SECTION 3:
This section surveys a bunch of modular architectures and two layered architectures. Modular architectures aim to organize functions so as to differentiate each technology such as communication, control systems and sensors. This makes it easy to engineer and test the modules individually. The modules are grouped to aid the differentiation in their hardware implementations. The cooperative adaptive cruise control system, PATH project, had communications between vehicles participating in the maneuvers, and platoon formation. The SARTE, a European funded project deals with strategies to help platoons in a public highway. Cooperative driving systems often assume all communicating vehicles follow the same control strategy and system. There are few modules that consist of layers, which help differentiate time constants and rates of communication required. Automated driving mode was the next step suggested in the paper. HAVEit, a project, is developing intelligent vehicles that switch between semi-automated and automated modes depending on the situation. Moving on to layered architecture; one for platoons and one for sensors. The platoon architecture is to deal with low-level functions controlling actuation and high level functions that perform traffic management. Sensor technology aims at designing a fusion algorithm, giving importance to the heterogeneity and asynchronous property of the sensors. The key point stated is that sensor technologies can evolve without having to change the applications that use the sensors. These architectures deal with a specific problem. The platoon architecture involving automated highway control has the driver handing over control to the automated system upon reaching the highway. It has 5 layers. The physical layer has the controllers and takes care of the decoupling of longitudinal and lateral vehicle guidance that help simplify the design of the regulation layer. The regulation layer handles the maneuvering if the vehicle is a platoon leader. If it’s not, then it handles the spacing from the preceding vehicle. This layer monitors entries and exits. The coordination layer maps out a specific maneuvering process to accommodate the activity plan laid out by the link layer. It also supervises the regulation layer. The link layer controls traffic flow on the highway stretching from 0.5 to 5km. It receives instructions from network layer. The network layer keeps track of traffic entering the highway, and planning of routes. The aim is to reduce congestion, and maximize capacity of the highway, leading to minimal average vehicle time. The layered architecture enables individual verification of each layer. A comparison of the Collaborative driving system and the PATH project suggests a similarity between the two; the fact that both rely on inter vehicle communication for maneuvering. The sensor architecture is proposed for the BMW research project. The system detects traffic in the environment and maps it. The 3 layers aid the design of a fusion algorithm, allowing components to be upgraded without implementing the whole system again. The sensor layer takes data from each sensor and outputs a list of objects. The outputs of different sensors may be received out of sync and involve variable delays. The fusion layer handles this issue by putting together all the outputs into a map, which makes it easy for the application layer to function without having to consider properties of individual sensors. An example of this architecture given is the project SASPENCE.
In the previous section, a set of reference architectures was illustrated. This section describes a specific architecture for intelligent driving applications on lines of the concepts in communication networks. The complexity of such systems arises as intelligent vehicles interact in more ways than communication protocols. Further, they are time critical and are large in number. The first issue of interaction in many ways is tackled by organizing the interactions into separate stacks, dimensions so the hardware is isolated from the software. Each layer has a set of functions. A function in one box can provide services to other functions in another box belonging to a different layer. The stacks are created in multiple stack architecture similar to communication networks in order to separate higher-level logic functions from physical implementations. The difference in this architecture is that these layers can use services provided by another stack as well as the layer beneath it, and they can be tested independently as long as there is no direct loop involving other layers. In the architecture mentioned in the paper, there is a loop between an anti-lock braking routine, and actuators on the wheels. These cannot be designed and tested individually; they have to be done in one unit. Higher-level applications use services offered by the lower level applications. The second layer applications only use information about the vehicle to make an informed decision. The third layer uses information from local sensors, and detects the environment for vehicles. A modification of the speed set by the cruise control is employed to maintain a safe following distance. There are lane maintenance systems to notify drivers when a lane change is detected and a park assist system to detect parking of cars. The fourth layer uses sensors from adjacent vehicles to improve the operation of the system. Monitoring adjacent vehicles rather than local sensors has an advantage of getting an earlier response in case of emergency braking or other measurements and road conditions. The fifth layer communicates between nearby vehicles to gather relative distances and provide with the merge process. There are two communication stacks in the architecture; local stack for nearby vehicle communication and infrastructure stack for all other communications. The TCP/IP stack used in the Internet is a best effort network; that is there is no guarantee that packets will reach on time. The infrastructure stack gathers reports for traffic estimation, route planning and incident reports for the roadway. This information enables efficient merge process. The highest layer of the local communication stack supports protocols specific to collaborative driving functions. For example, a version of Mobile Reliable Broadcast Protocol (MRBP) guarantees collision free merge protocol. This protocol works on unanimous voting for all transmissions. The timing stack guarantees useful vehicle coordination by suggesting a class of synchronized protocols. Timers are preferred rather than comparing with local time. There are 3 layers; the hardware layer using GPS receivers and crystal oscillators to maintain a local clock, the clock synchronization layer to communicate between vehicles and synchronize all the clocks together, and the third layer to maintain timestamps of participant vehicles in protocols. The sensor stack combines readings from all sensors to map the situation. An added feature is the coordination between vehicles to use identical maps. Two examples of protocols are mentioned; the lock protocol that guarantees a vehicle to be engaged in only one merge process (when exceeding 3 participant vehicles, a vehicle cannot be unlocked till all 3 vehicles are ready to be unlocked), and a collaborative merge protocol, which allows the driver to merge between two cars in a nearby lane. A driver requests a merge, and is notified when there are enough gaps for the merge to take place. The MRBP protocol has pre-specified transmit times. The protocol in a vehicle is aborted when a local fault is detected, and like a chain process, all local vehicles abort within a token cycle time of fault detection.