Chapter 8
Fault Tolerant Architecture for Infrastructure based Vehicular Networks

João Almeida, Joaquim Ferreira and Arnaldo S.R. Oliveira

Abstract Wireless vehicular communications have been a trending topic in the last few years, leading to the development of a complete set of new standards and the emergence of innovative vehicular applications. Despite the obvious benefits of vehicular networks, it has been a challenging issue to design dependable vehicular communication systems. This is mainly due to the high speed mobility scenarios that are involved and the open nature of these networks. As a consequence, there are scalability problems with the proposed medium access control (MAC) methods under highly dense traffic environments. This results in large values for the end-to-end delay and for the probability of packet drops, compromising the reliability of vehicular communications. Besides that, there are few strategies to enhance fault-tolerance in vehicular systems, whose operation strongly depends on the dynamic topology of the network and on the real-time guarantees provided by the communications protocol. Based on these arguments, this chapter presents a fault-tolerant architecture to improve the dependability of infrastructure-based vehicular networks. The presence of road-side units (RSUs) and a backhauling network adds a degree of determinism that is useful to enforce real-time and dependability, both by providing global knowledge and supporting the operation of collision-free deterministic MAC protocols. One of such protocols is V-FTT, for which the proposed architecture was designed as a case study. Notice, however that this architecture is protocol independent and can be adapted to any wireless communications system. The chapter's final sections specially focus on the design of fail silent RSUs, by presenting the proposed implementation and the obtained experimental results.

J. Almeida (✉) · A.S.R. Oliveira
Instituto de Telecomunicações, DETI - Universidade de Aveiro, Aveiro, Portugal
e-mail: jmpa@ua.pt

A.S.R. Oliveira
e-mail: arnaldo.oliveira@ua.pt

J. Ferreira
Instituto de Telecomunicações, ESTGA - Universidade de Aveiro, Aveiro, Portugal
e-mail: jjcf@ua.pt

M. Alam et al. (eds.), *Intelligent Transportation Systems*,
Studies in Systems, Decision and Control 52,
DOI 10.1007/978-3-319-28183-4_8

8.1 Introduction

Wireless vehicular networks, as a fundamental area of research in modern Intelligent Transportation Systems (ITS), aim to improve vehicle and road safety, passenger's comfort, efficiency of traffic management and road monitoring. Vehicular communications rely on the recent IEEE 802.11-2012, IEEE 1609 and ETSI ITS-G5 family of standards, in which there are still some open problems concerning the timeliness and dependability of the exchanged messages [12]. In order to ensure correct operation even under dense traffic conditions, vehicular communication nodes and protocols should be developed by taking into consideration the questions that commonly arise in the design of dependable real-time systems [1], namely deterministic operation, timely behaviour, safety, reliability, availability, among others. The need for these design concerns is even more evident, given the scenario described next.

During a transitory market penetration period, vehicular communication systems will be regarded as an auxiliary technology that could aid driver to take more informed decisions and warn him about dangerous situations. However, after this initial phase, and with the advent of autonomous vehicles, road traffic systems will completely rely on the information provided by this and other technologies (such as cameras or radars). At that stage of development, the human judgement, which is very prone to error, will likely be replaced by computer-driven decisions. In this scenario, dependability will be an essential aspect in road traffic safety systems, since their operation will strongly depend on the correct service provided by vehicular communication devices. Based on these arguments, vehicular networks must be analyzed as distributed computer control systems (DCCS) that interact together to achieve the common goal of guaranteeing traffic safety.

In the last few decades, DCCS have been widely used in many application fields, such as robotics, industrial process control, avionics and automotive systems. A large number of these applications pose strict requirements, which if not fulfilled may cause important economic and environmental losses or even put human life in danger [14]. These systems must exhibit a high probability to provide continuous correct service. Beyond that, many of them comprise real-time activities that must be performed within stringent time bounds. Therefore, in safety-critical DCCS with real-time constraints such dependability attributes are of uttermost importance, since its distributed nature requires a timely and reliable exchange of data among the several nodes, in order to achieve the envisaged control over the operating environment.

Due to the nondeterministic behaviour of the environment where each specific distributed computer control system operates, the use of the best design's practices does not fully guarantee the absence of faults. Thus, fault-tolerance methods need to be included in the system to avoid that possible faults can cause its failure. By combining these kind of mechanisms with system's real-time requirements, dependable DCCS can be developed for operating in safety-critical scenarios, such as the ones existing in vehicular environments.

Several types of faults must be considered in vehicular scenarios. For example, the wireless channel is regularly affected by transient faults in the communication link due to the constantly changing atmospheric and road traffic conditions. This

effect is much larger than the one observed for instance in wired or indoor wireless environments. Furthermore, channel permanent faults could also occur, due to unregulated interference, which can typically be considered as malicious faults, since the spectrum band assigned for wireless vehicular communications is reserved by law. Not only the wireless channel, which is a single point of failure in vehicular systems, but also the nodes of the network should be regarded as a possible source of problems. For instance, hardware and software faults can affect the operation of either the road-side units (RSUs) that constitute the network infrastructure or the on-board units (OBUs) placed inside each vehicle. For the given reasons, a careful design work must be performed in the deployment of vehicular communications systems, by taking into consideration the general dependability aspects of safety-critical applications, as well as the specific issues that arise in the vehicular context.

8.2 MAC Issues in Vehicular Networks

The medium access control (MAC) layer defined in IEEE 802.11 adopts a carrier sense multiple access with collision avoidance, in which collisions may occur indefinitely, due to the non-determinism of the back-off mechanism. Therefore, native IEEE 802.11 alone does not support real-time communications. However, this property is crucial for safety applications, where warning messages have to be always delivered with a bounded delay.

The probability of collisions occurring may be reduced if the load of the network is kept low, which can be accomplished by using an adaptive and distributed message-rate control algorithm, as described in [2]. Although this type of solution can reduce the probability of collisions, it does not provide strict real-time guarantees. Collision-free MAC protocols are considered deterministic as data collisions do not occur and a worst-case delay from packet generation to channel access can be calculated. This can only be achieved if the protocol restricts and controls the medium access to provide a deterministic behaviour.

In the design of a deterministic MAC protocol for vehicular communications, there are basically two main possible choices. The protocol could rely on the road side infrastructure [5, 18, 20] or it could be based on ad-hoc networks [11, 17, 23]. A hybrid approach that takes advantage of both models could also be implemented. Strict real-time behaviour and safety guarantees are typically difficult to attain in ad-hoc networks, but they are even harder to achieve in high speed mobility scenarios, where the response time of distributed consensus algorithms, e.g. for cluster formation and leader election, may not be compatible with the dynamics of the system. Therefore, the presence of the infrastructure, e.g. road-side units (RSUs) and the backbone cabled network, adds a degree of determinism that is useful to enforce real-time and dependability at the wireless end of the network.

In addition to this, if the road-side infrastructure can be made more predictable than the other parts of the network, by executing certain tasks faster or in a more reliable way, the system could be seen as an instantiation of the *wormhole* metaphor [29].

In this scheme, the uncertainty is neither uniform nor permanent across all system components, and the road-side units which are connected through the backhauling network, can be seen as *wormholes*, since they are more reliable and predictable than the mobile nodes of the network (the on-board units—OBUs).

8.2.1 Overview of V-FTT Protocol

Based on the above arguments, a deterministic medium access control (MAC) protocol was proposed [13, 26], taking advantage of the road-side infrastructure. This protocol, entitled Vehicular Flexible Time-Triggered (V-FTT), adopts a multi-master multi-slave spatial time division multiple access (STDMA). The road-side units (masters) are responsible for registering the on-board units (slaves) with the infrastructure and for scheduling their transmission slots. These masters are synchronized through GPS receivers placed in each one of the nodes. They also share common knowledge of the road traffic system, which is attained by exchanging update messages holding the current state of each individual RSU's database. These messages are transmitted through the backhauling network (e.g. fiber optics), enabling RSUs with a global vision of the entire vehicular network.

The protocol is divided into periodic elementary cycles (ECs) with 100 ms duration and, as can be seen in Fig. 8.1, each EC is further divided into three different parts. It starts with an Infrastructure Window, that is used by the RSUs to transmit two types

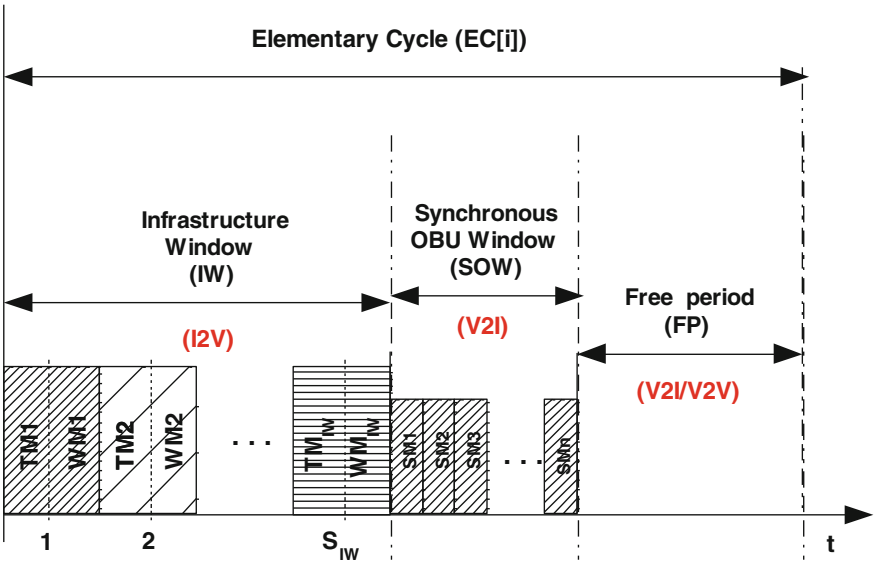

Fig. 8.1 Elementary cycle of Vehicular Flexible Time-Triggered protocol [26]

of messages: trigger messages (TM) with the schedule of the registered OBUs; and warning messages (WM) with cross-validated information regarding safety events that occurred in the road. In vehicular networks, these messages sent by the RSUs to the OBUs are commonly referred as Infrastructure-to-Vehicle (I2V) communication (Fig. 8.1). The second part corresponds to the Synchronous OBU Window, where each OBU has a fixed size slot to send information to RSUs (Vehicle-to-Infrastructure or V2I communication), either regular data (like vehicle's speed and heading) or a safety event. The Synchronous OBU Window duration is variable, depending on the number of OBUs (denoted as n in Fig. 8.1) scheduled in that particular elementary cycle. At the end of the EC, there is a free period, during which non V-FTT enabled OBUs can communicate (Vehicle-to-Vehicle or V2V communications), and the V-FTT nodes (both RSUs and OBUs) can exchange non-safety messages.

In order to increase the probability of successful reception of a TM or a WM by an OBU, RSUs' coverage areas can partially overlap, causing the same OBU to receive TMs/WMs from different RSUs. This redundancy level (S variable in Fig. 8.1) can be dynamically configured, depending on the number of RSUs in the same region and on the transmission power and sensitivity levels of the nodes of the network. However, some coordination is required among RSUs in order to ensure that for a given OBU, the transmission slot allocated to it in the Synchronous OBU Window (SOW) is the same in all TMs transmitted by the distinct RSUs. This is based on the fact that each OBU will only transmit a single message per EC. In a similar way, the OBUs' messages can be listened by several different RSUs during the SOW period.

From the above description, one can conclude that in the proposed protocol, RSUs play an extremely important role, since they are responsible for all traffic scheduling and admission control mechanisms. If an RSU stops working properly, all communications in its coverage area can be severely compromised. There are no guarantees anymore regarding the timeliness and deterministic properties of the protocol. This leads to the development of fault-tolerance mechanisms that are able to improve the dependability of V-FTT and other deterministic protocols with the same characteristics.

8.3 Fault-Tolerance Techniques

As it was referred in Sect. 8.1, the design of dependable systems is becoming pervasive in many domains, e.g. in automotive vehicles, in avionics, in nuclear plants, in factory automation, etc. These systems are usually distributed and rely on a communications network to interconnect sensors, actuators and controllers in a reliable and timely way. Although dependability aspects are traditionally studied in fieldbus technologies and wired networks, the same fundamental concepts apply to wireless systems. Since vehicular communications are expected to provide safety-critical road services with a high level of reliability, this work aims to develop methods to enhance dependability in vehicular networks. A real-time communication protocol, such as the one previously discussed, can attain an higher probability of deliver correct safety

services if dependability attributes are considered and mechanisms to improve them are implemented. For that purpose, fault-tolerant techniques need to be included in vehicular communication systems, in order to cope with the presence of unpredicted operating problems.

In the specific case of real-time communications, the mechanisms to achieve fault-tolerance are not just concentrated in the network nodes through, for instance, voting schemes or hardware redundancy. The faults propagated in the communication channel may also compromise the operation of the distributed system, since erroneous information can be shared and the ability to achieve common knowledge could be destroyed. A faulty message transmitted by a malfunctioning node may be propagated to all other nodes causing the whole system to collapse. Notable examples of these latter faults are timing or value failures in a node or replica non-determinism. Therefore, methods to ensure the validity of the transmitted messages should be employed in the design of a real-time communications system.

The rest of this section presents a survey of some relevant topics of dependability in the context of real-time communications, such as replication and fail silence failure mode. These techniques will later (Sect. 8.4) be employed in the design and definition of a fault-tolerant architecture for infrastructure based vehicular networks.

8.3.1 Replication

One of the most common methods to build fault-tolerant distributed systems is to replicate subsystems that fail in an independent way. The goal of this approach is to give other subsystems the idea that the delivered service is provided by a single entity. In order to enforce consistency among the replicas, there are two main categories of replication schemes: active and passive replication.

Active replication, also called state machine approach [24], is characterized by having all the replicas receiving and processing the same sequence of requests in parallel, and producing the same output result at the end. In order to ensure that the state of these replicas is kept consistent, i.e., they will produce the same output, it has to be guaranteed that all of them are fed with the same inputs in the same order, typically by employing an atomic broadcast protocol to disseminate the commands [32]. These requests are handled independently but must be processed in a deterministic way.

In active replication, there is no need for an explicit recovery procedure when one of the replicas fails, since the other ones will continue to provide correct service. Therefore, this technique is simple and transparent to the clients in case of node failure. However there are some disadvantages with this approach. For example, the determinism constraint may be difficult to enforce (e.g. in a multithreaded node) and it is resource demanding, because the operation of each replica requires a full set of resources.

Passive replication, also known as primary backup [7], is more economic than the active scheme, because only one replica, the primary one, processes the commands

and produces the output results. The remaining replicas, called secondary or backup, remain in idle state and only interact with the primary to update and log all commands. When the primary system fails, the output result can not be delivered immediately with this technique. Instead, one of the backup replicas is selected as the new primary and it will resume operation from the last well known state of the system with the aid of the information available in the log repository. In this case, the client will time-out and resend the request after detecting the new primary replica. This significantly increases the response time of the system in case of failure, making this method unsuitable for some time sensitive applications. In passive replication, no determinism constraint is necessary but special care must be put on the mechanisms that enforce agreement between primary and backups.

Semi-active and semi-passive replication are two variants of the replication schemes referred above. The idea behind semi-active replication [21], also called leader-follower, is that the replicas do not need to process requests in a deterministic way. Only one replica (the leader) is responsible for making non-deterministic decisions and inform the followers of these choices. In semi-passive replication [9], client requests are received by all replicas and every replica delivers its responses back to the client. This way, the clients do not need to know the identity of the primary and there is no need for time-outs to detect the crash of the primary, being system failures completely masked to the client. Semi-passive replication is fully based on failure detectors and thus it does not require an agreement method (e.g. membership service) to select the primary replica. This reduces the response time in case of crash of the primary, when compared with the passive replication approach.

The earlier replication strategies proposed in the literature were mostly focused on applications for which real-time behaviour was not an essential requisite. However, real-time applications usually operate under stringent timing and dependability constraints and therefore, several replication strategies had been developed [15, 19, 33], in order to solve the additional challenges posed by safety-critical environments.

An important concept related with the design of replication protocols is the fail silence failure mode that will be described next.

8.3.2 Fail Silence Failure Mode

A faulty node of a distributed system that sends unsolicited messages at arbitrary points in time (babbling idiot failure mode [14]) without respecting the media access rules can disable nodes with legitimate messages to access the network. However, this failure mode can only occur if a node fails in an uncontrolled way. Network topologies that support the operation of fail uncontrolled nodes are costly [21]. Thus a node should only exhibit simple failure modes and ideally it should have just a single failure mode, the fail silent failure mode [27], i.e., it produces correct results or no results at all. In this matter, a node can be fail-silent in the time domain, i.e., transmissions occur at the right instants, only, or in the value domain, i.e., messages contain correct values, only. With fail silence behaviour, an error inside a node cannot

affect other nodes and thus each node becomes a different fault confinement region [27] (a specific part of the system where the immediate impact of a fault is limited only to that defined region). Furthermore, if k failures of a functional unit in a system must be tolerated, then $k + 1$ replicas of that unit are needed as long as they are fail silent. If the replicas are fail uncontrolled, then $3k + 1$ will be required [16]. Thus, the use of fail silent nodes also reduces the complexity of designing fault-tolerant systems.

The alternatives to enforce fail silent behaviour may be generically divided in two main groups. The ones that result from adding redundancy to each node and ones that rely on behavioural error detection techniques [27].

Using replicated processing within a node with output comparison or voting calls for the use of mechanisms to keep the replicas perfectly synchronized and to avoid replicas to diverge due, e.g. to asynchronous events. Synchronization at processor instruction level is the most obvious way to achieve replica synchronism, driving identical processors with the same clock source and evaluating their outputs (either comparing or voting) at critical instants, e.g. every bus access. Special care must be taken with asynchronous events that must be delivered to the processors so that all perceive the same event at the same point of their instruction streams.

Over the years, many systems were designed based in double-processor fail silent nodes such as Sequoia [4] and Stratus [30]. However, these systems have some drawbacks [6]. First of all, the processors must exhibit the same deterministic behaviour every clock cycle and don't care states are not allowed so that they produce identical outputs. Secondly, the use of special purpose hardware as comparators or voters, reliable clock sources and asynchronous event handlers greatly increases the design complexity. Finally, due to their operation in lock step, a transient fault could affect both processors in the same way, making the node susceptible to common mode failures. An alternative approach to eliminate the hardware level complexity of the solutions referred above is to transfer the replica synchronism to a higher level using software protocols over a set of standard processors operating independently of each other in a node. Task synchronization approaches were used in SIFT [31] and Voltan [25].

Behavioural error detection mechanisms, either in software or in hardware, are another alternative for enforcing fail-silence behaviour. Mechanisms such as checksums, watchdog timers and processor monitoring, are usually implemented using commercial-off-the-shelf (COTS) components. Error detection latency is the major bottleneck of these systems since the error detection mechanisms are only able to detect errors a relatively long time after they occur, possibly forcing other nodes to put in place some sort of error recovery policy.

8.3.2.1 Bus Guardians

Bus guardians (Fig. 8.2), which are autonomous devices with respect to the node network controller and host processor, also implement behavioural error detection mechanisms. Usually used in wired networks to enforce fail silence, bus guardians

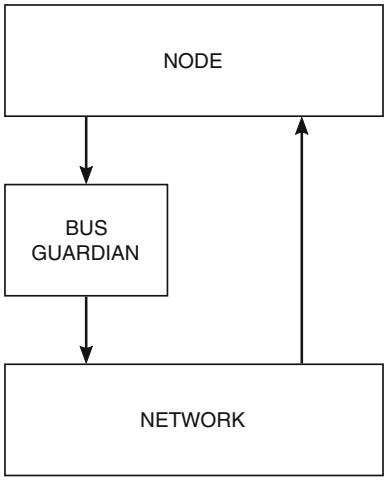

Fig. 8.2 Generic bus guardian scheme

act as failure mode converters, i.e., the failure modes of the component are, at the interface to other components, replaced by the failure modes of the guardian.

In order to be fail-independent with respect to the interface it monitors, the bus guardian must belong to a separate fault confinement region. A guardian would be of no use if it failed whenever the node that it is guarding also failed. Some potential sources of common mode failures are: clocks, CPU/hardware, power supply, protocol implementation, operating system, etc. Designing a bus guardian with independent hardware, with no common components and design diversity can help to avoid common failure modes. Despite the possible design compromises made between independence, fault coverage and simplicity/cost in any bus guardian architecture it is mandatory for the guardian to have some a priori knowledge of the timing behaviour of the node it is policing. In time-triggered (TDMA) networks this implies that each bus guardian needs to have its own copy of the schedule and an independent knowledge of the time.

Almost all the work developed in the area of fail silent systems is for wired industrial, or automotive systems [3, 28]. Although the principles are similar, nodes of wireless networks pose some additional problems implementing fail silent behaviour. For instance, the open nature of such networks, in contrast with the closed wired environments found in industrial and automotive systems, raises several issues to the identification of which and how many nodes are in the radio range of a given network at each moment in time. For the same reason, a number of other problems, like the hidden node, arise and need to be addressed. Besides, wireless communications systems are inherently half-duplex, i.e. a node is not able to transmit and receive at the same time, an important characteristic that must also be considered. Furthermore, the use of centralized bus guardians as in star topologies is not possible. However, in wireless communications, one can think in medium guardians as devices that protect non-faulty wireless network nodes from erroneous ones.

8.4 Achieving Fault-Tolerance in V-FTT

In master-slave networks, as V-FTT, the most obvious issue that must be dealt with, when addressing fault-tolerance and dependability, is the single point of failure formed by the master holding the traffic scheduler and vehicles registration database. In fact, if a master node (RSU) fails to transmit trigger messages with the EC-schedules, transmit them out of time or with erroneous contents, then all network activity could be seriously compromised or even disrupted.

This can be handled using replication, with one or more similar nodes acting as backup masters. In this way, as soon as a missing trigger message is detected, a backup master comes into the foreground and transmits it, maintaining the communication without any discontinuity of the traffic schedule. However, this is only possible if all master replicas are synchronized, with respect to value and time.

To ease the task of designing mechanisms that enforce masters replication, it is considered that nodes are fail-silent, i.e., nodes can only fail by not issuing any message to the network. This, however, must also be enforced by using adequate components as nodes can fail uncontrollably. These additional elements should guarantee that the messages sent to the medium by the RSUs are correct both in time and value domain.

After guaranteeing that all active RSUs exhibit fail-silence behaviour, an active replication scheme can be used to ensure that even in the presence of a failure in the primary RSU, the trigger messages with the EC-schedules will still be delivered to the OBU nodes. In this active scheme, the backup RSUs receive and process the exact same sequence of messages in parallel with the active one, and produce the same trigger messages in each EC. However, the packet transmission operation in the backup RSUs is deliberately delayed by a small amount of time, in the order of few μs, relatively to the primary one. This is done with the objective of facilitating the recovery procedure in case of RSU failure, by making it completely automatic and transparent to the slave nodes. This way, if the active RSU is able to transmit the trigger message in the planned instant, the replica will sense the wireless medium as occupied and will conclude that the primary system is free of error. Hence, it will not issue any message to the air, avoiding any overlap with the active node. On the other hand, if at that moment the medium is perceived as free, the replica will continue the transmission of its message and will replace the operation of the previously active RSU. As a result, the trigger messages will still be transmitted on time, since only a small delay is introduced at the beginning of the EC.

OBUs (slaves) should also exhibit fail-silence behaviour and although one could adopt the same mechanism used in master nodes, that would be expensive. The cost of that solution probably could not be supported by the vehicles' owners. Thus, slave nodes fail-silence enforcement both in time and value domain should only be adopted in special cases where the slave node information (value and timing) is absolutely essential, e.g. for police and emergency vehicles. For regular vehicles, a more inexpensive solution must be considered. Given the fact that slave nodes are not responsible for any type of network coordination, limiting OBUs ability to transmit uncontrollably will suffice. This corresponds to enforce fail-silence behaviour in the

time domain only. From the OBU's perspective, a schedule is valid only within the scope of an elementary cycle, thus an entity policing the node only needs to be aware of the node schedule in a EC by EC basis. This entity can be regarded as a medium guardian that avoids node's transmissions outside the time slot assigned to it by the masters of the network—the RSUs. For that, this medium guardian just needs to decode every trigger message contents and block any unscheduled transmission from the node.

8.4.1 Fault Hypothesis

Figure 8.3 presents the overall architecture of the network based on road side infrastructure. The fault-tolerance mechanisms are also depicted, giving rise to the following fault hypothesis:

- **Node faults**—Master nodes (RSUs) are assumed to exhibit fail-silence failure semantics, which is guaranteed by their internal redundancy and a fail silence enforcement entity that validates the agreement both in value and time domains. This mechanism will be explained in more detail in the following sections. Besides that, RSUs are also replicated (Backup RSUs), in order to undertake a failure in the active nodes (Active RSUs). In OBUs (slave nodes), medium guardians are used, to enforce fail silence only in time domain.
  In this scheme, the fail silence enforcement entity (as well as the medium guardian when considering OBUs) belongs to a fault containment region that is assumed independent of the one constituted by the remaining components of the node.

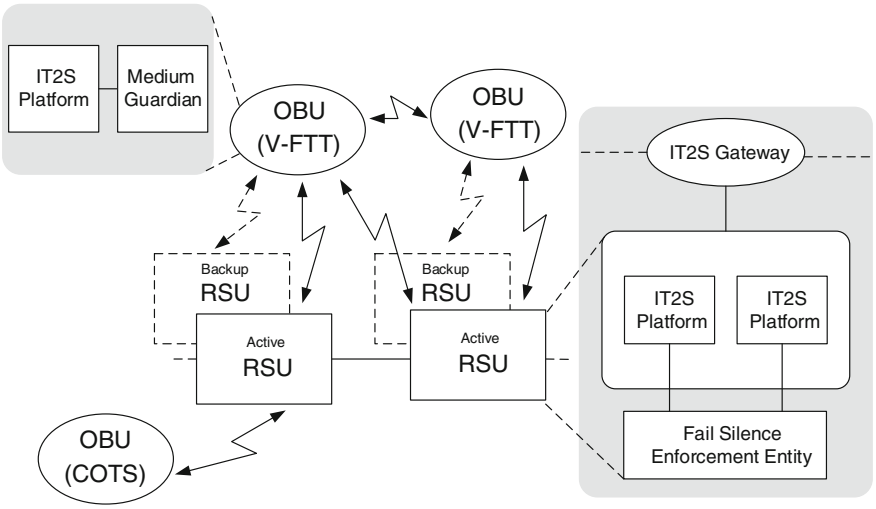

Fig. 8.3 Fault-tolerance mechanisms in V-FTT network

The covered faults within each node include hardware faults, both transient and permanent, and software faults. However, as it will be shown, faults in the analog part of the physical layer are not considered. Byzantine faults, notably intrusions, are not totally covered, since such kind of faults need to be handled also at the IT2S Gateway level, a component that provides communication with other RSUs through the backhauling network.

- **Channel transient faults**—Vehicular communications are regularly affected by transient faults, since the wireless channel conditions may vary, depending on atmospheric and traffic conditions. This effect is much larger than the one observed in communication protocols for wired environments. A way to circumvent the higher packet error rate is by introducing time and spatial redundancy in the Trigger Message transmission by the RSUs, as depicted in Fig. 8.1. This can be achieved by deploying a more dense RSU distribution along the road, leading to a partial overlap of the RSUs' radio coverage areas. This way, a single OBU transmission can be scheduled by a configurable number of adjacent RSUs, typically 3, for the same transmission slot. In this case, time and space diversity are employed together, since several RSUs placed in distinct locations transmit the same EC-schedule at different time instants. The required RSUs coordination for the execution of this redundancy mechanism is guaranteed through the backhauling network. It is thus assumed that every OBU receives at least one Trigger Message every elementary cycle, i.e., channel transient faults do not impact the trigger message transmission. In this RSU redundant scheme, space diversity is also attained for the transmission of OBU messages, since several RSUs can receive the same packet. Moreover, critical nodes could be assigned with several time slots to re-transmit their messages.
- **Channel permanent faults**—The transmission medium is a single point of failure for vehicular networks. Permanent faults may occur in the wireless medium, e.g. due to unregulated interference, however, such faults are not considered in this work. Depending on the severity of channel interference, medium redundancy could be included, by detecting the disturbance and commuting the operation of V-FTT protocol to another channel in the available frequency spectrum. As the band assigned for wireless vehicular communications is reserved by law, unregulated interference can be considered as malicious faults.
- **Synchrony assumptions**—Nodes synchronization, both masters and slaves, is ensured by a GPS receiver located at each one of the nodes. The accuracy provided by this system is typically below 333 μs (the maximum value used to determine if a device is synchronized to UTC or not [10]) and it must be sufficient to cope with the synchronization requirements of V-FTT. However, if by some reason, the GPS signal is not available or is not sufficiently accurate at some instant in time, another strategy could be used. The alternative relies on the fact that the trigger message transmitted by the master node, besides conveying the scheduling information, can also act as a synchronization mark to all network nodes. This way, master nodes are also time masters, and it is assumed that in between two consecutive trigger messages (typically 100 ms), the clock counters of each node do not diverge more than a negligible amount of time.

Based on this network architecture and associated fault hypothesis, the next sections discuss the design and implementation of one of the proposed mechanisms to increase dependability in infrastructure based vehicular networks: the fail silence behaviour of RSUs. A rationale with several possible design choices for the fail-silent RSU is presented, together with its implementation in dedicated hardware and some obtained experimental results.

8.5 Enforcing Fail-Silence in V-FTT

8.5.1 IT2S Platform: A Brief Description

The proposed system architecture takes advantage of a flexible implementation of an IEEE-WAVE/ETSI-ITS-G5 controller, the IT2S platform. This platform (Fig. 8.4) has been developed from scratch at Telecommunications Institute (Aveiro site) and it could either operate as an RSU or as an OBU. The IT2S platform is essentially constituted by three main modules: the Smartphone, the Single Board Computer and the IT2S board. The latter one implements the recent IEEE 802.11p standard, focused on the MAC and physical layers of the protocol stack. However, since the implementation of the MAC layer resides in a hardware/software partition co-design, only the low level functionalities are executed in the IT2S board by the FPGA. This sub-layer that comprises the time-critical and deterministic operations of the MAC scheme, is designated as Lower MAC (LMAC).

The physical layer is completely implemented in the IT2S board, being divided in two main parts: the Analog PHY and the Digital PHY. The Analog PHY is responsible for the signal processing operations in the analog domain, such as the up and down conversion from base-band to RF and vice-versa, respectively. On the other hand,

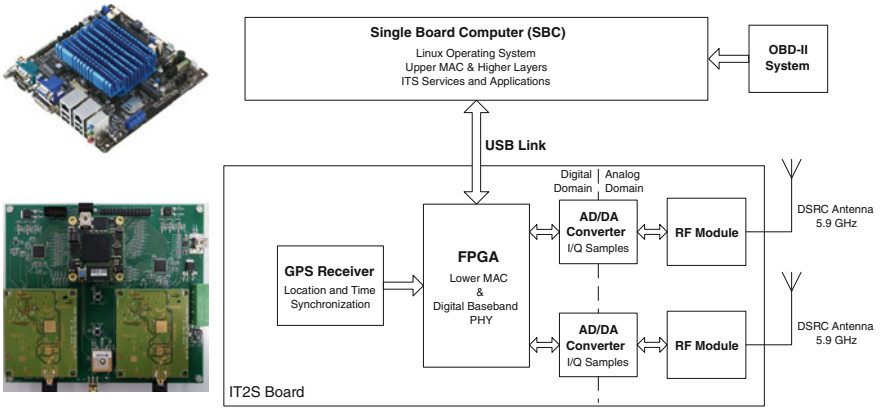

Fig. 8.4 Main blocks of the IT2S platform

the Digital PHY deals with signals in the digital domain, implementing the OFDM transmission and reception chains, converting bytes from a MAC frame into baseband In-phase and Quadrature (I/Q) samples and the reverse operation.

In order to cope with the simultaneous multi-channel operation requirement specified in the most recent versions of the standards [8], the board includes two complete sets of hardware units (2 DSRC antennas and RF modules, 2 AD/DA processors and 2 digital PHY and LMAC modules inside the FGPA) for the implementation of the IEEE 802.11p standard in both radios. The IT2S board also incorporates a GPS receiver for location and synchronization purposes. The interconnection with the SBC is established through an USB link and it is based on a time multiplexing scheme, allowing the co-existence of several independent channels for accessing each radio unit separately, retrieving information from the GPS device, performing updates on the FPGA bitstream, etc.

The main function of the Single Board Computer (SBC) is to execute the higher layers of the WAVE/ITS-G5 protocol stack, namely from the high level functionalities of the MAC layer, called Upper MAC (UMAC), to the Application layer. The SBC is a COTS embedded PC that runs a Linux-based operating system, providing high degree of flexibility and more control over the system's operation. When operating as an OBU, the SBC can also interact with the (On-Board Diagnostics) OBD-II system available in all recent vehicles. This way, it can access detailed information about vehicle's status and performance.

The Smartphone is responsible for implementing the graphical user interface that will provide more traffic related information to vehicle's driver and passengers. For instance, it will be able to display warnings in case of road accidents or traffic congestions. Another important advantage of the Smartphone is its capability to provide connectivity between the IT2S platform and a 3G or a 4G network. This feature allows the remote diagnostics and access to the information available in the platform, as well as a possible upgrade of the software and the reconfiguration of the bitstream in the FPGA. Finally, it could also enable the implementation of the eCall service, which basically consists in an automatic call to the 112 emergency number in the event of a serious road accident. As already referred, the IT2S platform can operate either as an RSU or an OBU. Obviously when the platform is working as an RSU, there is no need for a graphical user interface, and therefore the Smartphone is not included in the overall system's architecture.

8.5.2 Fail-Silent RSU Design

In order to implement a fail-silent RSU, all the possible device failure modes must be converted in fail-silent failure mode, enforced by a simpler, thus less prone to failures, component. The complexity of this fail silence enforcement entity can be greatly reduced if it is implemented at the lower layers of the OSI stack, in which the number of possible defects of system's design decrease and are easily identified [14, 22]. The design of the fail silence mechanism at the lower layers of the protocol

stack is simplified given the white box access to a flexible vehicular research platform. Implementing a similar solution in a COTS platform would probably imply higher latency, caused by the API.

As briefly discussed in Sect. 8.4, the operation of the proposed fail silence mechanism consists in an internal redundancy scheme, based on the replication of the IT2S platform (right side of Fig. 8.3). Two complete sets of single board computers and IT2S Boards are used to produce messages, whose purpose is to be disseminated through the air medium. The fail silence enforcement entity then compares the values and the timing of the messages produced by these two sets and, if everything is working correctly, it validates the frame and allows its transmission by one of the platforms. On the other hand, if the frames differ or are significantly out of phase, the entity silences the system until a restart signal is received.

Notwithstanding the decision to implement the fail silence mechanism at the lower layers of the protocol stack, there are still several design choices that should be taken when choosing the ideal place and method to perform the comparison of the samples produced by both platforms. As shown in Fig. 8.5, there are three main possible checkpoints where this verification can be executed, if one considers MAC and PHY as the appropriate layers to implement this mechanism. The rationale to choose only these lower levels is, as already explained, based on the observation that moving it higher on the protocol stack, will increase the design complexity and will potentiate design defects [22]. The three different checkpoints, numbered as 1, 2 and 3, correspond to the output interfaces of MAC, digital and analog PHY layers, respectively.

Output comparison at checkpoint number 1 could be attained using a voting scheme between both MAC frames produced at the output of Lower MAC sublayer

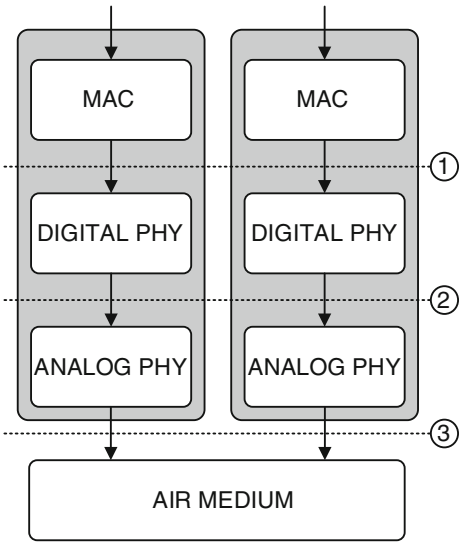

Fig. 8.5 Possible checkpoints for the fail silence mechanism

in the FPGA. The verification at this point, however, will not include possible faults in the operation of the digital physical layer, which basically comprises the baseband processing of the OFDM transmission chain. This could be attained if the fail silence mechanism is implemented at checkpoint 2, in the interface between the FPGA and the AD/DA processor, by comparing the I/Q samples that constitute the OFDM modulated frame. A verification at this stage already encompasses any discrepancy at higher levels of the IT2S platform, validating all the processing done at the SBC and at the FPGA. If, for instance, a software fault occurs at the higher layers, leading the SBCs to attempt transmitting different frames, e.g. due to distinct views of the network or inconsistent scheduling computation, the outputs at this checkpoint will differ and that fault can be detected. Nevertheless, in an ideal scenario, checkpoint number 3 would be the best place to verify the correctness of the outputs produced by both transmission chains, since it would also include the operation of the analog part of the IT2S platform. However, implementing an online verification algorithm of high frequency analog signals is a very complicated task, since signal comparison would probably require a downconversion to a lower frequency, which will add an excessive delay and will demand for resources in the analog part that may not be available. Furthermore, both transmission chains could produce correct signals although slightly different, due to minor mismatches on the digital to analog conversion and on the radio frequency amplification processes.

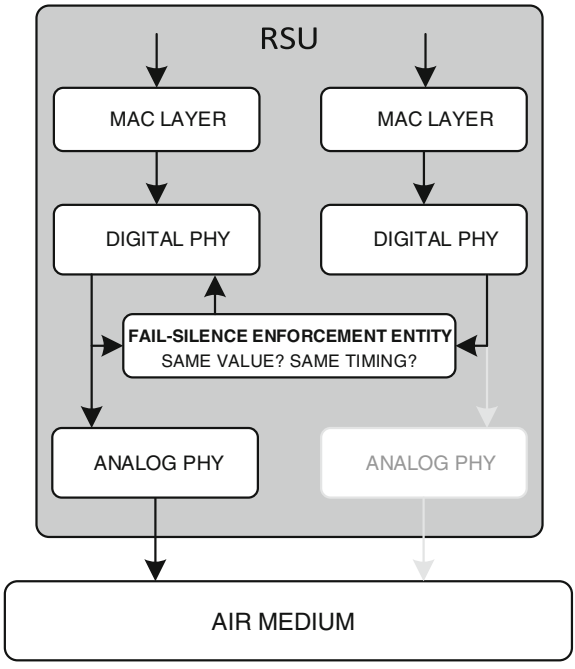

Fig. 8.6 Basic fail silent master RSU scheme

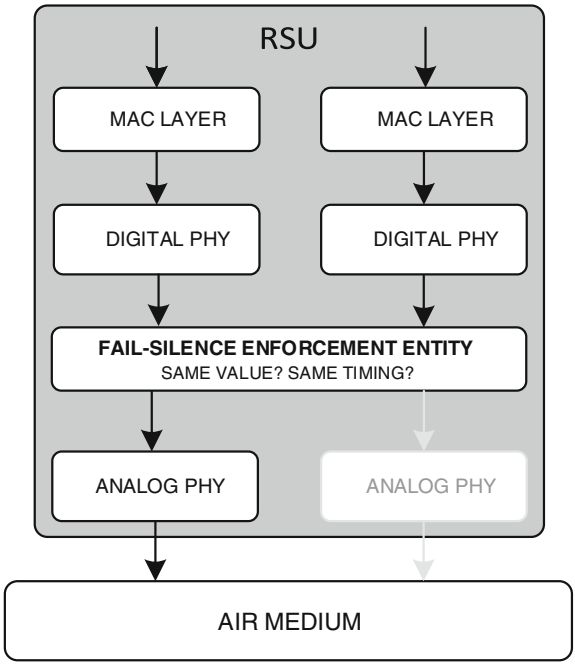

Fig. 8.7 Improved fail silent master RSU scheme

From the previous analysis, one can conclude that checkpoint 2 is the most appropriate place to implement the fail silence mechanism. A basic implementation, presented in Fig. 8.6, would be to perform a runtime comparison of the output produced by both digital PHYs and signal an error that would abort the ongoing transmission whenever a mismatch is detected. This method, however, would not prevent the medium from being occupied during at least part of an incorrect frame transmission, since a small tolerance must be allowed at the moment the results are produced. This time tolerance is needed to cope with slight variations between the internal clocks of both platforms that are synchronized through independent GPS receivers. As a consequence, the described solution would not result then in a true fail silent enforcement entity.

A possible improvement on the previous scheme is presented in Fig. 8.7, where each of the samples produced by the digital PHY is only allowed to proceed to the analog PHY after successful validation. This solution, although providing protection against frames that are completely different or out of time, still does not provide true fail silence behaviour, because a single error occurring at the middle of a message would invalidate the complete transmission. The medium could therefore be occupied with samples that although correct when analyzed independently, were invalid in the context of a full frame. Beyond that, given the fact that in the proposed real-time protocol, the RSU coordinates all the communications in the wireless channel, if an RSU transmission is interrupted, a complete elementary cycle will be wasted.

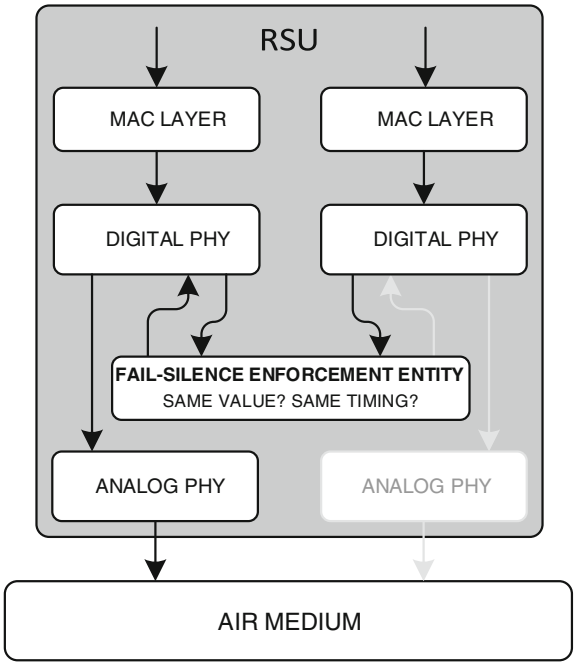

Fig. 8.8 Final fail silent master RSU scheme

A possible solution for this problem is to modify the transmission chain to produce the samples that are to be sent over the air in advance, relative to the moment when they are supposed to be sent. If enough advance is provided, the samples of an entire frame can be compared and validated before that frame transmission has even started. In this scheme, the validated samples are then fed back to the digital PHY that will, using a very simple mechanism, apply them at the correct moment to the analog PHY for transmission. If any difference is observed at any part of the frame or if the measured delay between the instants when samples are provided is greater than a certain fixed tolerance, the fail silent entity will not allow any of two units to transmit. In this case, the medium will not be occupied inadequately, contrarily to the previous proposals.

Based on these arguments, Fig. 8.8 presents the proposed scheme for the fail silent RSU and the final location of the fail silence enforcement entity inside the protocol stack. This entity was placed in the closest point possible to the antenna, at the end of the digital physical layer and before the analog part. This way, it is possible to validate the operation of the entire system and protocol, except the analog side of the physical layer. Ideally, the analog path should also be included, however and as already mentioned, implementing a runtime verification algorithm of high frequency analog signals is a very difficult task. Moreover, in order to guarantee that the fail silence enforcement entity belongs to a different fault containment region, with no common failure mode, it was developed in a external PCB with separated power supply and clock source.

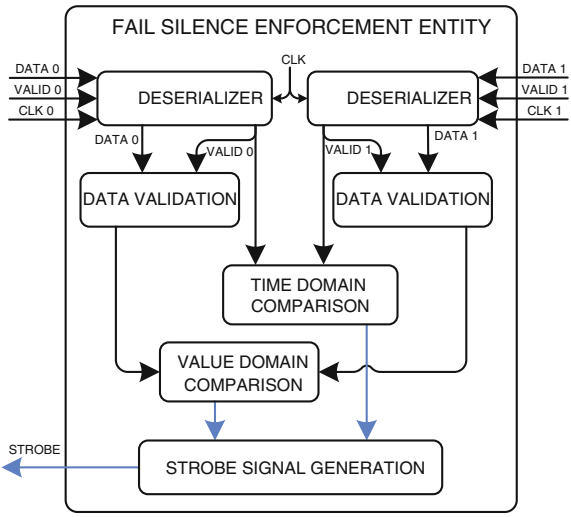

Fig. 8.9 Block diagram of the fail silence enforcement entity

In addition to this and as already referred, to truly implement a fail silent system, both transmission chains have to produce and send the frames to the fail silence enforcement entity in advance, so that the entire message could be verified before a single bit is transmitted through the antenna. More details regarding the internal architecture of the fail silence entity will be provided next.

8.5.3 Fail Silence Enforcement Entity

Figure 8.9 depicts the interfaces and the internal structure of the fail silence enforcement entity, which was implemented in a Xilinx FPGA Spartan-6. It receives the data generated by the digital transmission chains of the two IT2S Boards (10 bits of In-phase + 10 bits of Quadrature samples @10 Msps), which corresponds to signals DATA 0 and DATA 1. For each of these signals, there is a valid signal (VALID 0 or VALID 1) to indicate whether a frame is being transmitted or not in the data bus. After deserializing and removing the invalid bits, data from both sources of information are compared in the Value Domain Comparison module. Furthermore, the fail-silence entity also verifies if the two platforms are synchronized, i.e., producing information at the same time. It only allows a small time offset, to cope with the fact that the platforms may not be precisely aligned in time. However, this time tolerance should be small enough to guarantee the timely behaviour of the communications protocol (e.g. V-FTT). This operation is performed in the Time Domain Comparison module, based on the analysis of phase offset between the two valid signals.

For a given frame, if the results produced by these two comparison modules are both positive, a strobe signal is generated and sent to the primary platform (the one

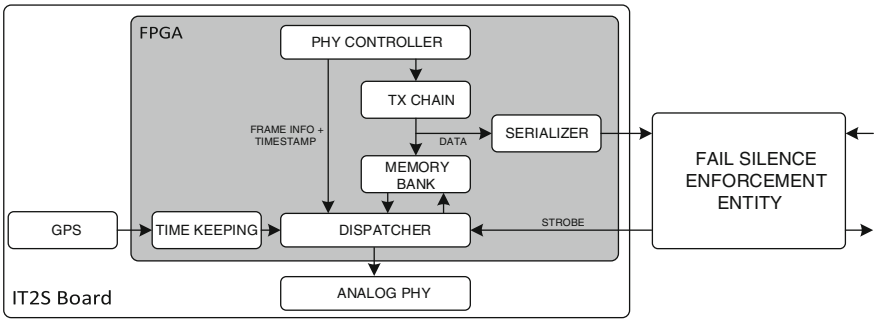

Fig. 8.10 Digital PHY scheme with fail silence behaviour

that will effectively send the message to the air). Otherwise, if at least one of the values is negative, no signal is generated and the entire RSU operation will stop until a restart signal is received in the fail silence enforcement entity.

The interaction of the IT2S Board with the fail silence enforcement entity is presented in Fig. 8.10. It depicts the architecture of the digital PHY in transmission mode and its integration with the fail silent scheme. A frame coming from the LMAC sublayer is handled in the PHY Controller module, which forwards its content to the digital transmission chain (OFDM modulator). Then, the resulting I/Q data samples are sent to the Serializer, in order to be verified by the fail silence enforcement entity, which will compare them with the ones provided by the other IT2S platform. The same samples are stored in a Memory Bank, waiting for the strobe signal to be sent to the wireless medium.

In case of successful frame validation, the Dispatcher module will receive the strobe indication and will start to prepare the transmission of the frame. Hence, it will compare the timestamp provided by the PHY Controller module, specifying in which moment the message should be sent, with the current time of the system, available from the GPS receiver. When both time values are equal, the Dispatcher reads the samples stored in the Memory Bank and send them to the Analog PHY. On the other hand, if the verification process performed by the fail silence entity fails, no strobe signal is received by the Dispatcher and consequently, no message will be issued to the wireless medium.

8.6 Experimental Evaluation and Results

The proposed architecture (Fig. 8.9) was successfully implemented in a Spartan-6 LX150 FPGA using the Trenz TE0300 carrier board. To validate the proposal, both comparison modules were tested. When two equal frames were sent with a phase offset below the maximum time tolerance limit, the strobe signal was successfully generated. However, when at least one bit error was introduced in one of the frames

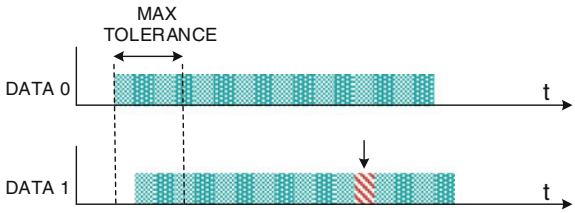

Fig. 8.11 Fault detection in value domain

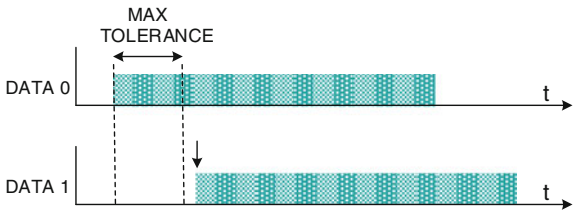

Fig. 8.12 Fault detection in time domain

(Fig. 8.11), the value domain comparison module was able to detect it, identifying a fault in the normal operation of the system. In this case, no strobe signal was generated and the fail silence entity entered into an idle state until a reset signal was received. This fault injection mechanism successfully tested the operation of the fail silence entity when the frames sent to validation were completely different as well as when there was only one error bit randomly inserted in a sample of one of the frames (as illustrated by the red striped mark in Fig. 8.11). The same result was achieved when a delay greater than the maximum tolerance allowed was introduced in the transmission of one of the frames (Fig. 8.12). In this situation, the time domain comparison module detected the excessive time difference between the beginning of the two frames and, similarly to the previous case, it signalled a fault to the strobe signal generation module.

As a proof of concept, the fail silence enforcement entity was developed in the same FPGA model that was used to implement the Lower MAC and the physical layer of the IT2S board. However, when comparing the resource usage of both projects, it can be concluded that the fail silence enforcement entity occupies much less resources (Table 8.1), around 1 % of the total available, than the project for the IT2S board (Table 8.2).

Table 8.1 Fail silence enforcement entity—resource usage on Spartan-6 FPGA

Logic resources	Used	Total	Percentage used (%)
Flip flops	314	184304	1
LUTs	284	92152	1
RAMB16BWERs	4	268	1
RAMB8BWERs	4	536	1
DSP48A1s	0	180	0
Logic max. frequency—160.805 MHz			

Table 8.2 IT2S board—resource usage on Spartan-6 FPGA

Logic resources	Used	Total	Percentage used (%)
Flip flops	39292	184304	21
LUTs	38111	92152	41
RAMB16BWERs	196	268	73
RAMB8BWERs	48	536	8
DSP48A1s	80	180	44
Logic max. frequency—158.188 MHz			

This result proves that the fail silence entity is much simpler than the entire vehicular communications platform and therefore, it is less prone to design errors and possible faults that can occur during system's operation. The simplicity of this unit is an extremely important characteristic that can be used to achieve a higher level of reliability, thus improving the dependability of the road-side infrastructure. Furthermore, in a future iteration of the fail silence enforcement entity, a smaller, less expensive FPGA, should be considered in order to reduce the implementation cost of this mechanism.

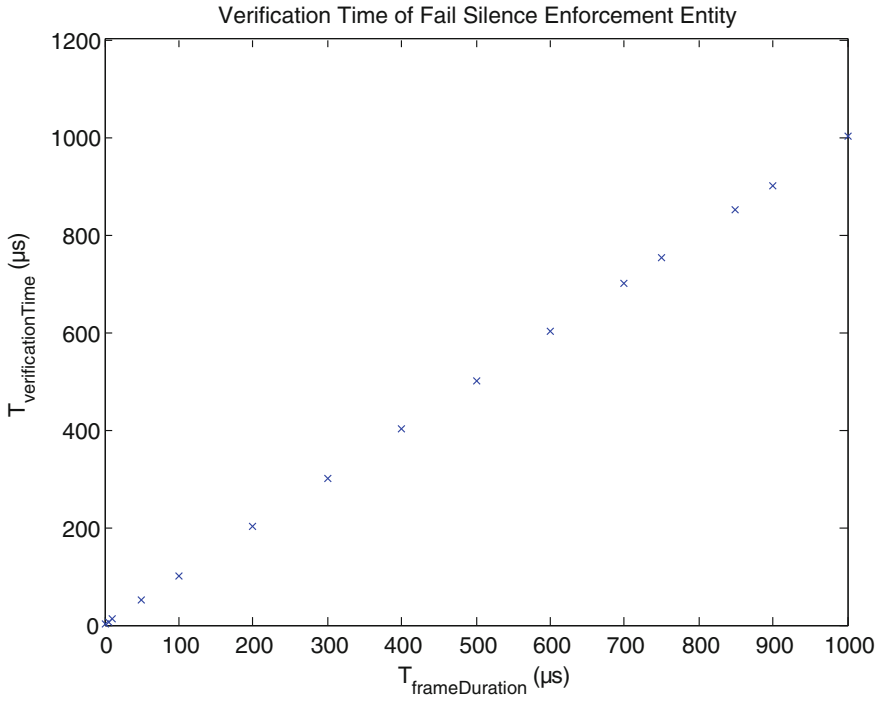

Fig. 8.13 Verification time as a function of frame duration

The frame verification time by the fail silence entity is another important aspect that should be considered when this mechanism is used to validate the operation of a deterministic wireless protocol, such as V-FTT. This verification time adds a delay to the beginning of the frame's transmission in the wireless medium, which should be taken into account by the real-time MAC protocol in use. For instance, this effect can be compensated if the frame is sent for validation with sufficient time in advance. Figure 8.13 shows the total delay introduced by the operation of the fail silence enforcement entity (FSEE) for various frame durations, from 1 µs to 1 ms. This verification time ($T_{verificationTime}$) is essentially constituted by two components (Eq. 8.1): the time that takes to transmit the entire message to the fail silence entity ($T_{frameDuration}$) and the delay introduced by the blocks inside the FPGA (T_{FSEE}). The latter one is constant and is approximately equal to 1.05 µs, while the first one is equal to the frame duration. Thus, the delay introduced inside the fail silence entity becomes negligible when the frame size increases, and in that case the frame verification time is approximately equal to the frame duration.

$$T_{verificationTime} = T_{frame\ Duration} + T_{FSEE} \tag{8.1}$$

Therefore, the samples should start to be sent to the fail silence enforcement entity, with a time advance ($T_{advance}$) greater than the one given by Eq. 8.2, so the message could be send to the air at the exact planned instant. The maximum tolerance allowed also contributes to this guard interval, since the verification time was measured when both platforms transmitted the same message at the same time, not considering a possible misalignment in the absolute clock sources.

$$T_{advance} \geq T_{verificationTime} + T_{maxTolerance} \tag{8.2}$$

This $T_{advance}$ time added to the normal operation of the system during the transmission of a message is completely acceptable, since the RSUs can start to prepare the packets for transmission with a large time advance. Given the fact that RSUs have a global vision of the road trough the backhauling network, they can easily compute the schedule of the next elementary cycle at the beginning of the current one. This represents an advance of approximately 100 ms (the total EC duration), which is more than enough to allow the inclusion of the fail silence mechanism in the operation of the system.

8.7 Conclusions

This chapter presented a fault tolerant architecture for infrastructure based vehicular networks. The rationale for designing dependable vehicular communication systems was explained together with the advantages of deploying networks that rely on the support provided by the road-side units. In addition to this, a brief overview of the MAC issues in dense vehicular scenarios was given, in order to explain the need for

real-time behaviour in the operation of the communications protocol. V-FTT protocol was presented as a possible solution to overcome the issues found in the previous proposals, and it was chosen as a case of study. As a result, the dependable vehicular network architecture was designed, by taking V-FTT protocol as an example of a deterministic medium access control scheme. Nevertheless, the proposed architecture is protocol independent and thus can be applied to any system based on the same principles.

Then, some techniques typically employed in the design of fault-tolerant systems were surveyed, leading to the definition of a fault-tolerant architecture and a fault hypothesis for V-FTT networks. As a consequence, several mechanisms were proposed namely, the replication of the road-side infrastructure, a fail silence enforcement entity for RSUs and medium guardians for OBUs. In the rest of the chapter, special attention was paid to the design of fail-silent RSUs, since fail silence behaviour in the master nodes of the network is an essential property that should be attained, in order to provide dependable network operation. The design choices behind the proposed fail silence enforcement entity were presented together with its final structure and respective integration with the operation of a custom vehicular communication station, the IT2S platform.

In summary, the fail-silence mechanism compares the output messages produced by two vehicular communications platforms, both in value and time domain. Ideally, these two systems should produce the same outputs based on completely different hardware and software implementations. This should be done to avoid common mode failures. However, as a proof of concept, two identical IT2S platforms were used. The obtained results show that the developed mechanism successfully detects system faults both in time and value domain. The mechanism can be implemented in an FPGA with few resources, but it should belong to a separate fault confinement region with different power supply and clock source. Moreover, the delay introduced by the operation of the fail silence entity could be significant for large frame sizes, so it should be taking into consideration when analyzing the whole performance of the wireless communications protocol.

Acknowledgments This work is funded by National Funds through FCT—Fundação para a Ciência e a Tecnologia under the Ph.D. scholarship ref. SFRH/BD/52591/2014, by the European Union's Seventh Framework Programme (FP7) under grant agreement n. 3176711 and by BRISA, under research contract with Instituto de Telecomunicações—Aveiro.

References

1. A. Avizienis et al., Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
2. G. Bansal, J.B. Kenney, Controlling congestion in safety-message transmissions: a philosophy for vehicular DSRC systems. IEEE Veh. Technol. Mag. **8**(4), 20–26 (2013). ISSN: 1556–6072. doi:10.1109/MVT.2013.2281675
3. R. Belschner et al., FlexRay requirements specification, version 2.0.2. FlexRay Consortium (2002). http://www.flexray-group.com

4. P. Bernstein, Sequoia: a fault-tolerant tightly coupled multiprocessor for transaction processing. IEEE Comput. **21**(2), 37–45 (1988)
5. A. Böhm, M. Jonsson, *Real time communications support for cooperative, infrastructure-based traffic safety applications* (Int. J. Veh, Technol, 2011)
6. F.V. Brasileiro et al., Implementing fail-silent nodes for distributed systems. IEEE Trans. Comput. **45**(11), 1226–1238 (1996). ISSN: 0018–9340. doi:10.1109/12.544479. http://dx.doi.org/10.1109/12.544479
7. N. Budhiraja, K. Marzullo, F.B. Schneider, S. Toueg, in Distributed Systems, 2nd edn., ed. by S. Mullender. The Primary-backup Approach (ACM Press/Addison-Wesley Publishing Co., New York, 1993) pp. 199–216
8. C. Campolo, A. Molinaro, Multichannel communications in vehicular ad hoc networks: a survey. IEEE Commun. Mag. **51**(5) 158–169 (2013). ISSN: 0163–6804. doi:10.1109/MCOM.2013.6515061
9. X. Défago, A. Schiper, N. Sergent, Semi-passive replication, in Proceedings of the The 17th IEEE Symposium on Reliable Distributed Systems, SRDS '98 (IEEE Computer Society, Washington, 1998) pp. 43–50
10. IEEE Standard for Wireless Access in Vehicular Environments (WAVE), Multi-channel Operation, in IEEE Std 1609.4-2010 (Revision of IEEE Std 1609.4-2006) (2011), pp. 1–89. doi:10.1109/IEEESTD.2011.5712769
11. Intelligent Transport Systems (ITS), Performance evaluation of self-organizing TDMA as medium access control method applied to ITS; access layer part, in ETSI TR 102 862 V1.1.1, Dec 2011, pp. 1–51
12. J.B. Kenney, Dedicated short-range communications (DSRC) standards in the United States, in Proceedings of the IEEE 99.7, July 2011, pp. 1162–1182. ISSN: 0018–9219. doi:10.1109/JPROC.2011.2132790
13. S. Khan, P. Pedreiras, J. Ferreira, Improved real-time communication infrastructure for ITS, in INForum. Sept **2014**, 430–445 (2014)
14. H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications. (Kluwer Academic Press, 1997)
15. H. Kopetz, G. Grunsteidl, TTP-a protocol for fault-tolerant realtime systems. Computer **27**(1), 14–23 (1994)
16. L. Lamport, R. Shostak, M. Pease, The byzantine generals problems. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
17. N. Lu et al., A distributed reliable multi-channel MAC protocol for vehicular ad hoc networks, in Intelligent Vehicles Symposium, 2009 IEEE, June 2009, pp. 1078–1082. doi:10.1109/IVS.2009.5164431
18. T.K. Mak, K.P. Laberteaux, R. Sengupta, A multi-channel VANET providing concurrent safety and commercial services, in Proceedings of the 2Nd ACM International Workshop on Vehicular Ad Hoc Networks, VANET '05 (ACM, Cologne, 2005), pp. 1–9. doi:10.1145/1080754.1080756. http://doi.acm.org/10.1145/1080754.1080756
19. A. Mehra, J. Rexford, F. Jahanian, Design and evaluation of a windowconsistent replication service. IEEE Trans. Comput. **46**(9), 986–996 (1997)
20. V. Milanes et al., An intelligent V2I-based traffic management system. Intell. IEEE Trans. Transp. Syst. **13**(1), 49–58 (2012). ISSN: 1524–9050. doi:10.1109/TITS.2011.2178839
21. D. Powell, Delta-4–A generic Architecture for Dependable Distributed Computing (ESPRIT Research Reports, 1991)
22. J. Proenza, J. Miro-Julia, *MajorCAN: a modification to the controller area network to achieve atomic broadcast, in IEEE International Workshop on Group Communication and Computations* (Taipei, Taiwan, 2000)
23. J. Rezgui, S. Cherkaoui, O. Chakroun, Deterministic access for DSRC/802.11p vehicular safety communication, in Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International, July 2011, pp. 595–600
24. F.B. Schneider, Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Comput. Surv. **22**(4), 299–319 (1990)

25. S. Shrivastava et al., Principal features of the voltan family of reliable node architectures for distributed systems, in IEEE Trans. Compute. (Special Issue on Fault-Tolerant Computing) **41**(5), 542–549 (1992)
26. T. Meireles, J. Fonseca, J. Ferreira, The case for wireless vehicular communications supported by roadside infrastructure, in Intelligent Transportation Systems Technologies and Applications (John Wiley and Sons, 2014)
27. C. Temple, Avoiding the babbling-idiot failure in a time-triggered communication system, in Fault Tolerant Computing Symposium (IEEE Computer Society, 1998), pp. 218–227
28. TTTech, Time-Triggered Protocol TTP/C High-Level Specification Document, 1.0 edn. (2002). http://www.ttagroup.org
29. P. Veríssimo, Uncertainty and predictability: can they be reconciled?, in Future Directions in Distributed Computing LNCS 2584 (Springer-Verlag, May 2003)
30. S. Webber, J. Beirne, The stratus architecture, in Digest of Papers FTCS-21 (1991), pp. 79–85
31. J. Wensley et al., SIFT: design and analysis of a fault tolerant computer for aircraft control. Proceedings of IEEE **66**(10), 1240–1255 (1978)
32. M. Wiesmann et al., Understanding replication in databases and distributed systems, in 20th International Conference on Distributed Computing Systems. Proceedings, (2000), pp. 464–474
33. H. Zou, F. Jahanian, A real-time primary-backup replication service. IEEE Trans. Parall. Distrib. Syst. **10**(6), 533–548 (1999)