its state machines. They defined all possible set of output sequences, *L* and the acceptable output sequences, *M*. They then checked that *L* is a subset of *M* to verify the protocol. In contrast, our approach defines all unacceptable behaviors of the protocol as *U* and for all states, *S*, we test for all possible sequences of states that may or may not contain *U* to verify that our protocol is safe. To simplify, we divided these sequences that cause state changes into two groups: which are high probability events such as message receptions that occur during the normal operation of the protocol, and low probability events such as the loss of communications or a hardware failure that don't usually occur.

The Zigangorov-Jellinek stack processing algorithm, that was first used to decode convolutional codes [9], is used to examine all of the paths that have no low probability events. We first examine all paths that contain a single low probability event, and then examine all paths with two low probability events, and so on. When we examine a path, we determine if some of the vehicles believe that there is an agreement to merge, when others don't, which may cause an accident. We also determine if it is possible for the protocol to terminate with some vehicles in a state where they cannot participate in the next merge. These events are caused by low probability events and we defined them to be tested in each state.

The following six low probability events may create abnormal operation. Such events are:

1. Aborting the merge process due to a sudden brake from the front vehicle.
2. Loss of communication from the communications channel.
3. Detection of unauthorized RFID tagged vehicles.
4. Lasting disagreement in sensor readings from vehicles.
5. Unresponsive human behavior.
6. Hardware failure from timers.

We expand a probabilistic search tree to implement the verification model using the above six error cases. In our search tree, the most likely states are investigated first. It then explores the less probable states of the tree. For example, a

successful transmission of messages is a high probability event. We stop examining sequences when the probability of occurrence is sufficiently small. We only show a part of a probabilistic search tree we used for validation in figure 4. The state transitions are shown in a form of (Sm Sf Sb E/A), where Sm, Sf, Sb are the states of the merging, front, and the back car, respectively. The state transitions can be followed with a parameter, E, error or A, accept.

When the verification is complete we can state that for *k* or fewer low probability events, we will not enter one of the composite states that are unsafe or that cause a deadlock. Based upon the upper bound on the probability of a low probability event, and the frequency with which we engage in the merge operation, we can determine the likelihood of a protocol failure during the lifetime of the vehicle.

We performed verification on our third generation machine by extending the probabilistic search tree. Figure 4 is a part of the probabilistic search tree we used. As a result, we found that the protocol is safe and does not cause failures or accidents. Using Probabilistic Verification, we proved that our merge protocol does not fail in situations where some car thinks it's safe to merge when others do not. We found that the protocol can result in a deadlock when three low probability events occur, which are caused by three independent timer failures in the three vehicles. We conclude that this is a very rare event and will not result in accidents or situations where vehicles lose their capability to form another merge group.

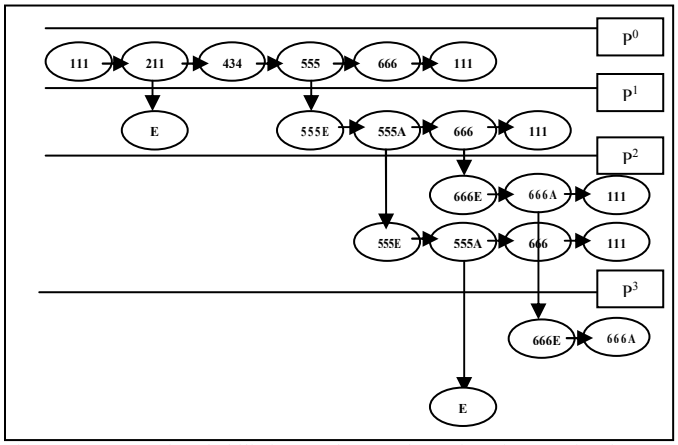

Figure 4. A part of probabilistic search tree of a Safe Driver Assisted Merge Protocol

VI. CONCLUSION

We present a new architecture for the automotive systems and its system verification. We introduce a modular architecture that allows rich interaction between multiple platforms and leads to the simplicity of verification. Using our Extended Finite State Machine, we generate error sequences that may cause failure or accident in our merge protocol. We then use Probabilistic Verification to examine all possible sequences of events that are both of high and low probability. We proved that our system will not result in accidents or system failures and that only at a very low probability, will result in a safe deadlock situation.

REFERENCES

[1] P. Hidas, "Modelling vehicle interactions in microscopic simulation of merging and weaving, Transportation Research Part C: Emerging Technologies", Vol. 13, Issue 1, Feb. 2005, pp. 37-62.

[2] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode. Traffic View: traffic data dissemination using car-to-car communication. SIGMOBILE Mob. Comput. Commun. Rev. 8, 3. Jul. 2004, pp. 6-19.

[3] A. Hsu, E. Sonia, F. Eskafi, P. Varaiya. "The design of platoon maneuvers for IVHS," American Control Conference, 1991. Vol. no., pp. 2545-2550, 26-28. Jun. 1991.

[4] Kato, S.; Tsugawa, S.; Tokuda, K.;Matsui, T.; Fujii, H.; , "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," Intelligenet Transportation Systems, IEEE Transactions on, vol.3, no.3, pp. 155-161. Sept. 2002.

[5] R. Vurulingesh, V. Shingde, P. Goyal, K. Ramamritham, A. Gudhe. "Algorithms for Automatic Merging of Vehicles."

[6] Z. Wang, L. Kulik, K. Ramamohanarao. "Robust Traffic Merging Strategies for Sensor-Enabled Card Using Time Geography," ACM SIGSPATIAL Internation Conf. 2009.

[7] N. F. Maxemchuk, K. K. Sabnani, "Probabilistic Verification of Communication Protocols," Distributed Computing Journal, Springer Verlag, no. 3, Sept. 1989, pp. 118-129.

[8] N. F. Maxemchuk, P. Tientrakool, T. L. Wilke, " Reliable Neighborcast," IEEE Trans. Veh. Technol., vol. 56, pp. 3278-3288, Nov. 2007.

[9] Haccoun, D.; Ferguson, M.; , "Generalized stack algorithms for decoding convolutional codes," *Information Theory, IEEE Transactions on*, vol. 21, no.6, pp. 638 – 651, Nov. 1975.

[10] P. Varaiya. "Smart cars on Smart roads: problems of control," IEEE transactions on Automatic Control, 38(3): 195-207, Feb. 1993.

[11] P. Varaiya, "Control design of an automated highway system," Proceedings of the IEEE, 88(7): 913-25, Jul. 2000.

[12] T. Simsek, P. Varaiya, J. Borges de Sousa. "Communication and control of distributed hybrid systems," Proceedings of the American Control Conference, 2001, vol. 6, no., pp. 4968-4983 vol. 6,2001.