Received March 8, 2022, accepted March 28, 2022, date of publication April 26, 2022, date of current version May 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3170102

A Flexible Communication Protocol With Guaranteed Determinism for Distributed, Safety-Critical Real-Time Systems

FAWAD RIASAT RAJA, DAVID CHEN, AND RENÉ HEXEL, (Senior Member, IEEE)
Institute for Integrated and Intelligent Systems, School of Information and Communication Technology, Griffith University, Nathan, QLD 4111, Australia
Corresponding author: David Chen (david.chen@griffith.edu.au)

This work was supported by the Machine Intelligence and Pattern Analysis Laboratory (MiPAL), Institute for Integrated and Intelligent Systems, Griffith University, Australia.

ABSTRACT Dependable, safety-critical real-time (SCRT) systems are becoming increasingly important and complex. Examples of such systems are autonomous or self-driving cars which are poised to revolutionise the transport industry. A critical part of these SCRT systems is the network communication protocol that is used by components in an SCRT system to exchange data. Communication protocols for SCRT systems are required to exhibit predictable, worst-case execution times and thus have to be designed in a more static and less flexible way. To ensure this predictability, current state-of-art communication protocols for SCRT systems are based on the Time-Triggered Architecture (TTA), where static and equal-length time-slots are used for all nodes to access the communication channel, irrespective of the size of their transmission payload. This determinism forms the basis of predictable timing, behaviour and fault tolerance. However, this determinism comes at the cost of poor channel and bandwidth utilisation, which hinders the development of SCRT systems. In this paper, we propose a more flexible approach, INCUS+, that allocates the slot length of a node based on its transmission requirements in a Time Division Multiple Access (TDMA) round. We achieve this while retaining the level of dependability required for SCRT systems and ensuring fail-silence. We validate this through formal verification of the timing parameters for the transmission windows of all participating nodes as well as independent bus guardians. Our design exhibits a significant improvement in bandwidth and channel utilisation, as we demonstrate in an autonomous vehicle case study.

INDEX TERMS Event-triggered communication, fault tolerance, flexibility, in-vehicle networks, real-time communication, safety-critical systems, time sensitive networking, time-triggered communication.

I. INTRODUCTION

Flexibility and dependability are the two parameters that are often considered as contrary to each other and choosing between them to solve an engineering problem is a hard task [1], [2]. There is a strong argument in the literature that to achieve and verify dependability, static prior knowledge about the sequence and timing of state changes is essential [3]. This idea has been exploited by the Time-Triggered Architecture (TTA) [4], [5] to produce more dependable real-time systems [6]. The TTA is a composable architecture that is used to design large real-time systems

The associate editor coordinating the review of this manuscript and approving it for publication was Filbert Juwono.

In TTA-based communication, permission to use a communication channel is determined by a Time Division Multiple Access (TDMA) scheme with predefined time slots. The most prominent example of such communication in real-time systems is the Time-Triggered Protocol (TTP) [3], [7]. To ensure collision-free channel access, all nodes are assigned time slots to transmit their information. All nodes have synchronised clocks, forming a global time base, and thus, they know the exact point in time each node will transmit its message. All static time slots are of equal duration and are distributed between participating nodes in equal-length TDMA rounds. These TDMA rounds repeat indefinitely, making channel access periodic.

This predictability ensures that channel access will be inherently free of collision and all the protocols that follow this approach will have known channel access latency. However, this comes at a cost. In a system, it is likely that different nodes have very different transmission requirements for their transmission payload. Therefore, they need different-length time slots, and assigning equal-length time slots will result in poor channel utilisation (CU).

While attempts have been made in the literature [8]–[14] to overcome this issue, we are not aware of any approaches that completely solve this problem while retaining the requirements that are imperative for SCRT systems, such as fault tolerance and guaranteed timeliness. In fact, this problem has long been recognised and attempts exist to split up communication and, at least, provide some flexibility and better channel utilisation for non-essential messages, while retaining the strict real-time guarantees for safety-critical communication [15].

Let us investigate different use-cases that exhibit the need for different transmission slot lengths during different TDMA rounds of a cluster cycle. Nowadays, Advanced Driver Assistance Systems (ADAS) are becoming common in modern vehicles. While this was originally considered under the Telematics domain, it has, over time, been accreting more and more new features, to the effect that the ADAS is considered as a separate domain [16]. These systems are also communicating with in-vehicle safety networks to provide autonomous operations for the self-driving vehicle. ADAS relies on videos collected from multiple cameras mounted on the vehicle. Compressed videos from the camera are sent to ADAS through the in-vehicle safety networks. Compression techniques such as H.264 produce frames of different data length and hence, a camera node may need a different transmission slot length in each TDMA round. Number of similar scenarios are discussed in the literature such as [17] where stream sender is using H.264 based compression technique for low latency video streaming for autonomous cars. Similarly, some approaches [18] are using TDMA based techniques to transmit compressed video frames but using multiple and equal length time slots to transmit Intra-coded and consequently, results in poor channel utilisation.

Another use-case is Unmanned Aerial Vehicles (UAVs). These are increasingly used for numerous applications, including surveillance, exploring and tracking targets [19]. For example, UAVs are in high demand for the inspection of large scale structures as well as search and rescue operations in a disaster area. One or more UAVs are used to transmit live video from an area of interest to a ground station where an operator can adjust the position of UAVs after analysing the streaming video [19]. In remote areas, multi-hop wireless networks are created where a number of UAVs are used as relays to extend the range. Each relay (a UAV in this case) forwards a received packet to the next hop closer to the sink node. This transmission is carried out by using a native wireless CSMA/CA arbitration scheme. Transient asymmetries between the relay links lead to unbounded packet buffering which further creates longer queuing delays and buffer overflow resulting in packet losses. To overcome this issue, Pinto *et al.* [19] proposed a TDMA-based approach on top of standard WiFi, where an adaptive slot length for each relay node in every round mitigates the issue of unbounded queuing delays and reduces packet losses. However, this approach was designed for soft real-time systems and does not provide any fault tolerance guarantees.

The Internet-of-Things (IoT) is a new paradigm that is used to connect surrounding physical objects containing sensor or actuator nodes in order to operate them remotely via the Internet [20]. This concept is used in a number of applications from different domains such as healthcare, smart homes, etc. Healthcare applications deployed in smart homes may consist of sensor devices to monitor the vital signs of a patient, alerting family members or physicians in an emergency situation [20]. Efficient communication is one of the biggest challenges in IoT as data networking is used to collect information (e.g. vital signs) from sensor devices. Researchers have attempted to improve communication efficiency for such systems. Saxena [20] proposed a context-aware, adaptive, forwarding (Cdf) strategy to transmit critical, health-related data, even in poor network conditions. For example, if patient's vital signs are consistently good, the acquisition interval increases, which reduces the transmission rate of packets, otherwise, more frequent data transmission is required. However, their approach to facilitate efficient data transmission is opportunistic [20], utilising a rules-based best-effort approach.

The objective of the present work is to overcome the issue of poor channel utilisation with the determinism required for predicability and fault tolerance in SCRT systems. Section II discusses established communication protocols for SCRT systems along with their properties. The principle of operation of our proposed protocol is discussed in Section III. We formally verify in Section IV the timing parameters required. In Section V, we examine the behaviour of our protocol under different faults and how these faults will be handled by the protocol in both the time and value domain by using the fault-tolerant services at the protocol level. Section VI discusses our approach in light of the need for *configurational flexibility* for slot allocation in TTA-based communication protocols, exemplified by a representative case study. In Section VII a computational model is presented, comparing different slot allocation approaches. We benchmark our approach and show how flexibility in slot allocation can significantly improve the channel and bandwidth utilisation. We present our conclusions in Section VIII.

II. RELATED WORK

In this section, we will discuss the existing state of the art in communication protocols for SCRT systems. ARINC-629 [21] is a communication protocol designed specifically for avionics systems [22]. The Boeing 777, for example, is using ARINC-629 for control and many related, safety-critical functions [23]. ARINC-629 utilises a data bus that is bidirectional and allows multiple access for transmitting safety-critical information (SCI) and non-safety-critical information (NSCI). Channel access in ARINC-629 is loosely based on time slots and it is regulated by transmission gap (TG) timer. The protocol is using a collision avoidance approach (CSMA/CA) for bus arbitration. The value of the TG is different for each node in order to prevent simultaneous channel access by different nodes. A node starts listening on the communication channel and once its TG has elapsed, it starts transmitting its messages only if the channel is idle. A node cannot transmit if its TG has elapsed but there is traffic on the channel. In this case the timer and the whole procedure restart afresh. ARINC-629 is similar to other approaches [24], [25] that are transmitting periodic and sporadic information. However, a timing analysis of ARINC-629 shows that the protocol supports periodic and sporadic traffic with deadlines, provided that the worst case sporadic traffic in the system is known [26].

Unfortunately, ARINC-629 does not prevent starvation; if a single node (often termed a *babbling idiot*) continually transmits, then other nodes will keep waiting indefinitely as the channel will never become idle. Furthermore, fault tolerance is not handled at protocol level: it is assumed that all such concerns can be handled at application level.

The predictable nature of the Time Triggered Architecture (TTA) [4], by contrast, provides a solid base to implement reliable and fault-tolerant distributed real-time systems. In the TTA, communication planning needs to be performed simultaneously for all nodes in a cluster. The communication schedule has two tightly coupled steps, message transfer and data elements. These data elements are the input/output data items that are consumed or produced by an application process within a node [27]. There is a specified validity time for every data element. A data element is called a phase-insensitive data element if the validity time is longer than the longest time interval between the point of observation and the point of use of the corresponding data element. Otherwise, the data element is phase sensitive. Application tasks receiving phase-sensitive data elements must be synchronised with the sending task. Otherwise, a state estimation task must be executed at the receiver [27]. Therefore, phase-insensitive data elements are preferred [5], [27], [28] as they make the system more resilient and less tightly coupled.

The Time-Triggered Protocol (TTP/C) [29] implements TTA communication by using a fixed, TDMA-based channel access scheme, where all nodes are allocated static and equal-length time slots. The scheduling of time slots is done offline and all nodes know the exact time of transmission and reception of data. The protocol is designed to provide reliable communication in accordance with a defined fault hypothesis, reducing the different kind of possible faults, such as omission failure, channel failure, and crash failure, that need to be tackled. Fault tolerance can be guaranteed through straightforward replication, while the deterministic timing allows independent bus guardians to be used to prevent babbling idiot faults [29]. A fault-tolerant, distributed clock synchronisation mechanism is used to form a global time base that ensures correct timing for all the nodes in TTP. However, this approach of static and equal-length time slots for all the nodes comes at the cost of overhead and flexibility. The lack of an ability to have different-length time slots for different nodes, in accordance with their payload requirements, results in poor channel utilisation.

FlexRay [15] is a hard real-time communication protocol for vehicular networks. The fundamental mechanism of channel access in FlexRay is based on a TDMA approach and uses the same principal mechanism as TTP. However, FlexRay adds flexibility over TTP with allocation of a time span in each TDMA round to transmit NSCI after sending SCI. Therefore, each TDMA round in FlexRay is divided into two parts. The first part is used to transmit SCI where all nodes have static and equal-length slots as per TTP, while the second part is used to send NSCI with dynamic slots similar to the ByteFlight protocol [30]. To ensure better channel utilisation, on-demand bandwidth sharing is allowed among nodes for NSCI. However, no channel access, timing, and fault-tolerance guarantees exist for nodes attempting to transmit in these dynamic slots. The fault-tolerance mechanism in FlexRay is similar to TTP for the first part of communication which transmits SCI. Consequently, the lack of flexibility (static and equal-length time slots) makes channel utilisation in FlexRay equally as bad as TTP for core, safety-critical communication. To improve the network utilisation in FlexRay, an algorithm is proposed in [31] to obtain the optimal length of static messages. Those messages that are longer than optimal-length are migrated to the dynamic segment of FlexRay. However, shifting safety-critical messages to the dynamic segment compromises the reliability of the protocol. A heuristic algorithm is proposed in [32] to efficiently utilise the bandwidth of the FlexRay network. The basic idea was to independently utilise both channels in FlexRay. Two modes have been considered, *independent* mode and *fault-tolerant* mode. It is assumed that some frames in the static segment do not need fault tolerance and hence, the independent mode can be used to send different messages on both channels in a given time slot. This idea contradicts the basic theme of a static and dynamic segment in FlexRay and therefore does not provide an optimal solution if all frames scheduled in static segment require fault-tolerance. The work of Lee *et al.* [33] to avoid transient failures in FlexRay by introducing retransmission of frames in the static segment further reduces the bandwidth utilisation and is prone to replicate communication errors. Similarly, other recent work such as [34] only addresses the issue of computation of end-to-end delay for the messages that are scheduled with slot-multiplexing in the dynamic segment of FlexRay.

Time-Triggered CAN (TTCAN) [35] was developed on top of the physical layer of the widely used, event-triggered CAN protocol. The idea was to develop a flexible, hybrid protocol that can transmit time-triggered as well as event triggered messages. The protocol uses an exclusive window to transmit a safety-critical message that needs a guaranteed latency. Unlike the original CAN protocol, safety-critical messages are transmitted at specific points in time (by using exclusive windows) and do not need to compete for bus access with messages transmitted using the CAN arbitration protocol. The system matrix of TTCAN consists of a number of basic cycles and it allows TTCAN to choose multiple sending patterns, e.g., transmit a message once per basic cycle, once in a whole matrix cycle, etc. A master node concept is used to synchronise clocks of all participating nodes, but this mechanism can add a significant delay in choosing a new master node in case of active master node failure. TTCAN does not provide important dependability services at protocol level such as membership, independent bus guardians, reliable acknowledgment, or similar. Some of these services can be build at application level, but at the expense of the efficiency of the protocol and timing bounds.

Another flexible approach is TDMA with slot-skipping (TDMA/SS) [9] to improve channel utilisation. The basic concept of TDMA/SS is to skip the transmission slot of a node if it does not start sending within a predefined time in its slot. The next node is permitted to send data before the scheduled time. Channel utilisation can be improved by using this and other similar approaches later [10]. However, these approaches are unsuitable for fault-tolerant, SCRT systems, as its flexibility compromises the determinism inherent in the distributed agreement achieved by the static schedule and fault-tolerant clock synchronisation, which is a basic requirement of fault-tolerance in the TTA.

TTEthernet [36] was developed to enable time-critical real-time traffic over a standard Ethernet network. It supports three classes of traffic, Time Triggered (TT), Rate Constrained (RC), and Best Effort (BE). The schedule is computed offline for TT traffic, hence guarantees contention-free communication over the same network [36]. A transmitter node sends TT messages in pre-defined, static time slots in order to avoid collisions, however these slots are distributed over equal-size communication cycles, repeating indefinitely. In TTEthernet, end systems (nodes) are connected through switches. Flows (frames) can be transmitted from one end system to multiple end systems through these switches. TT frames are periodically transmitted in pre-assigned time slots [36]. A transmission slot from one node, say Node A to Node C through switch 1 may be different in duration than the time slot from Node B to Node C through the same switch but remains the same (static and equal length) along the same path. Therefore, each node in the TDMA round may have a different slot length and the TDMA round cyclically repeats. This means that the length of TDMA rounds across the cluster cycle are required to be the same. Therefore, TTEthernet does not cover more dynamic scenarios where a node requires different slot lengths in different TDMA rounds. Moreover, fault tolerance is achieved through redundancy management and does not cover all fault scenarios at protocol level, missing important services such as membership, implicit acknowledgment, overhead-free, fault-tolerant clock synchronisation, clique avoidance and the like. Similarly, the bus guardian mechanism does not support variable slot lengths for the same node over different TDMA rounds.

The Audio/Video Bridging (AVB) Task Group developed a set of protocols to support deterministic communication of audio/video (AV) streams over a standard Ethernet network. Despite the advantages of standard Ethernet such as high bandwidth and low cost, it does not provide temporal properties that are essential for real-time traffic. The AVB Task group introduced a set of standards such as 802.1As [37], 802.1Qat [38], 802.1Qav [39], and 802.1BA [40] to support low latency and jitter requirements for multimedia streams. Time synchronisation is supported by the 802.1AS standard, which is based on the IEEE1588 Precision Time Protocol (PTP). The Stream Reservation Protocol (SRP), also known as 802.1QAT is utilised to reserve the bandwidth for high-priority traffic classes, while 802.1QAV supports a queuing and forwarding policy for AV traffic [41]. AVB was introduced to provide low latency and jitter for AV traffic, by reserving bandwidth along the whole path from transmitter to receiver. Despite its success and widespread use in the automotive industry, AVB fails to provide the real-time capabilities to support the rigid timing requirements of hard real-time applications [42].

Improvements have been made to IEEE AVB standards by the Time Sensitive Networking (TSN) Task group [43] in order to support real-time capabilities and performance improvements. TSN introduces different standards that are built on top of the AVB standards. The Credit-based Shaping (CBS) algorithm used by the IEEE AVB 802.1Qav standard does not support timing requirements of TT streams of AV traffic when non real-time traffic is in transmitted over the same channel. CBS is used to overcome the issue of starvation for low priority traffic but due to its non-preemptive nature, a low priority AV stream can block the transmission of time-critical AV streams. TSN introduced the Time-Aware Shaper (TAS) in IEEE 802.1Qbv [44] to resolve this issue. TAS adopts a preemptive approach where scheduled traffic can preempt low-priority traffic to fulfil its timing requirements. TSN defines high-priority queues for TT traffic while the rest of the queues are same as used in AVB. The traffic that does not require strict temporal properties is categorised as best-effort and assigned the least priority [45]. TSN is using IEEE 802.1AS-Rev [46] to synchronise the clocks to form a global time base for enabling deterministic communication. However, this mechanism comes at the cost of extra overhead (synchronisation frames) in addition to normal traffic over the communication network. TSN uses the concept of a Gate Control List (GCL) which is implemented on the egress ports of each participating device in the network. Each port can have multiple queues, where some of the queues are assigned to TT traffic and the rest of the queues are assigned to other traffic types such as AV or BE traffic. GCLs are computed offline and at each egress port, frames will be transmitted from a queue whose gates are opened. When gates for TT queues are opened the gates for other queues must be blocked. It is to be noted that when a TT queue has multiple frames and its gate is opened based on GCL then a FIFO mechanism is used to transmit the frames from the same queue [47]. This may lead to unavoidable delays in transmitting safety-critical information over the network as fragmenting a flow into numerous frames and adding sequence numbers need extra time. The complexity increases when there are multiple hops between end systems. A GCL scheduled for a priority queue defines the exact interval when that queue has exclusive access to the transmission channel. Interleaving frames from different TT flows to the same priority queue can significantly increase end-to-end transmission delays [14]. To avoid arbitrary transmission of TT frames and handling of babbling idiot faults, TSN introduces time-based ingress policing in IEEE 802.1Qci [48]. A time-aware Access Control List (ACL) is used to keep track of the arrival time of incoming TT frames. The ACL is computed offline and must be aligned with GCLs. A TT frame can be transmitted successfully only if the ACL grants permission to pass at the ingress port and at the same time the GCL has an active transmission time slot at the egress port. The GCL period is computed offline for a TT flow and repeats in cycles, which means the GCL has a static and equal-length time period for a TT flow as used in [14]. TSN introduces reliability and fault tolerance by using IEEE 802.1CB [49]. A transmitter or any intermediate device such as a switch will generate sequence numbers for all frames and multiple copies are generated for each frame before transmitting them over the network. Therefore, to identify and control the individual streams or flows, additional mechanisms such as Per-Stream Filtering and Policing [50] as well as Frame Replication and Elimination for Reliability [51] are required. To avoid a single point of failure, redundant routes are configured by using IEEE 802.1Qca [52] and copies of a frame are transmitted over these routes. The issue of network overloading is resolved by eliminating the duplicated copies of the frame, either by an intermediate device, such as a switch, or at the receiver end system. The TSN fault-tolerance mechanisms introduce additional complexity and latency, which is not suitable for low-latency SCRTs. Most of the TSN approaches [53] use uni-casting to transmit TT flows, therefore advantages of membership service such as atomic multicast, tracking the status of active and inactive nodes in the cluster, clique avoidance, and implicit acknowledgment without any extra overhead cannot be implemented at the protocol level in the TSN standards. The most complex issue with TSN is the computation of its TT communication schedules for a large number of network components. Interdependency of routing and communication schedules become computationally intensive due to combinatorial explosion. Recently, a number of TT schedulers [54]–[56] have been proposed to solve the aforementioned issues sequentially and some approaches [57]–[59] use ILP-based solutions but are very time consuming and not scalable for large real-time systems. Most of the TT schedulers [54], [55], [57]–[60] for TSN assume that the underlying network infrastructure is fault-free. In reality such assumptions don't hold for SCRTs. Consequently, the TT scheduling problem under faults environment can lead to a computationally intractable scheduling process [61]. In addition, although TTEthernet and TSN are aimed at using existing, low-cost Ethernet infrastructure for timing-critical traffic, their implementation comes at a high cost of maintenance and design complexity [62].

Wireless communication techniques are being actively developed to build Intelligent Transport Systems (ITS). Numerous car manufacturing and telecommunication companies as well as research and development institutions worldwide, are working to develop a variety of vehicular communication networks. With the advances in the technology, vehicles are becoming increasingly smart. Vehicle connected with each other and with Road Side Units (RSUs) allow updates about weather, traffic density on routes, and communication of safety-related information to other vehicles and surrounding infrastructure [63]. Such systems are referred in the literature as vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-everything (V2X), and vehicular ad hoc networks (VANETs) [64] [65], [66]. The most commonly used communication protocol under V2X architecture is 802.11p, defined in [67] and revised in [68]. The channel access in 802.11p is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme, where all transmitter nodes participate in channel sensing before starting their transmission. When sensing a busy channel, a node will delay its transmission by choosing a random backoff value. Such unsynchronised channel access mechanisms can result in significant transmission delays, which is not suitable for safety-critical real-time systems by any means. Another MAC method proposed for VANETs is Self-organising Time Division Multiple Access (STDMA), where time is divided into time slots constituting a frame, which means a node is transmitting its frame in multiple time slots. For example, a case study used in [69] is using 904 and 2283 time slots per frame in its two different data traffic models. It is important to note here, that any wireless protocol is subject to potential interference, resulting in packet loss or unbounded delays and thus is considered a best-effort protocol, not meeting the fault-tolerance requirements of safety-critical real-time systems. Therefore, using wireless communication in in-vehicle networks to connect different nodes of a safety-critical cluster such as an anti-lock braking (ABS) system is not an appropriate choice, as these protocols are not able to cover all the fault scenarios considered here. However, we do note that these protocols can play an important role in augmenting on-board systems. For example, a gateway can be designed to link safety-critical clusters of a wired network with a wireless network such as V2X architecture to exchange the information on road infrastructure, congestion, and the like.

III. THE INCUS+ PROTOCOL

In the previous section, we discussed different communication protocols for safety-critical distributed real-time systems. All the nodes in TTA-based communication protocols use static and equal-duration time slots for transmitting SCI. The term *FlexRay slot allocation* is used for the slot allocation mechanisms in TTA based protocol i.e. FlexRay. Here, we will present our slot allocation approach that allows variable slot lengths for each node over the TDMA rounds of a cluster cycle. We will refer to this flexible approach as the *INCUS+ slot allocation* approach. Importantly, the slot length of each node is configured according to its payload requirements in each TDMA round, which means a node may have a different slot length in a different TDMA rounds. Consequently, the length of each TDMA round may vary in a cluster cycle. The need of such flexibility is justified by using an example of an autonomous vehicle case study (see section VI).

The principle of operation of communication in our proposed approach follows [70] and [29]. The communication controller of each node is a subsystem that transmits and receives channel data and has a copy of the Message Descriptor List (MEDL) as shown in Fig. 1. The MEDL holds information about different parameters including data transmission and reception time for each node in each TDMA round of a cluster cycle. The MEDL is statically configured and therefore each node knows the exact time to access the communication channel.

Three classes of frames used to control the communication operations over the communication channel are Normal frames (N-Frame), Initialisation frames (I-frames) and Cold Start frames (CS frames). CS-frames are used for integration of nodes during system start-up, Initialisation frames are used to carry synchronisation information that helps the recovering nodes to reintegrate in the system and N-frames are carrying application data. Please note that in this paper, the I-frame term is used for intra-coded pictures from the camera (used in case study, see Section VI-A), therefore we use the full term *initialisation frame* instead of the usual I-frame moniker that is colliding with the video encoder frame type of the same name.

The mechanism to start the protocol cluster or to reintegrate a lost node back to the cluster is the same as in TTP [7] but requires different timeout parameters. For $node_i$, the startup delay is the duration of transmission slots in the TDMA round up until the start of $node_i$ transmission slot. Therefore, the start-up process for the proposed approach is much more efficient as it eliminates the node slot idle time. All nodes have unique parameters for the listen and cold-start timeout as they each have a unique start-up delay.

The configuration for the transmission of the CS-frame is same as used in [70] which is bound with the number of TDMA rounds in cluster cycle, unlike TTP where it is transmitted after each two TDMA rounds that may result in frequent transmission of the frame if a node did not receive and initialisation-frame for up to two TDMA rounds. Initialisation-frames are used to reintegrate recovering nodes and TTP is using a configuration of two TDMA rounds to transmit initialisation-frames over both replicated channels. However, the process of reintegration can be improved if an actuator node (not transmitting any application data but control information) is configured to transmit initialisation-frames (See section VI-C).

Our approach adds the flexibility necessary for better channel utilisation to time-triggered communication, while retaining the deterministic nature of the protocol regarding channel access. The communication schedule of each and every node is stored in the MEDL, and each node has a copy of the MEDL. Therefore, each node knows when to broadcast a frame over the communication channel and when to receive a frame from the communication channel. Please note that when we are talking about flexibility in this paper, we are referring to configurational flexibility as discussed in section II which will not impact the deterministic nature of the protocol. A greater flexibility could be achieved by setting up different MEDLs (each with different communication schedule) for different operational modes and these MEDLs can be switched at runtime. We utilise the same fault tolerance features of earlier time-triggered protocols, such as distributed clock synchronisation, membership and acknowledgment service, bus guardians, and replica determinism to handle faults at the protocol level. We also prevent the communication channel from being a single point of failure through the use of replicated communication channels.

A. MEMBERSHIP SERVICE AND IMPLICIT ACKNOWLEDGMENT

The membership service records the status of all nodes to facilitate fault tolerance. The membership status of each node is recorded in the the membership vector [7]. The bit size of the membership vector N reflects total number of participating nodes in all TDMA rounds of a cluster cycle such as $N=\{1,2,3,\ldots,n\}$.

The membership service informs all the nodes about active and inactive nodes with a latency of one TDMA cycle [7]. The presented approach removes transmission slot overhead time in all TDMA rounds, therefore membership service latency is also improved as compared to traditional time-triggered protocols such as FlexRay. Similarly, no explicit membership information or acknowledgment is required, as the corresponding information can be derived from the embedded frame CRC calculation, further reducing overhead through this implicit acknowledgment mechanism.

Algorithm 1 represents the mechanism of membership service [7] as a transmitter node. It has a membership vector with a size of the total number of active nodes in the cluster. Please note we are using the term *active nodes* for those that transmit frames in their allocated transmission slot. A node that is passive and has nothing to transmit, such as a non-critical actuator node, will not be a part of membership vector. A transmitter sets its membership flag to TRUE in the membership vector before calculating the CRC on the CState (Controller State). The CRC value is embedded in the frame, therefore, the CState is transmitted implicitly, saving more bandwidth on the network. We are using the term *remoteCRC* to represent the value of the CRC calculated on its controller state and the same is true for every transmitter node.

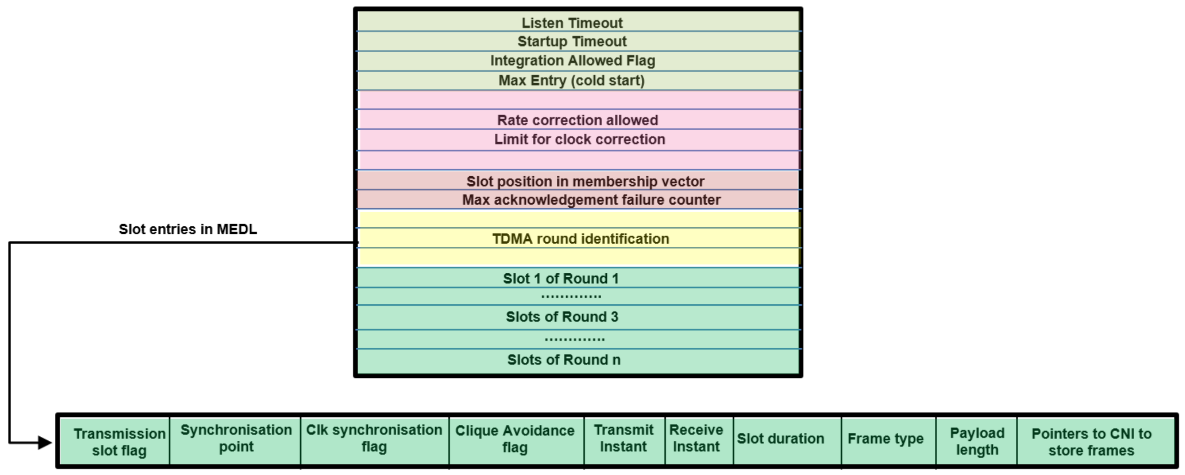


FIGURE 1. Layout of Message Descriptor List (MEDL) extended from [7].

Algorithm 1 Membership Service for a Transmitter Node

```
Require: $sizeof(MembshipVector) = n \in N$
1:  Let $T_{i,j}$ is a transmitter node in slot i of TDMA round j
2:  if ($currentTime == timetoTransmit$) then
3:      set flag of $T_{i,j}$ in membership vector as TRUE
4:      set $agreedSlotCounter$ to ONE
5:      $remoteCRC \leftarrow PerformCRConCState$
6:      Transmit Frame
7:  else
8:      Wait to transmit
9:      Go to 2
10: end if
```

When the local clock of a transmitter node reaches an instant that is marked as the transmission time for the node, the node starts transmitting the frame over the communication channel.

Algorithm 2 represents the mechanism of the membership service [7] as a receiver node. Before receiving the frame from a transmitter $T_{i,j}$, a receiver sets the membership flag of $T_{i,j}$ to TRUE in its membership vector. Then it calculates the CRC check on its controller state. The calculated CRC value is represent by the term *localCRC* and same term is used for all the receiver nodes. When the local clock reaches a window around the time to receive a frame from $T_{i,j}$ in the MEDL, it starts receiving. The CRC value it receives as *remoteCRC* (processed by the transmitter node) will be compared with *localCRC* (processed by the receiver node) and if the result is true then it means the frame received from $T_{i,j}$ is intact, otherwise it will set the membership flag to FALSE for $T_{i,j}$ in its membership vector.

1) FRAME ACKNOWLEDGMENT

The acknowledgment of a frame happens implicitly through the membership service, by including the CState into a

Algorithm 2 Membership Service for a Receiver Node

```
Require: $T_{i,j}$ as transmitter node
1:  Let $R_{i,j}$ is a receiver in slot i of TDMA round j
2:  $R_{i,j}$ sets membership flag of $T_{i,j}$ as TRUE in its
    membership vector
3:  $localCRC \leftarrow PerformCRConCState$
4:  if ($currentTime == timeToReceiveFrame$) then
5:      Receive frame
6:  else
7:      wait to receive
8:      Go to 4
9:  end if
10: Compare CRC values
11: if $remoteCRC == localCRC$ then
12:     $R_{i,j}$ received correct frame from $T_{i,j}$
13: else
14:     $R_{i,j}$ sets membership flag of $T_{i,j}$ as FALSE in its
        membership vector
15: end if
```

frame's CRC calculations [29]. A transmitter transmits replicated frames on two different communication channels and if any one of them is received correctly by a receiver then it will consider the transmitter as an active node at its *membership point* (a post receive phase after the transmission phase of a transmitting node).

If a transmitting node views itself as fully functional then it sets its membership flag to TRUE in its membership vector and its *agreedSlotCounter* to one as shown in algorithm 1. If the successor ($T_{i+1,j}$) of the transmitting node $T_{i,j}$ has received a correct frame on any of the two replicated communication channels then the membership flag for $T_{i,j}$ is set to TRUE in the membership vector of $T_{i+1,j}$, therefore, $T_{i,j}$ can use the $T_{i+1,j}$ transmission as an acknowledgment. The $T_{i,j}$ will only consider the transmission of $T_{i+1,j}$ as an acknowledgment if it receives a correct frame on any of the replicated communication channels. Otherwise, the membership flag of $T_{i+1,j}$ is set to FALSE in the membership vector of $T_{i,j}$ and $T_{i,j}$ will look for the transmission of $T_{i+2,j}$ (next successor) to find an acknowledgment of its transmission.

When $T_{i,j}$ acts as a receiver to receive a frame from its successor, say $T_{i+1,j}$, then there are two cases that are checked when performing CRC calculations on the received frame.

- **CASE 1:** $T_{i,j}$ sets its own and $T_{i+1,j}$ membership flag to TRUE in its membership vector and then performs CRC calculations on its local CState. Then the comparison is performed on *localCRC* and *remoteCRC*.
- **CASE 2:** $T_{i,j}$ sets its own membership flag to FALSE and $T_{i+1,j}$ membership flag to TRUE in its membership vector and then performs CRC calculations on its local CState. Then the comparison is performed on *localCRC* and *remoteCRC*.

If the result of CASE 1 is TRUE then $T_{i,j}$ assumes that its transmission was correct and remains in the membership vector. $T_{i,j}$ will also increase its *agreedSlotCounter* by one. But if the result of CASE 1 is FALSE then CASE 2 will be considered and if the result of CASE 2 is TRUE then it means either transmission of $T_{i,j}$ was not successful or there was an error with $T_{i+1,j}$. At this stage, it is not confirmed whether $T_{i,j}$ or $T_{i+1,j}$ is correct, therefore, $T_{i,j}$ will look for its second successor i.e. $T_{i+2,j}$. If CASE 1 and CASE 2 both fail then it can be predicted that either transmission of $T_{i+1,j}$ is corrupted or $T_{i+1,j}$ is not operational at all. If transmission activity from $T_{i+1,j}$ on any of the replicated communication channel is observed then $T_{i+1,j}$ will be considered as faulty and *failedSlotCounter* will be incremented by one. If $T_{i,j}$ is unable to make a decision by using CASE 1 and CASE 2 then it will use the transmission of its second successor i.e. $T_{i+2,j}$ and following two cases will be tested.

- **CASE 3:** $T_{i,j}$ sets its own and $T_{i+2,j}$ membership flags to TRUE while $T_{i+1,j}$ membership flag to FALSE in its membership vector and then performs CRC calculations on its local CState. Then the comparison is performed on *localCRC* and *remoteCRC*.
- **CASE 4:** $T_{i,j}$ sets its own membership flag to FLASE while $T_{i+1,j}$ and T_{i+2j} membership flags to TRUE in its membership vector and then performs CRC calculations on its local CState. Then the comparison is performed on *localCRC* and *remoteCRC*.

The result of CASE 3 (if TRUE) indicates that transmission of $T_{i,j}$ was correct and $T_{i+1,j}$ was faulty. Therefore, $T_{i+1,j}$ will be removed while $T_{i,j}$ will remain in the membership vector. Both *agreedSlotCounter* and *failedSlotCounter* will be incremented by one. Thus, $T_{i,j}$ is acknowledged. Otherwise, if the result of CASE 4 is TRUE then it means the original transmission from $T_{i,j}$ was erroneous and transmission from $T_{i+1,j}$ was correct. Therefore, $T_{i,j}$ will remove it from membership vector. Both *agreedSlotCounter* and *failedSlotCounter* will be incremented and $T_{i,j}$ marks the transmission of $T_{i+2,j}$ as correct. $T_{i,j}$ will consider its transmission as not acknowledged and if the acknowledgement failure reaches its maximum value (defined in the MEDL) then $T_{i,j}$ will freeze its controller. In the worst case, if both CASE 3 and CASE 4 fail then $T_{i+2,j}$ will be removed from the membership vector and *failedSlotCounter* will be incremented by one. $T_{i,j}$ will choose $T_{i+3,j}$ as second successor and the loop continues, depending upon the number of nodes in the TDMA round.

B. CLIQUE AVOIDANCE

It is possible that an erroneous condition may leave a cluster with multiple cliques where a few nodes do not agree with other nodes on their CState and hence, removes them from their membership list [7]. This leads to formation of multiple cliques. To avoid such errors, we use a clique avoidance algorithm [71] as shown in algorithm 3. A point in time when a node reaches a conclusion about its CState agreement with the rest of the nodes is called a membership recognition point, and different nodes may reach this point at different points in time [7]. At a membership recognition point, a node is able to decide whether the majority of nodes agree with its CState or not by using two of its counters [7] that is *agreedSlotCounter* (which shows the number of other nodes in a TDMA round that are agreeing with the slot status of the node) and *failedSlotCounter* (which shows number of other nodes in a TDMA round that do not agree with the slot status of the node). If the node resides within a majority clique then it will continue its function, otherwise it will restart and reintegrate into the cluster [71].

Algorithm 3 Clique Avoidance

```
Require: with latency of one TDMA round and before
    transmitting the frame in next TDMA round
1: if agreedSlotCounter > failedSlotCounter then
2:     agrees with majority of cluster nodes
3:     set 0 ← agreedSlotCounter
4:     set 0 ← failedSlotCounter
5:     do transmit the frame
6: else
7:     freeze the CC
8:     restart in healthy state and reintegrate with the cluster
9: end if
```

C. CLOCK SYNCHRONISATION

To enable the deterministic behaviour of the protocol where each node has predefined schedule saved in its local MEDL to send and receive the information over the shared channel, it is essential for each node to synchronise its clock with other nodes in order to establish a global time base and run the protocol operations nominally. Clock synchronisation in INCUS+ follows the same procedure as discussed in [70] and [29]. The key difference here, in contrast to these time-triggered protocols, is the resynchronisation point $R_p(t)$ – a slot defined in the MEDL to perform the clock synchronisation. Let k be the number of slots set as resynchronisation point then R_p duration becomes:

$$R_p^{ttp}(t) = k * \tau^{max} \tag{1}$$

where t be the real time and τ^{max} is the slot length. In INCUS+, the formula needs to be adjusted to accommodate different slot lengths:

$$R_p^{INCUS+}(t) = \sum_{i=0}^{k-1} S_i(t) \tag{2}$$

where $S_i(t)$ represents slot length and ($k \geq 4$).

The resynchronisation interval in INCUS+ is shorter than the resynchronisation intervals of traditional protocols such as Flexray and TTP. The reason for this is the node slot net idle time, which is zero for all node slots in our approach. This provides the ground for fast-tracking the process of clock synchronisation. The fault-tolerant average algorithm [72], which was formally verified [73] to synchronise the clocks of all nodes in the cluster is shown in algorithm 4.

```
Algorithm 4 Clock Synchronisation
Require: received correct frame only
 1: Initialised stack of four with zero
 2: Capture the time interval when received the first bit
    from sending node
 3: Let $\hat{t_{RF}}$ is the actual time when receiver starts receiving
    the frame from the sender
 4: Let $t_{RF}$ is time when receiver supposed to receive the
    frame from the sender - defined in MEDL
 5: Calculate correction term $\Delta_{corrterm}$
 6: $\Delta_{corrterm} = \hat{t_{RF}} - t_{RF}$
 7: PUSH $\Delta_{corrterm}$
 8: if $R_p^{incus+}(t)$ then
 9:     POP the values
10:     discard the largest and smallest values
11:     take the average of remaining two values
12:     apply this average to correct the local clock
13: else
14:     wait for the interval $R_p^{incus+}(t)$
15:     push new values of $\Delta_{corrterm}$ to the stack
16: end if
```

D. BUS GUARDIAN

The issue of babbling idiot faults is handled through independent bus arbitration, where each node is equipped with an independent bus guardian in a bus topology.[1] We are using an independent bus guardian design that has its own clock and has the complete transmission schedule for all TDMA rounds of its respective node and listens for the incoming traffic at specific instants (defined in the MEDL). The guardian prevents bus monopolisation by a sending node that is trying to transmit more often or longer than its schedule allows during a TDMA round [7]. It also prevents CC from transmitting outside of its allocated transmission window [7].

[1]Please note that while we use a bus topology here, at a protocol level we only require broadcast semantics. Therefore, the same applies to other topologies with equivalent semantics, such as a star topology.

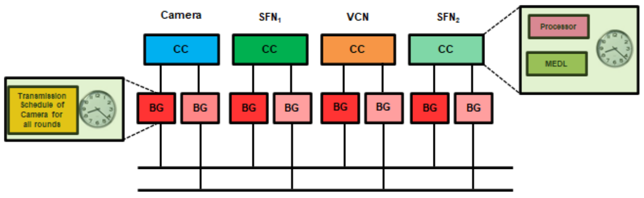


FIGURE 2. Bus guardian layout in bus topology with full redundancy.

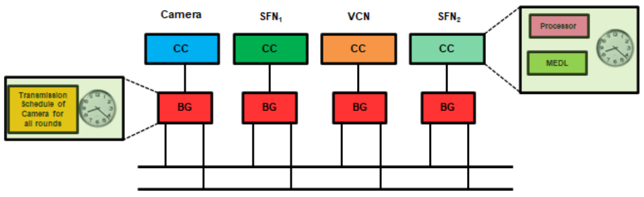


FIGURE 3. Bus Guardian layout in bus topology with reduced redundancy.

In Fig. 2, we show the architecture of maximum redundancy in a bus topology, where each bus is protected from each node by a separate bus guardian. In case of a bus guardian failure, a node is still able to transmit on the other channel. However, since critical nodes have to be replicated to avoid a single point of failure, a simpler design can then be used, where only a single bus guardian is used for each node over the redundant communication channels (Fig. 3).

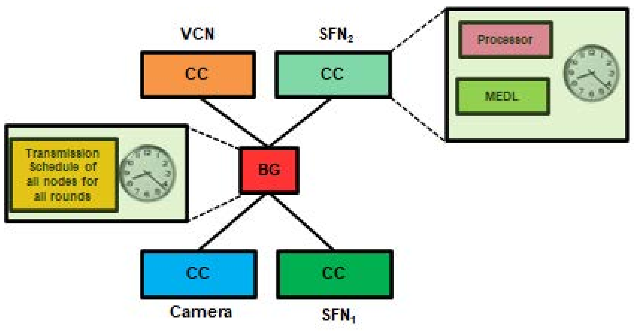


FIGURE 4. Bus guardian layout in star topology with reduced redundancy.

In a star topology, by comparison, a single bus guardian for all nodes as shown in Fig. 4, would introduce a single point of failure and in-case of a fault in the guardian, traffic on both channels would be interrupted. Therefore, the required architecture would be star couplers in a redundant star topology as shown in Fig. 5.

The bus guardian in the bus topology, where it is using a periodic signal from its respective node to synchronise its clock with the CC, will prevent the node from transmitting more than once in a TDMA round but will fail to stop it from sending outside it scheduled transmission time (only if the CC's clock get faulty). However, as mentioned in Section III-C, each node synchronises its clock with all other

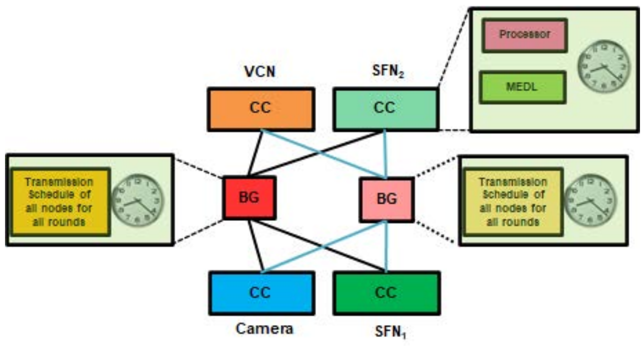

VCN
SFN$_2$
CC
CC
Processor
MEDL
Transmission Schedule of all nodes for all rounds
BG
BG
Transmission Schedule of all nodes for all rounds
CC
CC
Camera
SFN$_1$

FIGURE 5. Bus guardian layout in star topology with full redundancy.

nodes using the fault-tolerant average algorithm, thus clock synchronisation between a node and its BG utilises the same input from the other nodes as well. In the worst case, where the node's clock gets faulty and the BG's clock adjust itself with the node's clock, this may result in bus access outside the node's scheduled time. This fault can be identified by using the membership service and such a node will be cut off from the rest of the cluster until it restarts and reintegrates into the cluster again. The bus guardian must allow the CC to transmit at the correct time and should not block the full transmission of its frame within the correct transmission boundary as defined in the MEDL. Therefore, in order to tolerate the timing differences between bus guardian and the CC, the bus guardian will open its gate to the bus a little earlier than the actual transmission time and closes the the shutter a little later than the schedule time, as we will detail in the next section.

IV. FORMAL VERIFICATION OF SLOT TIMING

Rushby [74] published a formal verification for TTA-based communication protocols such as TTP. Hence, we will adopt Rushby's model for our formal verification, however since his model only works for fixed and equal-length node slots in each TDMA round, we need to modify it for our approach. In our approach, the slot length of each node in each TDMA round is configured in accordance with its transmission payload. Hence, transmission slot lengths of a node can vary in different TDMA rounds.

We will now formally verify the window timings of each slot of a node in different TDMA rounds of the cluster cycle. The basic pattern of communication is based on the global schedule which holds the information about slot positions of transmitters, as well as slot start times and durations.

A transmitter may start its frame transmission after some delay from its slot start time and finish the transmission some time after the allowed slot duration has elapsed. Similarly, a receiver starts listening for a frame at the beginning of its receive window and closes its receive window when the frame-receive duration has expired. All the events are controlled at each node through its clock. It is possible that clocks in different nodes may deviate from each other but must be synchronised within a small threshold. As a consequence, it is possible that a transmitter starts its frame transmission before some receiver starts listening for the frame or finishes its transmission when some receivers have already stopped listening for the frame. This situation can lead to inconsistencies among participating nodes which can create different cliques among nodes. Therefore, care must be taken while selecting parameters for window timings of transmitters, receivers, as well as bus guardians (responsible to block any transmission outside an allocated time slot).

TABLE 1. Definition of the parameters used in the formal verification model.

Parameter	Description
$Slot_{st}$	Time to start the node slot
$Slot_{len}$	Length of the node slot
F_tS	Maximum time to start frame transmission
F_tE	Time to end frame transmission
GS	Time to open bus guardian window
GB	Time to block the transmission
GE	Time to shut bus guardian window
R_wS	Time when receiver open its window to receive the frame
R_wE	Time when receiver closes its window

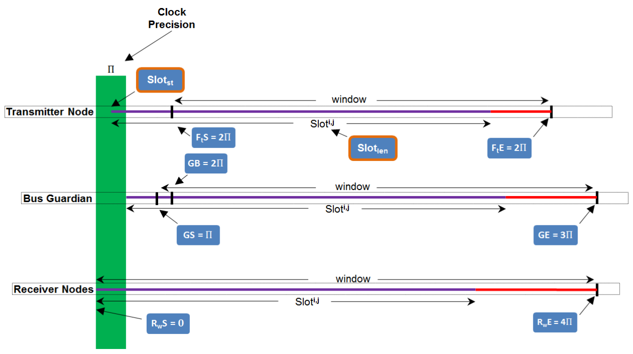


FIGURE 6. Slot window timing parameters extended from [74].

The parameters used in our formal verification model are shown in Table 1. The values of these parameters, as shown in Fig. 6, need to satisfy the following requirements:

- **Agreement:** If a frame transmission is received by any non-faulty receiver then all non-faulty receivers must receive the transmission.
- **Validation:** If a frame is transmitted by a non-faulty transmitter then all non-faulty receivers must receive the transmitted frame.

Before starting the analysis, we will recap the requirements and assumptions of INCUS+ model which are as follows:

- We are not using multiplexed slots which means one slot per node in a TDMA round.
- Slot duration ($Slot_{len}$) of a node is configured according to its transmission payload in each TDMA round.
- One frame per slot.

The hierarchy of communication is as follows:
Transmitter →Bus Guardian→ Receivers

Thus validity requirement is decomposed into two sub-requirements which are:

1) Frame transmission timing from the transmitter to its bus guardian.
2) Timing from the bus guardian to the receivers.

We build on and extended Rushby's [74] model as follows: We use notion C for the local clock time of each node. We use uppercase variable names for clock time quantities while lowercase names are used to represent real-time quantities. Therefore we can say that $C_s^{i,j}(t)$ is the value of s's clock at time t in slot i of round j. For any participating node in a cluster, whether it is acting as a sender node such as s, or as a receiver node such as r in any slot i of TDMA round j, their clocks are synchronised if the reading of the clocks of both nodes are within a precision of Π. Therefore, according to clock synchronisation it should be:

A. CLOCK SYNCHRONISATION

$$\left|C_s^{i,j}(t) - C_r^{i,j}(t)\right| \le \Pi$$

B. R1

If a frame is transmitted by a communication controller of a non-faulty transmitter then its non-faulty bus guardian must also allow the transmission.

Proof: Assuming a node slot i start at $Slot_{st}^{i}$ where the length of the slot is exactly configured according to the transmission payload or frame size in that slot of a TDMA round. Therefore, we can say that slot length for frame f in slot i of a TDMA round j should be ($Slot_{len}^{i,j}(f)$). We assume that the transmitter starts its transmission at some offset F_tS after the start of the slot and ends the frame transmission at some offset F_tE after the time needed to transmit the frame. The associated bus guardian also opens its window at some offset GS prior to the start of slot and closes its window at some offset GE after the time needed to transmit the frame.

If s is a transmitter node at real time *t1* then the transmission start time for its frame f in its allocated slot i of TDMA round j will be:

$$C_s^{i,j}(t1) = Slot_{st}^{i,j}(f) + F_tS \tag{3}$$

At this point, the bus guardian for node s must already have opened its window in transmission slot i of round j to allow transmitter to transmit its frame f, therefore, we need

$$C_{bg}^{i,j}(t1) \ge Slot_{st}^{i,j}(f) + GS \tag{4}$$

Clocks of s and its bus guardian must obey

$$-\Pi \le C_{bg}^{i,j}(t1) - C_s^{i,j}(t1) \le \Pi$$

By using (3) it should be

$$C_{bg}^{i,j}(t1) \ge Slot_{st}^{i,j}(f) + F_tS - \Pi$$

To satisfy (4) we must have

$$F_tS \ge GS + \Pi$$

and this is clearly proved by the fact that parameters selected for window timing are $F_tS = 2\Pi$ and $GS = \Pi$.

Suppose t2 is the physical time where s is transmitting its frame ($F_tE \ge F_tS$); hence we can say that

$$C_{bg}^{i,j}(t2) = Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + F_tE \tag{5}$$

At this point in time its bus guardian window $C_{bg}^{i,j}(t2)$ must open therefore we can say that

$$C_{bg}^{i,j}(t2) \le Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE \tag{6}$$

According to clock synchronisation it should be

$$-\Pi \le C_{bg}^{i,j}(t2) - C_s^{i,j}(t2) \le \Pi$$

Therefore, we can say that

$$C_{bg}^{i,j}(t2) \le Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + F_tE + \Pi \tag{7}$$

To satisfy (7), $GE \ge F_tE + \Pi$ has to be satisfied. As parameters selected for window timings (see Fig. 6) can illustrate that ($F_tE = F_tS$) therefore we can say that

$$GE \ge GS + 2\Pi \tag{8}$$

(8) can clearly be proven by the fact that ($GS = \Pi$) and ($GE = 3\Pi$).

C. R2:

If a non-faulty bus guardian passes a frame then it will be received by all non-faulty receivers.

Proof: For a frame f of a transmitter in its node slot i of TDMA round j, assuming the bus guardian opens its window GS clock units after the slot start time for its receptive node and closes the window at ($Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE$). Let us suppose receiver nodes are ready to listen for a frame at R_wS units after the slot start time of the transmitter node and stop receiving the frame at ($Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + R_wE$).

At physical time t1 we have

$$C_{bg}^{i,j}(t1) = Slot_{st}^{i,j}(f) + GS \tag{9}$$

At time t1 when the bus guardian opens its window, the receiver r must already have opened its window to listen for a frame f, therefore, we can say that

$$C_r^{i,j}(t1) \ge Slot_{st}^{i,j}(f) + R_wS \tag{10}$$

For the receiver at instant t1 and by using clock synchronisation rule we have

$$-\Pi \le C_r^{i,j}(t1) - C_{bg}^{i,j}(t1) \le \Pi$$

By using (9)

$$C_r^{i,j}(t1) \ge Slot_{st}^{i,j}(f) + GS - \Pi$$

and to prove (10) we need $GS \ge R_wS + \Pi$ and this can be proved by substituting the values of parameters shown in Fig. 6.

At any physical time t2 when the bus guardian window is still open to allow the transmission such that

$$C_{bg}^{i,j}(t2) = Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE \tag{11}$$

Therefore, receiver window must be open at t2.

$$C_{r}^{i,j}(t2) \leq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + R_wE \tag{12}$$

According to clock synchronisation, we have

$$C_{r}^{i,j}(t2) \leq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE + \Pi$$

To validate (12), we need $R_wE \geq GE + \Pi$ and this can be proven by substituting the values of parameters GE and R_wE.

The validation property is clearly proved by the requirements R1 and R2, first requirement (R1) ensures that a frame transmitted by a non-faulty transmitter will be passed by its non-faulty bus guardian whereas second requirement (R2) makes sure that a frame passed by non-faulty bus guardian should be received by all non-faulty receivers.

For the agreement property, R2 proves that a frame transmitted by a non-faulty transmitter must pass by its non-faulty bus guardian, which ensures that all non-faulty receivers will receive the frame. The other way round, where a faulty transmitter is trying to transmit a frame at an incorrect time, then its bus guardian will block such a transmission if it falls entirely outside the allowed transmission window, or it will truncate the frame if it falls partially outside the transmission window. Therefore, if a transmission was blocked by a bus guardian, it will not be observed by any receiver. Similarly, if a transmission is truncated by a bus guardian then all the non-faulty receivers will reject such a transmission.

D. PREVENTION OF SLOT OVERLAPPING

So far, the model assure that a non-faulty transmitter must transmit its frame within its slot boundaries and its non-faulty bus guardian must allow that transmission. Now, another issue that we need to tackle is overlapping slots. A frame transmitted by a non-faulty transmitter must not interfere with the next transmitter opening its transmission window.

The design rule to prevent such erroneous scenarios is [74]:

- The next transmission takes place no earlier than 4Π after the end of the previous transmission.

Please note we are using slot length configuration of a node on the basis of its payload and transmitting one frame per slot. If we are transmitting a frame *f* in a slot *i* of round *j* then the next slot should be (*i+1,j*). Therefore, the above design rule can be formalise in INCUS+ as follows:

$$Slot_{st}^{i+1,j}(f) \geq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + 4\Pi \tag{13}$$

E. R3

The window of one communication controller must not overlap with the window of next communication controller in the following slot. Therefore, we can say that a non-faulty communication controller of a transmitter node must finish its transmission before its non-faulty bus guardian open its window for the next slot.

Proof: At real time *t*, a communication controller of a transmitter node *s* ends its transmission for a frame *f* in its slot *i* of round *j*, therefore, it should be

$$C_{s}^{i,j}(t) = Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + F_tE \tag{14}$$

Now, its bus guardian C_{bg} window should be open for the next slot (*i,j+1*) not earlier than *t*

$$C_{bg}^{i,j+1}(t) \leq Slot_{st}^{i,j+1}(f) + GS \tag{15}$$

As clock synchronisation requires

$$-\Pi \leq C_{bg}^{i,j+1}(t) - C_{s}^{i,j}(t) \leq \Pi$$

By using (14)

$$C_{bg}^{i,j+1}(t) \leq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + F_tE + \Pi \tag{16}$$

To satisfy (15), we need

$$Slot_{st}^{i,j+1}(f) \geq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + F_tE + \Pi - GS \tag{17}$$

This satisfies (13) by using the values of $F_tE = 2\Pi$ and $GS = \Pi$

Slot overlapping may also occur because of a faulty bus guardian that may result in a potential transmission collision. The design rule to prevent such issues in INCUS+ is as follows:

- No transmission is allowed if a transmitter does not start its transmission within the boundary of $Slot_{st}^{i,j}(f) + F_tS$.

As we now use variable slot lengths for each node in each TDMA round, the above requirement R3 to prevent slot overlapping would not be sufficient for all scenarios such as a bus guardian and the transmitting node agreeing on their slot positioning, but assume to be in different TDMA rounds. Bus guardian parameter *GB* serves to overcome such an issue. To this end, we have the following requirement.

F. R4

If a transmitter node is trying to transmit its frame in the allocated slot but after its F_tS has elapsed then its bus guardian must block such a transmission.

Proof: Suppose *s* is a transmitter node in slot *i* of round *j* to transmit its frame *f* at physical time *t* where F_tS has already elapsed, therefore

$$C_{s}^{i,j}(t) > Slot_{st}^{i,j}(f) + F_tS \tag{18}$$

At this point, bus guardian for *s* must already blocks the controller from transmission. Therefore, it should be

$$C_{bg}^{i,j}(t) = Slot_{st}^{i,j}(f) + GB \tag{19}$$

Again, clock synchronisation requires

$$-\Pi \leq C_{bg}^{i,j}(t) - C_{s}^{i,j}(t) \leq \Pi.$$

Therefore, we can say that

$$C_{bg}^{i,j}(t) < Slot_{st}^{i,j}(f) + F_tS + \Pi \tag{20}$$

To satisfy (19), we must have $GB < F_tS + \Pi$ and this is clearly proved by the fact that $GB = 2\Pi$ and $F_tS = 2\Pi$.

V. PROTOCOL BEHAVIOUR UNDER FAULTS

In our formal verification model above, the timing constraints are proved by assuming that all the participating nodes (transmitters, receivers and bus guardians) are non-faulty. For example, a frame transmitted by a non-faulty node should be passed by its non-faulty bus guardian. Now, we will discuss different fault scenarios and explain the behaviour of our protocol to detect such errors in both time and value domains. Please note that as with all TTA-based SCRT protocols, the fault hypothesis is to handle a single fault at a time.

A. WHAT IF A TRANSMITTER FAILS?

There are multiple reasons for a transmitter to fail and in this section we will discuss multiple failure scenarios and protocol behaviour against these scenarios.

- *Completely off the scheduled frame transmission:* It is quite possible for a faulty node to transmit at an instant that is not scheduled for the frame transmission. In this case it will violate R1 (3) which requires:
  $C_s^{i,j}(t1) = Slot_{st}^{i,j}(f) + F_tS$
  This will be handled by (4) that is:
  $$C_{bg}^{i,j}(t1) \geq Slot_{st}^{i,j}(f) + GS$$
  As the bus guardian has a copy of the MEDL, it knows the transmission scheduled for its node. The bus guardian window will remain closed for any other instant, therefore any such transmission attempt by the associated node will fail and no other node will be able to receive such a transmission.
- *Transmitter transmits longer than expected:* A faulty node may try to occupy the the communication channel longer than its slot length. By this time, the bus guardian will already shuts it window such as:
  $$C_{bg}^{i,j}(t) \geq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE$$
  Therefore, such transmission will be truncated by the bus guardian (as bus guardian window will be shut after the actual transmission time has been elapsed (defined in MEDL)). This truncated frame still be received by the receivers, however this incorrect frame will be further handled by other protocol services in the value domain. The CRC check on the received frame will fail at non-faulty receivers. As a consequence, all the non-faulty receivers will discard such frame and they will remove the transmitter from their membership list.
- *Transmission slot position is incorrect:* Another fault scenario may arise where the transmitter node assumes itself in a different slot than the actual slot. For example, actual slot positioning of a node is slot *i* of round *j* and node is considering itself in a slot *i* of round *j+1*. Unlike the existing TTA-based approaches, the slot length of the node in INCUS+ may vary in different TDMA rounds, therefore, further failure scenarios may arise out of it.
  - If the slot length of a node in round *j+1* is less than the slot length in round *j* i.e. ($Slot_{len}^{i,j}(f) > Slot_{len}^{i,j+1}(f)$). In this case, the bus guardian window, which is greater than its window in round *j+1*, will allow the transmitter to to transmit its frame. While that means, in the time domain, this error will not be detected, at receivers, this error will be detected in the value domain due to the CRC checksum of the CState for the transmitter being different from the rest of the nodes. Therefore, such frames will be rejected by the receivers and the transmitter node will not be acknowledged in subsequent slots. This will cause transmitter to restart after one round and reintegrate with the cluster.
  - If the slot length of a node in round *j+1* is greater than the slot length in round *j* (i.e. $Slot_{len}^{i,j}(f) < Slot_{len}^{i,j+1}(f)$). In this case, the bus guardian window, which is less than its window in round *j+1*, will truncate the frame as the transmitter will try to transmit outside the allowed duration. At this instant, bus guardian will shut its window
  $$C_{bg}^{i,j}(t) \geq Slot_{st}^{i,j}(f) + Slot_{len}^{i,j}(f) + GE$$
  Therefore, such transmissions will be truncated by the bus guardian (as the bus guardian window will be shut after the actual transmission time defined in the MEDL for slot *i* of round *j* has elapsed). This truncated frame will still be received by the receivers, however, truncated frames are already by other protocol services in the value domain, such as the CRC check. The CRC check on the received frame will fail, and as a consequence, all non-faulty receivers will discard such a frame and will remove the transmitter from their membership list.

B. WHAT IF THE BUS GUARDIAN FAILS

In the previous scenario, we discussed a faulty transmitter and here, we will discuss what happens if a bus guardian fails. There are multiple failure scenarios for a bus guardian and we will discuss them one by one in the remainder of this section.

- *Bus guardian blocks correct transmission:* If a bus guardian gets faulty and blocks the correct transmission from its respective transmitter node. This will violate the (9) of R2 which is:
  $$C_{bg}^{i,j}(t1) = Slot_{st}^{i,j}(f) + GS$$
  In this case none of the receivers will receive the expected frame. Therefore, in the value domain this will be detected by the membership service and all the non-faulty receivers will remove the transmitter from their membership vector, which will force the transmitter to restart and attempt to reintegrate.
- *Bus guardian truncates correct transmission:* If the bus guardian gets faulty in such a way that it truncates a correct frame then, in the value domain, again the CRC checksum at receivers will be detected as incorrect and all the receivers will discard such a frame and remove the

transmitter from their membership list. The transmitter will detect such an error when it finds its CState not being consistent with the majority of the nodes in the cluster.

- *Slot positioning for Bus guardian is incorrect:* A fault scenario may arise when a bus guardian assume itself in a different slot than the actual slot. For example, the actual slot position of a guardian is slot *i* of round *j* while the bus guardian is considering itself in a slot *i* of round *j+1*. Unlike the existing TTA-based approaches, the slot length of the node in INCUS+ may vary in different TDMA rounds and so does the bus guardian window. Therefore, further failure scenarios may arise out of this.
  - If the slot length of a node in round *j+1* is less than the slot length in round *j*. In this case, the bus guardian window which should be less than its window in round *j* will truncating the correct frame which again violates R2. Therefore, the CRC check on the receiving frame will fail at non-faulty receivers. As a consequence, all the non-faulty receivers will discard such a frame and will remove the transmitter from their membership list. This then will force the transmitter to restart and attempt to reintegrate into the cluster.
  - If the slot length of a node in round *j* is less than the slot length in round *j+1*, the bus guardian window which should be greater than its window in round *j* will allow the transmitter to to transmit its frame (if the transmitter starts transmission within the boundary of $Slot_{st}^{i,j}(f) + F_tS$). This error will be detected in subsequent rounds where bus guardian window starts truncating the correct frame when its window opens for less time than the required time to transmit a frame, as per the previous scenario. In the worst case scenario, it may take up to a full cluster cycle time to detect such an error.
  - If the slot length of a node in round *j* is less than the slot length in round *j+1*, the bus guardian window would be greater than its window in round *j*, which will again allow the transmitter to to transmit its frame. As the bus guardian window would be open longer than it should be, therefore, frame transmission could be thought to overlap with the next slot (if transmission starts after $Slot_{st}^{i,j}(f)+F_tS$). But this will violate R4, which states:

    $$C_s^{i,j}(t) \leq Slot_{st}^{i,j}(f) + F_tS$$

    Therefore, the bus guardian will shut its window at this instant such that:

    $$C_{bg}^{i,j}(t) = Slot_{st}^{i,j}(f) + GB$$

    This means the bus guardian will block any transmission that starts after $Slot_{st}^{i,j}(f) + F_tS$.

VI. AUTONOMOUS VEHICLE CASE STUDY

The bandwidth requirements to deploy ADAS systems are very high, specifically by adding multiple cameras with different focal lengths to detect multiple objects such as traffic lights, pedestrians, road signs, etc. Therefore, multiple compression techniques are used to handle huge data traffic in in-vehicle networks. A generic video compression technique, H.264, has been used in this case study and it has been shown by Tankred Hase *et al.* [75] that using the H.264 compression at a low bit rate of up to 0.125Mbps provides sufficient quality to detect an obstacle. To elaborate the need of flexible TDMA round lengths, i.e., the length of rounds may vary depending on differing payloads of a node in different TDMA rounds, we will use this case study. The autonomous vehicle moves along the road and avoiding obstacles as detected by sensors such as cameras. Importantly, emergency breaking systems depend on these sensors to detect critical situations, such as a pedestrian on the road, that require automated emergency braking within a tightly constraint deadline to prevent a collision.

To develop a safety-critical, distributed real-time system node cluster for this case study, we utilise the concept of a platooning system [76], [77] and a pedestrian detection system [78].

The layout of the autonomous vehicle is shown in Fig. 7. The basic building blocks of the system are the Sensor fusion Node (SFN), Vehicle Controller Node (VCN), and a Camera with on-board H.264 compression. Two redundant SFNs are deployed to improve fault tolerance in the hardware domain. The components, VCN and Camera, are not replicated as the drive controller interface is able to perform a fail-safe operation (emergency stop) if it does not get a valid life sign from the VCN. The communication among camera, SFN and VCN is performed through replicated communication channels with a communication capacity of 1Mbps. The layout of the time-triggered cluster of our case study is shown in Fig. 8.

Each node in the cluster is divided into three layers i.e. Host, Communication Netwrok Interface (CNI) and Communication Controller (CC) layer as shown in Fig. 9. The CC is a part of the communication subsystem, interacting with the host through the CNI. To avoid the design complexities of the protocol, we have used Logic-Labelled Finite State Machines (LLFSMs) to implement the functionality of each communication controller [79]. We used a subsumption architecture where each controller is decomposed into sub-machines. The host processes the information from/to sensors/actuators and exchanges the data with the CC through the CNI. The CCs fetch and transmit the data from and to the communication channel at the periodic instances stored in the MEDL. Each CC has a copy of the MEDL and hence, knows in advance when to transmit and receive the data to and from the communication channel. This mechanism guarantees the deterministic behaviour of the protocol.

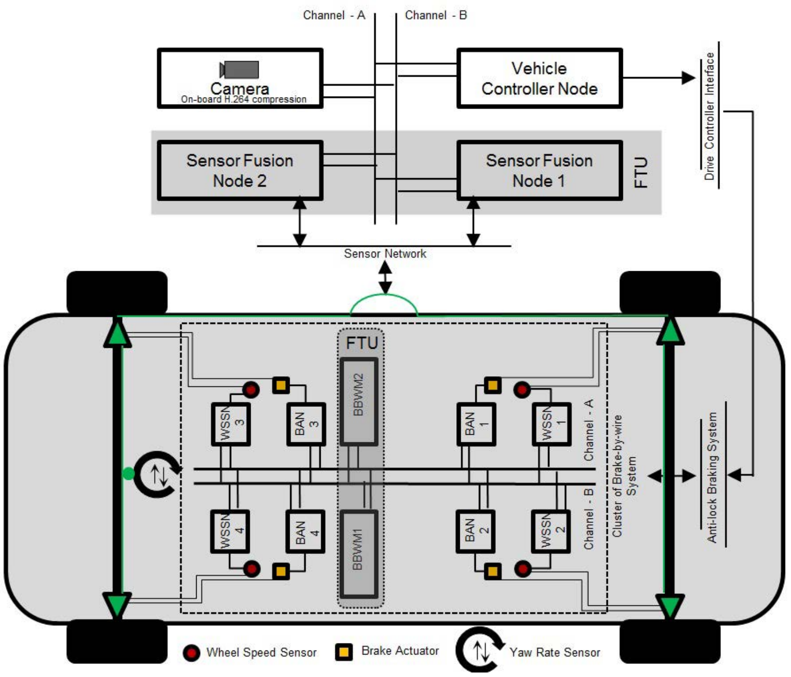


FIGURE 7. Layout of autonomous vehicle.

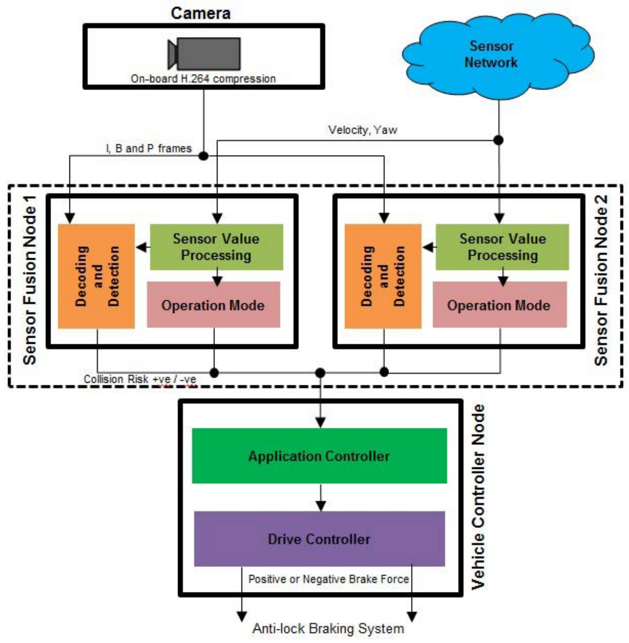
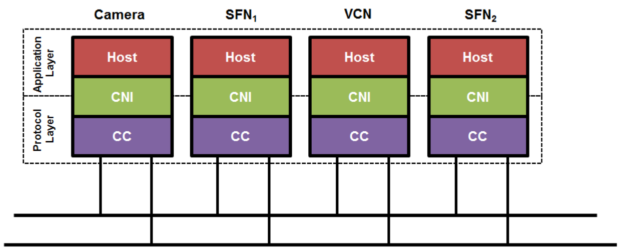


FIGURE 8. Layout of time-triggered class-C network of autonomous vehicle [77].

A. CAMERA

The camera is transmitting compressed frames such as Intra-coded (I), Bidirectional (B) and Predicted (P) frames by using H.264/AVC codec. To deliver videos in real-time, a frame from a camera (e.g. I frame) is transmitted in a TDMA round, followed by the next frame (e.g. B frame) in the next round, etc. These frames are of different sizes, an I frame is many times larger than a P and B frames. Using the same transmission slot length (required by I frame) in the following TDMA rounds to send B and P frames results in huge inefficiency regarding channel utilisation as payloads of B and P frames are much smaller than I frames. and a GOPs (Group of Pictures) configuration is used with M = 2 (distance between two anchor frames) and N = 3 (distance between two full images) which leads to the sequence `IBPIBP` as shown in Fig. 10. The camera is transmitting GOPs on the basis of one frame per round, therefore a complete sequence of the compressed frames is transmitted in three different TDMA rounds. Consequently, the cluster cycle of the case study consists of three TDMA rounds.



CC: Communication Controller Host: Host Computer CNI: Communication Network Interface

FIGURE 9. Internal structure of each node in the protocol cluster.

As the camera is transmitting different size frames in different TDMA rounds of a cluster cycle, the slot length of this node should be different for each TDMA round (Table 2).

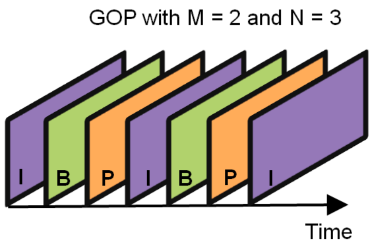


FIGURE 10. GOP configuration.

B. SENSOR FUSION NODE

The SFN decodes the encoded stream received from the camera and forwards it to the detection module. The principle of operation to detect a pedestrian is same as used in [78]. The detection module extracts the region of interest in each frame and to detect a pedestrian, it exploits the fact that object size increases when relative distance decreases. The distance to a possible collision is calculated on the basis of change in object size, velocity and yaw of the autonomous vehicle. A more detailed functionality of pedestrian detection can be found in [78]. The sensor value processing module takes velocity and yaw rate value from the sensor network and forwards it to the detection module. The decoding and detection module sends an output of 8 bits to the VCN that informs the actuator node about a possible risk of collision and the VCN acts accordingly. To keep it simple, a single mode of operation is used in this case study and the state of each node is marked as a ready state.

C. VEHICLE CONTROLLER NODE

The Application controller in the VCN receives the information on collision risk and passes on this information to the Anti-lock-braking System (ABS) through its drive controller interface. All modern vehicles are equipped with the ABS system as it helps to stop the vehicle on slippery roads by avoiding uncontrolled skidding. The cluster of the ABS system consists of Wheel Speed Sensors Nodes (WSSN), Brake Actuator Nodes (BAN) and Brake-by-Wire Manager Nodes (BBWM). ABS sits on top of the Brake-by-wire system and is responsible for applying automated brakes to stop the vehicle when it receives a positive signal for brake force from the VCN. A detailed description of the ABS's cluster can be found in [70].

For the case study, the VCN is designed to send an initialisation-frame in each TDMA round. This is because the VCN acts as an actuator node and does not need to transmit any application data within its local cluster which includes SFN and Camera nodes. This approach helps to speed up the reintegration process of recovering nodes.

VII. PERFORMANCE COMPARISON

We discussed a number of TTA-based communication protocols in section II and gave a detailed reasoning of why they are not suitable when it comes to flexibility while retaining the necessary safety and reliability, to develop fail-safe operations for SCRT systems. In this section, we will analyse the impact of fixed and equal-length slot configurations on bandwidth and channel utilisation. We will compute the corresponding results for INCUS+ and compare it with the existing TTA-based FlexRay communication protocol that is prevalent in automotive industry to manage in-vehicle real-time communications. FlexRay was initially used in BMW X5 (E70) in 2006 and the full use of FlexRay was introduced in the braking system of BMW 7 series (F01) in 2008. Currently, FlexRay is still actively operating in BMW G11 (normal version), BMW G12 (long-wheelbase version) and BMW F12 generations. Table 3 shows a descriptive list of the terms used in the computational model in order to compare FlexRay and INCUS+. Please note for simplicity, we are not using the timing parameters discussed in Section IV as these, while hardware-dependent, would be the same for all approaches.

A. FlexRay SLOT ALLOCATION

τ^{max} of each node comprises of T_i^{trans} and T_i^{idle} as given in Fig. 11. T_i^{ovhd} and T_i^{idle} of $node_i$ are:

$$T_i^{idle} = \tau^{max} - T_i^{trans} \tag{21}$$

$$T_i^{ovhd} = T_i^{idle} + T^{ifg} \tag{22}$$

$\hat{T}^{ovhd\_r}$ for $\boldsymbol{n}$ number of nodes in a TDMA cycle should be:

$$\hat{T}^{ovhd\_r} = \sum_{i=0}^{n-1} T_i^{ovhd} \tag{23}$$

If there are $\boldsymbol{k}$ number of TDMA rounds then the total overhead time ($\hat{T}^{ovhd\_c\_cycle}$) in a cluster cycle can be calculates as:

$$\hat{T}^{ovhd\_c\_cycle} = \sum_{i=1}^{k} \hat{T}_i^{ovhd\_r} \tag{24}$$

T^{fr_r} for $\boldsymbol{n}$ number of nodes is:

$$T^{fr\_r} = n \cdot (\tau^{max} + T^{ifg}) \tag{25}$$

The length of a $T^{fr\_c\_cycle}$ in FlexRay slot allocation approach with k number of TDMA rounds should be:

$$T^{fr\_c\_cycle} = k \cdot (T^{fr\_r}) \tag{26}$$

By using (24) and (26), CU in a FlexRay cluster cycle can be calculated as:

$$CU = \left[\frac{T^{fr\_c\_cycle} - \hat{T}^{ovhd\_c\_cycle}}{T^{fr\_c\_cycle}}\right].100 \tag{27}$$

B. INCUS+ SLOT ALLOCATION

We now evaluate our INCUS+ enhanced slot allocation approach, where we customise the length of the slot for each node in relation to its transmission payload in each TDMA round of the cluster cycle. This customisation avoids the

TABLE 2. The ideal transmission slot length for each node of the autonomous vehicle system during a cluster cycle.

	First TDMA Round		
Node Name	Data	Frame Length	Slot Length
Cam-I	75032 bits	$75032 + 28 = 75060$ bits	75060 μs
SFN	8 bits	$8 + 28 = 36$ bits	36μs
VCN	Nil	$0 + 28 = 28$ bits	28μs
	Second TDMA Round		
Node Name	Data	Frame Length	Slot Length
Cam-B	4400 bits	$4400 + 28 = 4428$ bits	4428μs
SFN	8 bits	$8 + 28 = 36$ bits	36μs
VCN	Nil	$0 + 28 = 28$ bits	28μs
	Third TDMA Round		
Node Name	Data	Frame Length	Slot Length
Cam-P	20392 bits	$20392 + 28 = 20420$ bits	20420μs
SFN	8 bits	$8 + 28 = 36$ bits	36μs
VCN	Nil	$0 + 28 = 28$ bits	28μs

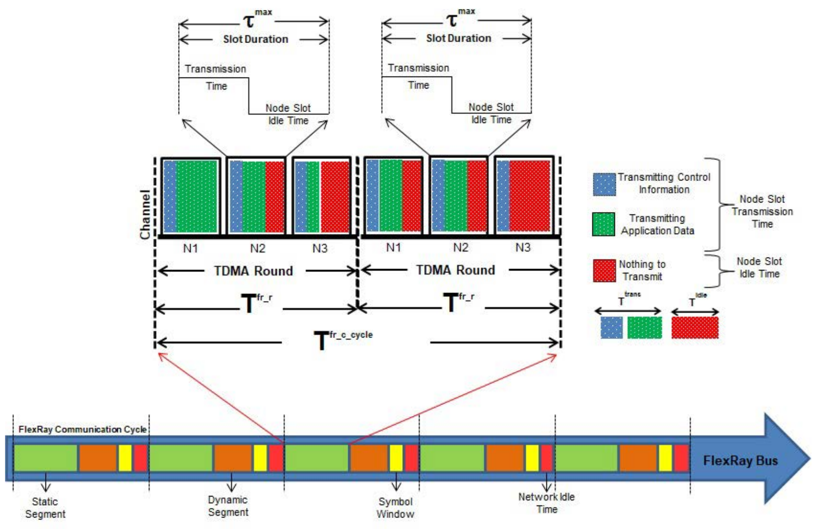


FIGURE 11. Static segment of FlexRay slot allocation over a cluster cycle.

overhead time as demonstrated in Fig. 12. Therefore, T_i^{idle} is zero and τ_i^{inc+} in each TDMA round becomes:

$$\tau_i^{inc+} = T_i^{trans} \tag{28}$$

For a number of nodes $\boldsymbol{n}$, T^{inc+_r} in INCUS+ is:

$$T^{inc+\_r} = \sum_{i=0}^{n-1}(\tau_i^{inc+} + T^{ifg}) \tag{29}$$

Similarly, for a number of TDMA rounds $\boldsymbol{k}$, $T^{inc+\_c\_cycle}$ becomes:

$$T^{inc+\_c\_cycle} = \sum_{i=1}^{k}(T_i^{inc+\_r}) \tag{30}$$

CU in a cluster cycle then can be defined by using (24) and (30):

$$CU = \left[\frac{T^{inc+\_c\_cycle} - \hat{T}^{ovhd\_c\_cycle}}{T^{inc+\_c\_cycle}}\right].100 \tag{31}$$

As it is now guaranteed that $T_i^{ovhd} = 0$ for every i, every node will fully utilise its allocated τ_i^{inc+} for transmitting application data and control information over the channels. This results in transmitting the same data while removing the slot overhead of previous approaches. INCUS+ slot allocation approach therefore avoids the additional overhead caused by unequal transmission payloads across TDMA cycles.

We will analyse how flexibility improves the performance of the protocol by illustrating its impact on the autonomous vehicle case study in the subsequent sub-sections.

C. IMPACT OF FLEXIBILITY ON OVERHEAD TIME

Every node in the autonomous vehicle system has a unique functionality and, hence, requires a different transmission time. The slot length of a node not only differs from other nodes but also the same node may need different transmission requirements in different TDMA rounds, such as the

TABLE 3. Definition of the terms used in the computational model.

Symbol	Description
T^{fr_r}	Length of FlexRay TDMA round (where $T^{fr\_r_m} = T^{fr\_r_n}$)
T^{inc+_r}	Duration of TDMA round in INCUS+ (where $T^{inc+\_r_m} \,!= T^{inc+\_r_n}$ is possible)
τ^{max}	Slot length of each node in FlexRay approach (where $\tau_i^{max\_r_m} = \tau_j^{max\_r_m}$ and $\tau_i^{max\_r_m} = \tau_i^{max\_r_n}$)
τ^{inc+}	Slot length of each node in INCUS+ (where $\tau_i^{inc+\_r_m} \,!= \tau_j^{inc+\_r_m}$ and $\tau_i^{inc+\_r_m} \,!= \tau_i^{inc+\_r_n}$ is possible)
T_i^{trans}	Transmission time for control information and application data for node i during its allocated node slot in TDMA round
T_i^{idle}	Node slot idle time for node i that is not utilised to transmit application data or control information
T^{ifg}	Time when there is no transmission between frames known as Inter Frame Gap (IFG) overhead time. This is used to accommodate latency and jitter because of propagation delay and clock synchronisation limits etc.
T_i^{ovhd}	Total overhead for node i for its allocated slot
$\hat{T}^{ovhd\_r}$	Total overhead time in a TDMA round
$\hat{T}^{ovhd\_c\_cycle}$	Total overhead time in a cluster cycle
$T^{fr\_c\_cycle}$	Total length of cluster cycle in TTA-based slot allocation approach
$T^{inc+\_c\_cycle}$	Total length of cluster cycle in INCUS+ approach

TABLE 4. Allocated slot length and potential overhead time in the static segment of FlexRay slot allocation approach.

First TDMA Round		
Node	τ^{max}	T_i^{ovhd}
Camera	$75060\mu s$	$(0+4) = 4\mu s$
SFN$_{1,2}$	$75060\mu s$ each	$(75024+4).2 = 150056\mu s$
VCN	$75060\mu s$	$(75032+4) = 75036\mu s$
Second TDMA Round		
Node	τ^{max}	T_i^{ovhd}
Camera	$75060\mu s$	$(0+4) = 4\mu s$
SFN$_{1,2}$	$75060\mu s$ each	$(75024+4).2 = 150056\mu s$
VCN	$75060\mu s$	$(75032+4) = 75036\mu s$
Third TDMA Round		
Node	τ^{max}	T_i^{ovhd}
Camera	$75060\mu s$	$(0+4) = 4\mu s$
SFN$_{1,2}$	$75060\mu s$ each	$(75024+4).2 = 150056\mu s$
VCN	$75060\mu s$	$(75032+4) = 75036\mu s$

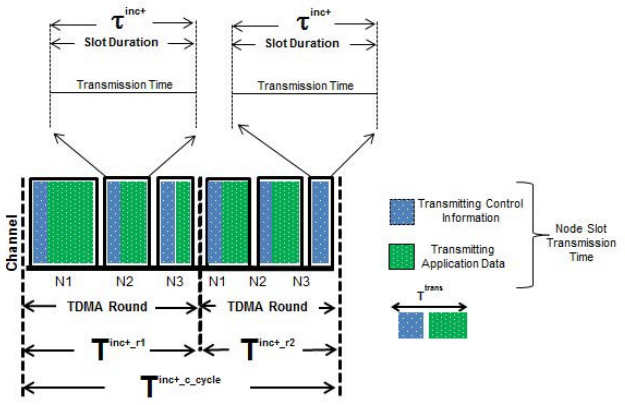


FIGURE 12. INCUS+ slot allocation over a cluster cycle.

camera in this case study. For the FlexRay approach, τ^{max} for each node would have been 75060 microseconds, as required by the camera to transmit maximum size packets (camera I-frames). Therefore, the FlexRay slot allocation approach adds a significant T^{idle} to each node slot that requires less transmission time than assigned, as shown in Fig. 11. Hence, we would get a significant transmission overhead time for most slots.

We can achieve a much better result in our flexible slot allocation approach, INCUS+, as the transmission times of all τ^{inc+} are configured on the basis of their actual transmission requirements in each TDMA round of a cluster cycle. Here, the camera is allocated different transmission slot lengths of 75060 microseconds, 4428 microseconds, and 20420 microseconds for the first, second, and third TDMA rounds respectively. While still statically configured, this slot allocation approach is more flexible and, importantly, eliminates the T^{idle} in each slot as shown in Fig. 12. Therefore, $T^{idle} = 0$ for all node slots over the cluster cycle, which significantly reduces the T_i^{ovhd} for a $node_i$.

One might wonder, what if we take the same fixed and equal length slot allocation approach as demonstrated by FlexRay and TTP but configure it with zero slot idle time by allocating to all nodes the minimum (instead of maximum) slot length required by a node in the whole cluster? As per the case study, VCN is the node that requires 28 microseconds slot length to transmit an initialisation frame. This slot length can be configured for each node in the cluster which eliminates slot idle time. However, we may need fragmentation for the nodes having bigger frames such as Camera and SFN nodes. A fragmentation of a larger frame can be used to accommodate the frame within the duration of transmission slot length and this single frame can be transmitted over multiple TDMA rounds. Only Camera is transmitting three frames with different payload length such as I, B and P frames as discussed above. Let us have a look how many slots are required to transmit an I-frame using this approach. We need 2680 slots to transmit an I-frame by the Camera node. This means, instead of one TDMA round, 2680 TDMA rounds are required to transmit a complete I-frame. Each fragment of an I-frame is transmitted once in a TDMA round to avoid the complexity such as giving more weightage to one node by allowing it to transmit more than once in a TDMA round as compared to other nodes in the membership service. If such node is actually a faulty node then this can disrupt the whole fault tolerance mechanism ensured by the membership service. An extra overhead of 75040 microseconds will be included to append control information in each frame while an extra 10720 microseconds overhead time is required for inter-frame gap (IFG) time as compared to our proposed approach which requires 28 microseconds of control information and 4 microseconds for IFG to transmit an I-frame. Similarly, if we consider B and P frames with multiple fragments and also multiple fragments for SFN node then this transmission overhead time will be huge and may results in significant transmission latency. This approach also decouples the time to detect an error by using membership service. For example, if one or more fragments failed to reach the destination due to any error in the network then the whole message from the transmitter node will be discarded. This fragmentation approach adds significant overhead in system complexity which decouples error handling at protocol level and may need higher level services for fault tolerance.

Allowing a variable slot length for each node on the basis of its payload requirements may, at first glance, appear to result in jitter, caused by data transmission at irregular intervals (due to the unequal TDMA slot lengths). However, this variability is deterministic and known in advance, as all communication schedules are created at design time. We therefore argue that this does not actually constitute jitter (which would cause inherent uncertainties due to its stochastic nature), as here, the exact latency is known in advance. As long as the transmitted data are phase-insensitive, taking into account a simple measure such as the maximum latency is sufficient to allow the system to be designed in the same fashion as with existing TTA-based protocols. If, on the other hand, the transmitted data are phase-sensitive, the exact interval at which data are scheduled to be transmitted (i.e. the timing of the corresponding TDMA slot as designed) needs to be taken into account when interpreting the data. It should be noted that in either case, the maximum latency of our protocol is expected to be better (and guaranteed to be no worse) than that of TTP/C or FlexRay.

D. IMPACT OF FLEXIBILITY ON CHANNEL UTILISATION

Four nodes are connected via the replicated bus[2] (see Fig. 8). In a FlexRay approach, τ^{max} for each node in all TDMA rounds would be 75060μs. Only one node, the camera (during first TDMA round) makes full use of τ^{max} for transmission and all remaining slots, including those of the camera in subsequent TDMA rounds, will have T^{ovhd} in their allocated τ^{max} (see Table 4, Fig. 13). By using four bits time (4 μs) for T^{ifg}; slot length, plus T^{ifg}, is 75064 μs for each node. We have configured each slot with 4 microseconds of Inter-Frame Gap (IFG) time. This accommodates the latency and jitter due to number of factors such as propagation delay and clock synchronisation limits etc. The same IFG time is set for all the existing protocols. Therefore, latency and jitter impact are not considered here as both, existing and proposed protocol is having same time limits to accommodate latency and jitter. According to (25), the first T^{fr_r} is 300256μs where $\hat{T}^{ovhd\_r}$ by using (23) is 225096 μs. Similarly, the values for T^{fr_r} and $\hat{T}^{ovhd\_r}$ in the second and third TDMA rounds are (300256μs, 295728μs) and (300256μs, 279736μs) respectively.

[2]The timing requirements of both replicated buses are exactly the same.

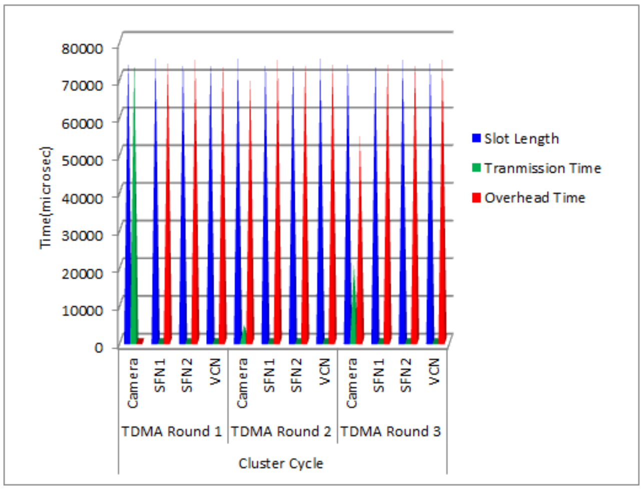


FIGURE 13. Slot length, transmission time and overhead time (where IFG = 4 bits time ≅ 4μs) using FlexRay slot allocation method.

TABLE 5. Allocated slot length and potential overhead time in INCUS+ slot allocation approach.

	First TDMA Round	
Node	τ^{inc+}	T_i^{ovhd}
Camera	75060μs	$(0+4) = 4\mu s$
$\text{SFN}_{1,2}$	36μs each	$(0+4)\cdot 2 = 8\mu s$
VCN	28μs	$(0+4) = 4\mu s$
	Second TDMA Round	
Node	τ^{inc+}	T_i^{ovhd}
Camera	4428μs	$(0+4) = 4\mu s$
$\text{SFN}_{1,2}$	36μs each	$(0+4)\cdot 2 = 8\mu s$
VCN	28μs	$(0+4) = 4\mu s$
	Third TDMA Round	
Node	τ^{inc+}	T_i^{ovhd}
Camera	20420μs	$(0+4) = 4\mu s$
$\text{SFN}_{1,2}$	36μs each	$(0+4)\cdot 2 = 8\mu s$
VCN	28μs	$(0+4) = 4\mu s$

According to (26), the length of cluster cycle in the FlexRay slot allocation approach is 900768μs while the total overhead time according to (24) is 800560μs. Substituting the above-mentioned values in (27), channel utilisation for a cluster is only **11.124%** when following the FlexRay approach.

By comparison, in INCUS+, the τ^{inc+} of each node in each TDMA round is dependent on the transmission requirements (see Table 5, Fig. 14). Therefore the net overhead T^{ovhd} is zero for all nodes in each TDMA round. As T^{ifg} is 4μs and T^{idle} is zero, therefore, by using (30) the length of a cluster cycle in INCUS+ is 100256μs while total overhead time according to (24) is 48μs. Substituting the above-mentioned

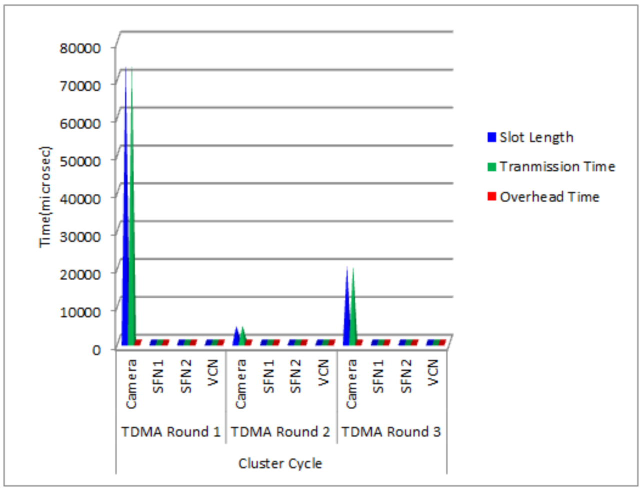


FIGURE 14. Slot length, transmission time and overhead time (where IFG = 4 bits time ≅ $4\mu s$) using INCUS+ slot allocation method.

values in (31) yields a channel utilisation for a cluster cycle in INCUS+ of **99.95**%.

VIII. CONCLUSION

Dependable safety-critical real-time (SCRT) communication protocols are becoming increasingly important and are being used in more complex applications. An example application of SCRT communication protocols is a self-driving car, where the protocol should have the ability to transfer a relatively large amount of safety-critical data (e.g. video frames) in real-time between the various components of an autonomous vehicle.

The TTA approach for SCRT protocols has been to keep the protocol simple and inflexible, to achieve predictability and dependability. However, such an approach results in protocols that suffer from poor bandwidth and channel utilisation, particularly for increasingly complex, safety-critical payload. Attempts have been made to increase flexibility in these protocols, e.g. FlexRay and TTCAN, however, this so far, has only been possible for information that is not safety-critical.

In this paper, we have demonstrated that our SCRT communication protocol, INCUS+, significantly improves channel utilisation over existing TTA-based protocols while guaranteeing atomicity and safety features at the protocol level. INCUS+ achieves increased channel utilisation by allowing the slot length of each node to be configured in accordance with its actual transmission payload requirements for each TDMA round of a cluster cycle. This eliminates node slot idle times for all nodes, hence reduces transmission overhead. Compared to FlexRay which is a TTA-based communication protocol for safety-critical real-time systems, this significantly improves bandwidth utilisation. In our analysis, we have shown that this kind of flexibility makes it possible to reduce the gross overhead time by almost 99%, improving overall bandwidth utilisation efficiency almost nine times compared to the FlexRay approach in an autonomous vehicle system case study. Despite the added flexibility, the level of predictability (predefined schedules for channel access) has been maintained. This is crucial to ensure the safety of the system at the communication protocol level, and for implementing critical services such as the membership service, clock synchronisation, as well as the ability to utilise independent bus guardians. We have shown that our design not only increases flexibility and channel utilisation for safety-critical payload, but also maintains the ability to handle faults in a fail-silent way, as we have formally verified.

REFERENCES

[1] L. M. P. de Almeida, "Flexibility and Timeliness in Fieldbus-Based," Ph.D. dissertation, Univ. Aveiro, Aveiro, Portugal, 1999.
[2] J. P. Thomesse, "A review of the fieldbuses," *Annu. Rev. Control*, vol. 22, pp. 35–45, Jan. 1998.
[3] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*. New York, NY, USA: Springer, 2011.
[4] C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, and C. Temple, "Time-triggered architecture (TTA)," in *Advances in Information Technologies: The Business Challenge*. Amsterdam, The Netherlands: IOS Press, 1997, pp. 758–765.
[5] H. Kopetz, "The time-triggered model of computation," in *Proc. 19th IEEE Real-Time Syst. Symp.*, Dec. 1998, pp. 168–177.
[6] N. Suri, C. J. Walter, and M. M. Hugue, *Advances in Ultra-Dependable Distributed Systems*. Piscataway, NJ, USA: IEEE Computer Society Press, 1994.
[7] *Time-Triggered Protocol TTP/C High-Level Specification, Document Protocol Version 1.1*, TTTech document d-032-s-10-028, 2004.
[8] J. Lee and H. Shin, "A variable bandwidth allocation scheme for ethernet-based real-time communication," in *Proc. 1st Int. Workshop Real-Time Comput. Syst. Appl.*, 1994, pp. 28–32.
[9] B. Andersson, E. Tovar, and N. Pereira, "Analysing TDMA with slot skipping," in *Proc. 26th IEEE Int. Real-Time Syst. Symp.*, Dec. 2005, p. 24.
[10] C. Li, M. Nicholas, and Q. Zhou, "A new real-time network protocol-node order protocol," in *Proc. 11th Real Time Linux Workshop*, 2009, pp. 105–110.
[11] F. Heilmann and G. Fohler, "Impact of time-triggered transmission window placement on rate-constrained traffic in TTEthernet networks," *ACM SIGBED Rev.*, vol. 15, no. 3, pp. 7–12, Aug. 2018.
[12] V. Eramo, F. Lavacca, F. Valente, A. Pisculli, and S. Caporossi, "Simulation and experimental evaluation of a flexible time triggered Ethernet architecture applied in satellite nano/micro launchers," *Aerospace*, vol. 5, no. 3, p. 84, Aug. 2018.
[13] M. Sugihara, "Dynamic slot multiplexing under operating modes for TDMA-based real-time networking systems," *Electronics*, vol. 9, no. 2, p. 224, Jan. 2020.
[14] Z. Luxi, P. Paul, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41803–41815, 2018.
[15] F. Consortium, "FlexRay communications system-protocol specification," *Version*, vol. 2, no. 1, pp. 198–207, 2005.
[16] N. Ragesh, "Data traffic in new generation vehicles," *CSI Commun.*, vol. 35, p. 12, Mar. 2012.
[17] O. E. Marai and T. Taleb, "Smooth and low latency video streaming for autonomous cars during handover," *IEEE Netw.*, vol. 34, no. 6, pp. 302–309, Nov. 2020.
[18] N.-S. N. Ismail, F. Yunus, S. H. Ariffin, A. Shahidan, R. A. Rashid, W. Embong, N. Fisal, and S. Yusof, "MPEG-4 video transmission using distributed TDMA MAC protocol over IEEE 802.15.4 wireless technology," in *Proc. 4th Int. Conf. Modeling, Simulation Appl. Optim.*, Apr. 2011, pp. 1–6.
[19] L. R. Pinto, L. Almeida, H. Alizadeh, and A. Rowe, "Aerial video stream over multi-hop using adaptive TDMA slots," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2017, pp. 157–166.
[20] D. Saxena and V. Raychoudhury, "Design and verification of an NDN-based safety-critical application: A case study with smart healthcare," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 5, pp. 991–1005, May 2019.
[21] A. E. E. Committee, *ARINC 629: IMA Multi-Transmitter Databus Parts 1–4*, Aeronautical Radio, Annapolis, MD, USA, 1990.
[22] AEEC, *Design Guidance for Integrated Modular Avionics*, ARINC, Annapolis, MD, USA, 1997.

[23] S. Berger, "ARINC 629 digital communication system—Application on the 777 and beyond," *Microprocessors Microsyst.*, vol. 20, no. 8, pp. 463–471, 1997.
[24] P. Nutzerorganisation, "Profibus technology and application-system description," Profibus Profinet Int. (PI), Karlsruhe, Germany, Tech. Rep. IEC 61158 Type 3, 2002.
[25] J. Azevedo and N. Cravoisy, *The Worldfip Protocol*, vol. 2, J. De Azevedo and N. Cravoisy, Eds. Antony, France: WorldFIP Int. Tech. Center, 1998.
[26] N. C. Audsley and A. Grigg, "Timing analysis of the ARINC 629 databus for real-time applications," *Microprocessors Microsystems*, vol. 21, no. 1, pp. 55–61, Jul. 1997.
[27] H. Kopetz and R. Nossal, "The cluster compiler—A tool for the design of time-triggered real-time systems," in *Proc. ACM SIGPLAN Workshop Lang., Compilers, Tools Real-Time Syst.*, 1995, pp. 108–116.
[28] J. Rushby, "An overview of formal verification for the time-triggered architecture," in *Proc. Int. Symp. Formal Techn. Real-Time Fault-Tolerant Syst.*, vol. 2469, Lecture Notes in Computer Science Springer, Berlin, Germany: Springer, 2002, pp. 83–105.
[29] H. Kopetz and G. Grunsteidl, "TTP—A protocol for fault-tolerant real-time system," *Computer*, vol. 27, no. 1, pp. 14–23, 1994.
[30] J. Berwanger, M. Peller, and R. Grießbach, "Byteflight a new protocol for safety critical applications," in *Proc. 28th FISITA World Automot. Congr.*, Seoul, South Korea, 2000.
[31] Y. X. Wang, Y. H. Xu, and Y. N. Xu, "Efficient utilization of FlexRay network using parameter optimization method," *Int. J. Eng. Technol.*, vol. 8, no. 6, pp. 439–443, 2016.
[32] J. Dvořák and Z. Hanzálek, "Using two independent channels with gateway for FlexRay static segment scheduling," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1887–1895, Oct. 2016.
[33] T.-Y. Lee, I.-A. Lin, J.-J. Wang, and J.-T. Tsai, "A reliability scheduling algorithm for the static segment of FlexRay on vehicle networks," *Sensors*, vol. 18, no. 11, p. 3783, Nov. 2018.
[34] N. Kumar and A. Mondal, "Timing analysis of precedence constraint messages scheduled with slot multiplexing over dynamic segment of FlexRay," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 222–236, Jan. 2020.
[35] *Road Vehicles-Controller Area Network (Can)—Part 4: Time-Triggered communication*, International Standard Organization, Standard 11898-4, 2000.
[36] *Time-Triggered Ethernet*, SAE International AS6802, 2011.
[37] I. W. Group, *IEEE Standard for Local and Metropolitan Area Networks Timing and Synchronisation for Time Sensitive Applications in Bridged Local Area Networks*, IEEE Standard 802.1AS, 2011.
[38] *IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)*, IEEE Standard 802.1Qat, 2010.
[39] *IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancement for Time Sensitive Streams*, IEEE Standard 802.1Qav, 2010.
[40] *IEEE Standard for Local and Metropolitan Area Networks Audio Video Bridging (AVB) Systems*, IEEE Standard 802.1BA, 2011.
[41] L. Zhao, F. He, and J. Lu, "Comparison of AFDX and audio video bridging forwarding methods using network calculus approach," in *Proc. IEEE/AIAA 36th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2017, pp. 1–7.
[42] E. Heidinger, F. Geyer, S. Schneele, and M. Paulitsch, "A performance study of audio video bridging in aeronautic Ethernet networks," in *Proc. 7th IEEE Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2012, pp. 67–75.
[43] (2017). *Institute of Electrical and Electrnoics Engineers, Time Sensitive Networking*. Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html
[44] (2016). *Institute of Electrical and Electrnoics Engineers, Time Sensitive Networking, 802.1Qbv—Enhancement for Scheduled Traffic*. Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html
[45] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 183–192.
[46] (2017). *Institute of Electrical and Electrnoics Engineers, Time Sensitive Networking, 802.1AS-Rev—Timing and Synchronisation for Time-Sensitive Applications*. Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1AS-rev.html
[47] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44165–44181, 2019.
[48] (2016). *Institute of Electrical and Electrnoics Engineers, 802.1Qci–Per stream Filtering and Policing*. Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/pages/802.1ci.html
[49] (2017). *Institute of Electrical and Electrnoics Engineers, 802.1CB–Frame Replication and Elimination for Reliability*. Time-Sensitive Networking Task Group. [Online]. Available: http://www.ieee802.org/1/files/private/cb-drafts/d2/802-1CB-D2-9.pdf
[50] I. W. Group, *IEEE Standard for Local and Metropolitan Area Network Bridges and Bridged Networks*, Standard 802.1 Q-2018, IEEE Standard 1–1993 pages. 2018.
[51] *IEEE Standard for Local and Metropolitan Area Networks Frame Replication and Elimination for Reliability*, IEEE Standard 802.1CB-2017, 2017.
[52] *Institute of Electrical and Electrnoics Engineers, 802.1Qca—Path Control and Reservation*, Time Sensitive Networking Task Group, USA, 2015.
[53] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 86–94, 2016.
[54] L. Bingqian and W. Yong, "Hybrid-GA based static schedule generation for time-triggered Ethernet," in *Proc. 8th IEEE Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2016, pp. 423–427.
[55] A. M. Kentis, M. S. Berger, and J. Soler, "Effects of port congestion in the gate control list scheduling of time sensitive networks," in *Proc. 8th Int. Conf. Netw. Future (NOF)*, Nov. 2017, pp. 138–140.
[56] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018.
[57] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks," in *Proc. 54th Annu. Design Autom. Conf.*, Jun. 2017, pp. 1–6.
[58] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "ILP-based joint routing and scheduling for time-triggered networks," in *Proc. 25th Int. Conf. Real-Time Netw. Syst.*, Oct. 2017, pp. 8–17.
[59] J. Falk, F. Durr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for TSN with ILP," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 136–146.
[60] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (TSSDN) for real-time applications," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2016, pp. 193–202.
[61] M. Pahlevan, "Time sensitive networking for virtualized integrated real-time systems," Ph.D dissertation, Univ. Siegen, Berlin, Germany, 2020.
[62] E. Kyriakakis, J. Sparsø, and M. Schoeberl, "Implementing time-triggered communication over a standard Ethernet switch," in *Proc. Workshop Fog Comput. IoT*, Apr. 2019, pp. 21–25.
[63] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Veh. Commun.*, vol. 23, Jun. 2020, Art. no. 100214.
[64] F. Arena and G. Pau, "An overview of vehicular communications," *Future Internet*, vol. 11, no. 2, p. 27, Jan. 2019.
[65] K. Kiela, V. Barzdenas, M. Jurgo, V. Macaitis, J. Rafanavicius, A. Vasjanov, L. Kladovscikov, and R. Navickas, "Review of V2X–IoT standards and frameworks for ITS applications," *Appl. Sci.*, vol. 10, no. 12, p. 4314, 2020.
[66] Q. Pei, B. Kang, L. Zhang, K.-K.-R. Choo, Y. Zhang, and Y. Sun, "Secure and privacy-preserving 3D vehicle positioning schemes for vehicular ad hoc network," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, pp. 1–12, Dec. 2018.
[67] *Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard ISO/IEC/IEEE 8802.11(TM), 2012.
[68] *802.11p PART 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 7: Wireless Access in Vehicular Environment*, IEEE Standard 802.11p-2010, 2010.
[69] K. Sjoberg, E. Uhlemann, and E. G. Strom, "How severe is the hidden terminal problem in VANETs when using CSMA and STDMA?" in *Proc. IEEE Veh. Technol. Conf. (VTC Fall)*, Sep. 2011, pp. 1–5.
[70] D. Chen, R. Hexel, and F. R. Raja, "INCUS: A communication protocol for safety critical distributed real time systems," in *Proc. 20th Asia–Pacific Conf. Commun. (APCC)*, Oct. 2014, pp. 309–314.
[71] G. Bauer and M. Paulitsch, "An investigation of membership and clique avoidance in TTP/C," in *Proc. 19th IEEE Symp. Reliable Distrib. Syst. (SRDS)*, Oct. 2000, pp. 118–124.

[72] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *IEEE Trans. Comput.*, vols. C–36, no. 8, pp. 933–940, Aug. 1987.

[73] H. Pfeifer, D. Schwier, and F. W. von Henke, "Formal verification for time-triggered clock synchronization," in *Proc. Dependable Comput. Crit. Appl.*, Jan. 1999, pp. 207–226.

[74] J. Rushby, "Formal verification of transmission window timing for the time-triggered architecture," Deliverable 24b, SRI Project, Comput. Sci. Lab., SRI Int., Menlo Park, CA, USA, Tech. Rep. 11003, 2001.

[75] T. Hase, W. Hintermaier, A. Frey, T. Strobel, U. Baumgarten, and E. Steinbach, "Influence of Image/Video compression on night vision based pedestrian detection in an automotive application," in *Proc. IEEE 73rd Veh. Technol. Conf. (VTC Spring)*, May 2011, pp. 1–5.

[76] W. Kubinger, H. Hemetsberger, and J. Kogler, "Platooning platform for analyzing embedded control algorithms," in *Proc. Ann. DAAAM*, 2005, pp. 211–213.

[77] S. Ramberger, W. Herzner, E. Schoitsch, and W. Kubinger, "TTIPP3—A fault-tolerant time-triggered platooning demonstrator," in *Proc. Int. Workshop Intell. Solutions Embedded Syst.*, 2008, pp. 1–11.

[78] J.-E. Kallhammer, D. Eniksson, G. Granlund, M. Felsberg, A. Moe, B. Johansson, J. Wiklund, and P.-E. Forssen, "Near zone pedestrian detection using a low-resolution FIR sensor," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2007, pp. 339–345.

[79] D. Chen, R. Hexel, and F. R. Raja, "Engineering real-time communication through time-triggered subsumption," in *Proc. 11th Int. Conf. Eval. Novel Softw. Approaches Softw. Eng.*, 2016, pp. 272–281.

FAWAD RIASAT RAJA received the B.Sc. degree (Hons.) in software engineering and the M.S. degree in computer engineering from the University of Engineering and Technology Taxila, Pakistan, in 2006 and 2010, respectively. He is currently pursuing the Ph.D. degree with the School of Information and Communication Technology (ICT), Griffith University, QLD, Australia. His research interests include communication protocols and networking, embedded systems, real-time systems, and safety-critical systems.

DAVID CHEN received the Ph.D. degree in distributed collaborative systems from Griffith University, Australia, in 2001. He is currently a Senior Lecturer with the School of Information and Communication Technology, Griffith University. He holds the roles of the Head of the Discipline in IT and IS and the Program Director of Bachelor of Information Technology. His research interests include distributed and real-time systems, consistency maintenance in collaborative systems, bioinformatics, and web technologies.

RENÉ HEXEL (Senior Member, IEEE) received the M.Sc./Eng. degree, in 1995, and the Ph.D. degree, in 1999. He has been a leading Researcher in software engineering of safety-critical real-time systems. He is currently the Deputy Head of the School of ICT, Griffith University, and the Co-Director of the Machine Intelligence and Pattern Analysis Laboratory (MiPal), Institute for Integrated and Intelligent Systems (IIIS), Brisbane, Australia. His achievements include methodologies for modeling, model checking, evaluation, and validation of safety-critical real-time systems, including failure mode effects analysis (FMEA) through fault injection. His research interests include software engineering of complex distributed systems and autonomous robots as well as robust and reliable safety critical systems.