Safe Collaborative Driving Systems

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Project Number 1329593 *N. F. Maxemchuk, Shou-pon Lin, Yitian Gu*

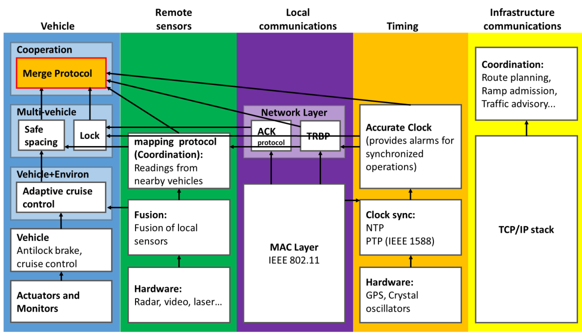
Objective

To apply the testing and design techniques developed for communication networks to intelligent vehicles.

A collaborative lane merge protocol is used to test the techniques .

Previous Accomplishments

1. Multiple Stack Layered Architecture



- One stack for each interaction with the physical world.
- Well defined Interfaces
- Change and test one layer without affecting others

2. Synchronized Clocks

- Reduce number of sequences of events by having events occur simultaneously, rather than in different orders – *Simplifies verification*
- New Protocols

1) *A **Broadcast protocol** with a unique message that cannot be lost*
   - Each vehicle communicates at scheduled times.
   - The message that is not sent at the scheduled time cannot be lost *(It will not be received when it is not transmitted)*.
   - This message is used to guarantee that an emergency operation, such as aborting a lane merge, will take place at all vehicles.
2) *A **Lock protocol** in which all vehicles simultaneously release the lock, even when communications is lost.*
   - Used to guarantee that each vehicle participates in only one collaboration at a time.
   - This reduces the difficulty to verify that there are no dangerous interactions between vehicles. We must only prove that a single protocol is safe, rather than proving that all combinations of protocols are safe.

3. Probabilistic Verification

- **Objective:** Bound the probability that unexamined interactions between vehicles may cause an accident, to a level that is required in automotive applications.
- *Vehicle safety requirements are extremely demanding* – We have verified that the merge protocol only enters an unexplored state, and may fail, less than once every **$5*10^{13}$** protocol invocations.
- Each component of the architecture may fail, and this failure affects the failure rate of the other components that are dependent upon it.
- By designing the interconnections between protocol components, in the architecture, to eliminate feedback loops, we can bound the individual protocols separately, and obtain a safety bound for the entire vehicle.

Current Work

1. Conformance Testing

***Definitions**:*

1. Conformance testing guarantees that protocols implemented by different vendors will interoperate.
   - We assume that the protocols have been verified and that the vendors will interoperate if the protocols are implemented correctly.
2. The Rural Postman package, developed by Bell Labs for telephone systems, tests each implementation against a finite state machine (FSM) model of a protocol, instead of testing implementations against one another.
   - If there are "N" implementations of a 2-party protocol there are N, rather than N^2 tests.
   - The Postman package finds a tour with a relatively small number of tests

***Problem**:*

- Collaborative driving systems have time critical operations.
- The FSM model in the postman package cannot accommodate timers.

***Solution**:*

By design, the protocols in our architecture do not have timers, or tests based on time.

- Instead, all time related events are handled in the timing stack.
- A protocol sends a message to the timing stack to request an interrupt, and receives a message at the appropriate time.
- Messages are consistent with the FSM model in the Postman package, and all of the protocol implementations, excluding the timing stack and physical interfaces, are tested with Postman sequences.

2. Protection Against Malicious Users

Implementation:

The intelligent interactions between vehicles are defined as finite state machines. There is a small allowable message set for each machine, and external hackers have limited access to the software.

Punishment:

Messages in collaborative operations are signed. We assume that intentionally causing accidents is illegal, and that identifying malicious users will act as a deterrent.

Prevention and Detection:

Two types of messages:

1) **Data messages:** e.x. sensor reading for distances between vehicles
   We require measurements from multiple vehicles to agree before participating in a collaborative operation. This prevents a single vehicle from affecting the collaboration.
2) **Control messages:** Messages that cause the transitions in the FSM's.
   - We have defined a set of potentially dangerous situations, such as two vehicles moving into the same space.
   - We assume that a malicious user can send any combination of the messages allowed by the FSM's. We use a depth first search to determine if there is any message combination that will result in a dangerous situation.
   - This analysis has resulted in a change in the lock protocol, to prevent a malicious user from claiming that he has not granted a lock when he has.
   - We are automating this testing procedure.