INCUS: A Communication Protocol for Safety Critical Distributed Real Time Systems

David Chen, René Hexel, and Fawad Riasat Raja
School of Information and Communication Technology
Griffith University, Nathan, QLD 4111, Australia
{d.chen,r.hexel}@griffith.edu.au
fawad.raja@griffithuni.edu.au

***Abstract*—Safety-critical, distributed real-time systems, such as avionics, automotive or factory automation and control systems, and the like, require efficient communication mechanism between their nodes in order to deliver information within defined time frames. Protocols that follow the time-triggered architecture paradigm guarantee timeliness under a given fault and load hypothesis through the use of a stringent, equivalently-spaced time division multiple access (TDMA) scheme. This, however, comes at the cost of poor channel and bandwidth utilisation in real-world scenarios where functionality and transmission requirements often differ considerably between nodes. Here, we propose a new approach and protocol, INCUS, that allows the slot length of nodes to be configured in accordance with their payload requirements. We show the feasibility of our approach while retaining the level of reliability required for safety-critical real-time systems. Our analysis shows an almost twofold improvement in efficiency in a typical automotive, brake by wire scenario.**

I. Introduction

Efficient and reliable communication among different nodes that are connected with each other through a shared communication channel is a crucial part in distributed real-time systems. Timely delivery of information from source nodes to destination nodes is an important aspect of distributed real time communication systems. This is particularly true for safety-critical systems, including control systems for avionics, automotive electronics system, factory automation systems, and the like. In such scenarios, intact and error-free delivery of information crucially needs to occur within the strict temporal deadlines imposed by the system. In addition to timeliness, fault tolerance becomes another important requirement for any safety-critical system. When nodes are sharing a single communication channel to transmit their messages, collisions can occur on the communication channel if multiple nodes are trying to transmit at the same time. Retransmission after a collision will result in a significant delay in message delivery that is not acceptable for any hard real-time critical system, as missing a deadline (delivery of a message outside its defined time frame) may have catastrophic consequences.

A number of techniques exist in traditional, event-driven communication architectures to avoid collisions. A typical example are Carrier Sense Multiple Access (CSMA) schemes [1] based on collision detection or collision avoidance. In a collision avoidance system, each node has to sense traffic on the channel and wait until there is no traffic on the channel, while in a collision detection scheme, after detecting a collision all nodes have to back off (terminate their transmission) and retransmit after waiting for some random amount of time. Due to their probabilistic nature, these techniques result in unpredictable channel access (and consequently, delivery) latency and are therefore only suitable for non-safety-critical systems, where best-case message delivery times can differ greatly from worst case scenarios and missing deadlines is tolerable. Such protocols and techniques are not suitable for safety-critical systems in which timeliness is compulsory.

The Time Triggered Architecture (TTA), introduced by C. Scheidler et al. [2], and now in widespread use in industry, guarantees that communication that occurs within a system's specified fault hypothesis arrives within its specified deadline. In the TTA, channel access is based on a Time Division Multiple Access (TDMA) scheme where communication time slots for nodes to access the channel are scheduled offline. All nodes use synchronised clocks and know exactly at what point of time each node will transmit its message, guaranteeing that channel access will be free of collision. Nodes are allocated static and equal-length time slots in which they can transmit their information in each TDMA-round (the sequence of transmission slots for all the nodes). After each round, the cycle repeats.

This level of determinism, however, comes at a cost. Although communication is guaranteed to be collision free and existing TTA protocols have a known channel access latency, they have the disadvantage of poor channel utilisation. As each node has different functionality and, consequently, different transmission payload requirements, allocating the equal-length slots to all nodes inevitably causes inefficiencies as nodes with lower payload will not be able to fill their allotted time slots with meaningful data. While some approaches exist that try to overcome this, we are not aware of any that have succeeded in retaining the deadline and fault-tolerance guarantees required for safety-critical systems [3][4].

The objective of this paper is to present a solution to overcome the issue of inefficiency of existing TTA-based protocols while retaining the level of fault tolerance and reliability required for safety-critical systems. The remainder of this paper is organised as follows. In Section II we briefly discuss existing TTA-based communication protocols and their properties. Section III introduces INCUS and our proposed approach for efficient channel utilisation. We analyse and

compare the efficiency of our protocol in Section IV and provide an outlook and conclusion in Section V.

II. Communication Protocols

We now give a brief overview of communication protocols for safety-critical real-time systems. ARINC-629 [5] is a communication protocol for a bidirectional and multiple access data bus and is capable of transmitting safety-critical and non-safety-critical information. It is largely used in avionics systems for communication between different control units (nodes). Channel access is controlled by using a transmission gap (TG) timer. The value of the TG is different for each node to prevent multiple nodes from transmitting at the same time. If the TG of a node has elapsed it will listen on communication channel and, if the channel is idle, the node will start transmitting. No other node can transmit if there is traffic on the channel even its TG has elapsed. In this case, the TG will recommence and the procedure restarts afresh. This scheme does not prevent starvation nor does it provide fault tolerance at the protocol level.

The Time Triggered Architecture (TTA) [2] aims at addressing these issues. The Time-Triggered Protocol (TTP) [6] implements TTA communication through a TDMA scheme with statically configured timing. All nodes participating in the system are allowed to access the channel on their turn for a fixed period time slot. The duration is the same for every node's time slot. A fault hypothesis can be defined in TTP for reliable communication in the presence of different kind of faults, including omission failures, crash failures, and channel failure. Fault tolerance can be guaranteed through replication and replica determinism between redundant nodes (including shadow nodes, if present). Additionally, babbling idiot faults are prevented through use of an independent bus guardian. TTP ensures correct timing through distributed clock synchronisation, however, due to the requirement of having equal-length time slots, nodes with fewer data to transmit will not be able to fully utilise their allotted time slot.

FlexRay [7] is an automotive network communication protocol developed by the FlexRay Consortium [8]. It can be viewed as the combination of TTP and ByteFlight [9]. The basic communication channel access mechanism is the same as in TTP, however, it adds flexibility by dividing the TDMA round into two parts. In the first part, all nodes are allocated static and equal length node slots to transmit safety critical information, while the second part is based on dynamic slots (inherited from the ByteFlight protocol) to transmit non-safety critical information. Channel bandwidth in the non-safety-critical part is shared on-demand among the nodes to ensure better bandwidth utilisation. FlexRay utilises similar fault-tolerance mechanisms to TTP for the safety-critical part of communication. Consequently, channel utilisation for safety-critical information from nodes that do not require the maximum slot length for transmission is equally bad.

An approach to improve channel utilisation is proposed in TDMA with slot-skipping (TDMA/SS) [3]. In this approach if a node is not transmitting in its slot for some amount of time after the start of its slot duration then its allocated slot will be skipped and the next node will be allowed to transmit its information earlier than its scheduled time. While this and similar approaches [4] achieve better channel utilisation, they do so at the expense of transmission determinism required for distributed agreement, a fundamental requirement for fault tolerance in the TTA [2].

III. INCUS: Principle of Operation

All the existing TTA communication protocols for safety-critical, distributed real-time systems are using a TDMA approach for channel access where static and equal length node slots are used in each TDMA round for transmitting the safety critical information as shown in Fig. 1. Here, we will use the term *traditional approach* for such an approach. We argue

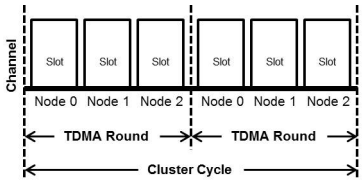

Fig. 1. TDMA based approach for channel access

that, while care must be taken to facilitate fault tolerance and ensure replica determinism, assigning same length node slots will cause low channel utilisation and will affect the efficiency of the communication protocol. This is due to each node in the system typically having different functionality and, thus, transmission requirements. We use the Brake-by-Wire (BBW) case study [10] to illustrate this problem further.

X-by-wire systems such as steer-by-wire, brake-by-wire, and the like, are nowadays starting to become more prevalent in the automotive industry. These systems are designed to replace mechanical components with more sophisticated electrical components including sensors and electrical actuators. The goal is to make these systems more reliable and fault tolerant than traditional mechanical systems (and it is expected for them to also become cheaper, once used in a higher number of mass-produced vehicles). The Anti-lock-Braking System (ABS) widely used in modern vehicles is an example application that can sit on top of a brake-by-wire system. ABS provides more safety to drivers by avoiding wheel lock-up and uncontrolled skidding. It also decreases the stopping distance on dry and slippery roads. The layout of a typical brake-by-wire system is shown in Fig. 2. Each wheel has a wheel speed sensor and a brake actuator. The function of the Brake-by-Wire-Manager (BBWM) is to use the brake pedal sensor value from the pedal node (BPN) and the speed of each wheel from the wheel speed sensor nodes (WSSNs) to calculate the brake force for each brake actuator node (BAN). Each BAN will take this brake force value and apply it to their corresponding wheel through their brake actuators. The BBWM periodically monitors the wheel speeds reported by the WSSNs and checks the difference between them. If a wheel is about to lock, the BBWM will send a lower brake force

to that wheel's BAN. Wheels that spin faster than others will have a stronger brake force applied. To provide fault tolerance at hardware level, two redundant BBWM are used. However, the WSSNs and BANs are not replicated, as it is possible to brake the car with the remaining wheels, even if one wheel unit fails. The whole communication among BBWM, BPN, WSSN and BAN is based on replicated communication channels with a bandwidth of 1 Mbps. The BAN for each wheel is not transmitting any application data, but control information to show its membership on the network is still transmitted. Table I shows the payload data and ideal transmission lengths

TABLE I
FRAME SIZES FOR THE NODES OF THE BRAKE-BY-WIRE SYSTEM

Node	Data	Frame Size	Slot Length
BBWM	12 bits	$12 \cdot 4 + 28 = 76$ bits	76 μs
BPN	10 bits	$10 + 28 = 38$ bits	38 μs
WSSN	10 bits	$10 + 28 = 38$ bits	38 μs
BAN	nil	$0 + 28 = 28$ bits	28 μs

for all the nodes in the Brake-by-Wire system. We also take into account 28 bits of control information with the same functionality as in TTP [6] (a 1-bit frame identifier, 3 bits of mode information, and a 24-bit CRC value). Each type of node in the BBW system requires a different node slot length. If we use the traditional TTA slot allocation approach, the slot length for each node will need to be the maximum 76 microseconds required by the BBWM. All other nodes transmit only half the data or less and would have to pad their node slot to the static, maximum length. No other node can transmit during that padding time, so we call this the node slot overhead time as illustrated in Fig. 3.

We propose an alternate protocol called **INCUS** (Individually Node Slot CUStomizable protocol) that configures node slot length in accordance with the transmission requirements of a node. Rather than allocating same-length slots to all nodes, each slot length is configured on the basis of its transmission data requirements. Consequently, node slots vary

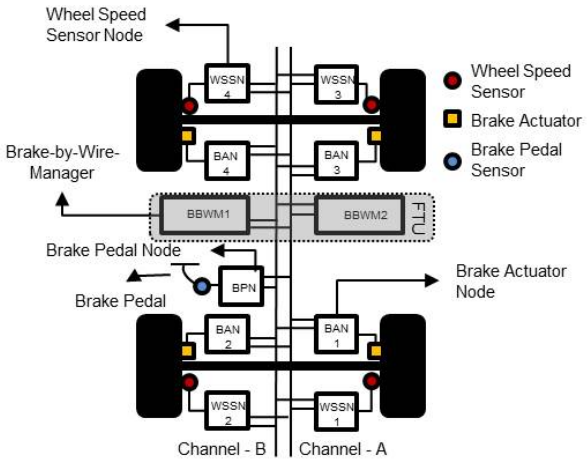

Fig. 2. Brake-by-wire Architecture

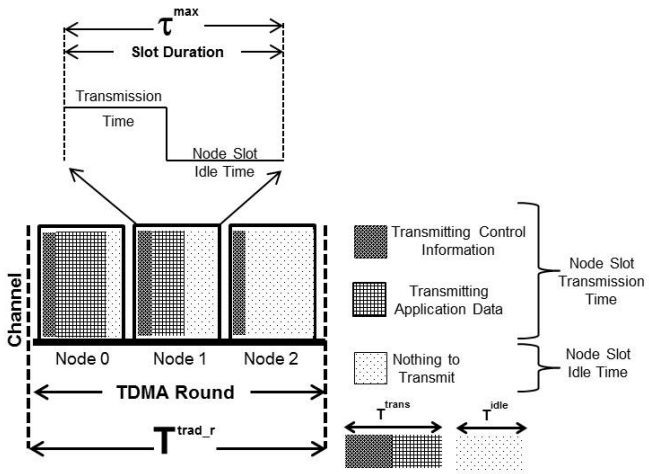

Fig. 3. Traditional slot allocation approach

in their length within a TDMA round as shown in Fig. 4. This concept eliminates the overhead time of a node during transmission and improves the overall efficiency of channel utilisation (as shown in Section IV).

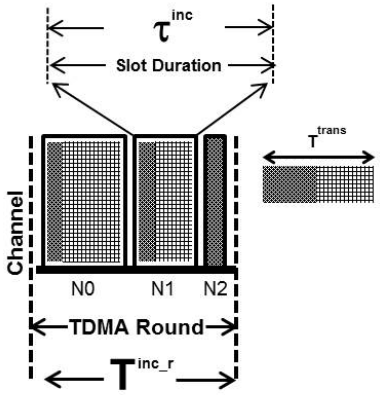

Fig. 4. INCUS slot allocation approach

The principle of operation of INCUS is based on TTP [6]. The communication controller subsystem of a node that is actually responsible for transmission and reception of data over the communication channel operates under similar principles. The *Message Descriptor List* (MEDL, a data structure within controller memory) holds the time schedule for the data transmission and data reception phase for all the nodes. Each node has a replicated copy of the MEDL, hence it knows the exact time when it and other nodes have the right to access the communication channel. Depending on the nature of transmission, three classes of frames are used. Normal frames (N-frames) carry application data, initialisation frames (I-frames) carry synchronisation information for reintegration of recovering nodes, and coldstart frames (CS-frames) carry synchronisation information for the integration of nodes during

system start-up. Node startup and resynchronisation is based on TTP [11, p. 48], but INCUS requires different values of timeout parameters. All nodes have a unique listen timeout (time for a node to wait for I-frames) and coldstart timeout (time for a node to wait for a response from other nodes on the network after transmitting a CS-frame), because of their unique startup delay defined in Equation 1.

$$T_0^{start} < T_1^{start} < .... < T_i^{start} < ... < T_{n-1}^{start} \quad (1)$$

In INCUS, a startup delay for a $node_i$ is the duration of all *incus-slots* from the beginning of the *incus-round* up to the beginning of the *incus-slot* of $node_i$. The term T^{inc_r} (*incus-round*) denotes a TDMA round in INCUS and τ^{inc} (*incus-slot*) represents a specific node slot. It is important to note that this differs from TTP in that $\tau_i^{inc} \neq \tau_j^{inc}$ is now possible! In TTP, the listen timeout parameter for a recovering node i during the integration process is $T_i^{start} + 2T^{trad\_r}$ with T^{trad_r} being the time for a traditional TDMA round. This may result in the frequent transmission cold-start frames in case node i did not receive any I-frames for up to two traditional-rounds. In INCUS, we bound the listen timeout by the number of incus-rounds in a cluster cycle.

$$T_i^{listen} = T^{c\_cycle} + T_i^{start} \quad (2)$$

where the cluster cycle time for r incus-rounds is

$$T^{c\_cycle} = \sum_{i=1}^{r} T_i^{inc\_r} \quad (3)$$

such that the incus cold-start time is

$$T_i^{inc\_cold} = T^{inc\_r} + T_i^{start} \quad (4)$$

I-frame Transmission. TTP has to transmit I-frames on both channels after two traditional rounds. INCUS can accommodate a more flexible approach in transmitting I-frames. E.g., if there are some nodes such as actuator nodes in the network that are not transmitting any data at all (other than control information to show their presence on the network) then I-frames can also be transmitted by these nodes. This may be desirable for faster reintegration of recovering nodes, as they will receive I-frame in each incus round. Conversely, any two nodes can be statically configured to transmit I-frame only once, in each cluster cycle as defined in Eq. (3), transmit I-frames on both channels, reducing the overhead from I-frame transmissions.

To facilitate fault tolerance and keep track of active and inactive nodes within a round, we implement a TTA membership service [2], by recording the membership status of each node in a membership vector whose bit size is equal to the number of nodes in the cluster. It is important to note that, following the TTA, this comes without any additional acknowledgement overhead or timeout requirements, as each node transmits once per incus-round. For every slot, the transmission of a node (or lack thereof) is registered in the membership vector. As in TTP, the membership vector does not have to be transmitted explicitly, but is used in the frame CRC calculation, causing dissenting minority nodes (those with a differing view from the majority) to no longer be heard (and no longer being able to receive frames from the majority of nodes within the cluster). The membership vector of such minority nodes will quickly drop to half the nodes in the system or below, causing them to restart and re-integrate into the cluster [6].

Clock synchronisation is a core requirement of time-triggered communication protocols. All the nodes have physical clocks (internal or local clocks) and a predetermined schedule defined in their own copy of the MEDL. Each node operating in the system times its slots (transmit or receive actions) in accordance with a predetermined schedule that is triggered by the progression of time. For these node slots to be synchronised, the nodes' clocks have to be synchronised. Clock synchronisation in INCUS is based on TTP [11] and follows the same principles. Hence, we will first describe clock synchronisation in TTP before discussing the differences in INCUS. Any clock skew between sender and receiver is identified at the receiver side by using the difference between the expected arrival time defined in the MEDL and the actual arrival time of a frame. The slots that are to be used for clock synchronisation are denoted in the MEDL (e.g., transmissions from nodes with cheaper, less accurate clocks may be disregarded). These slots are called resynchronisation points R_p. The Fault Tolerant Average (FTA) [12] is then used to correct the local clock based on the measured deviations. There are two phases to perform clock synchronisation. In the first phase, the difference is stored in a sorted array with a size of four where the highest and lowest value are discarded and then the average of the remaining two values is calculated. This average value is used as the correction term to adjust the local clock's time. If resynchronisation point is set after *n* number of slots then duration of R_p can be calculated in TTP as follows: let *t* be the the real time such that:

$$R_p(t) = n * \tau^{max} \quad (5)$$

where τ^{max} is the node slot length in TTP.

As in TTP all nodes have same length, the resynchronisation interval can be calculated by using Equation (5). For INCUS, the nodes can have different length slots, so we need to adjust the formula as follows:

$$R_p(t) = \sum_{i=0}^{n-1} S_i(t) \quad (6)$$

where $(n \geq 4)$ and $S(t)$ is the slot length of each node.

With the membership and distributed clock synchronisation mechanism described here, INCUS provides the basis for fault tolerance through node replication and ensures replica determinism. Use of duplicated communication channels makes it fault tolerant in-case of a single channel failure. Furthermore, we can utilise the additional mechanisms in the TTA, such as bus guardians, to prevent a "babbling idiot" node from transmitting outside its allotted slot [2]. Hence, in terms of fault tolerance and other safety critical features, INCUS is on par with TTP.

IV. Efficiency Analysis for Channel Utilisation

In order to analyse the channel utilisation in the existing traditional approach and compare it with INCUS, we need to define the timing terms we will use in our analysis.

- T^{trad_r} represents (the length of) one TDMA round using the traditional slot allocation approach.
- T^{inc_r} represents one TDMA round in our proposed slot allocation approach for INCUS.
- τ^{max} represents the node slot length in the traditional slot allocation approach (same for all the nodes).
- τ_i^{inc} represents the slot length for each node in INCUS (possibly different for each node i).
- T_i^{trans} is the transmission time for control and payload data for node i during its allocated node slot.
- T_i^{idle} is the remaining allocated slot time for node i not utilised for data or control information.
- T^{ifg} is the Inter Frame Gap (IFG) overhead time, i.e., the time when no transmission occurs between frames.
- T_i^{ovhd} is the total overhead for node i for its allocated slot.

A. Traditional slot allocation approach

Due to equal length slot allocation to all the nodes, we can say that slot length (τ^{max}) of each node consists of node slot transmission time (T_i^{trans}) and a node slot overhead time (T_i^{ovhd}) as shown in Fig. 3. Node slot overhead time of a $node_i$ can be calculated as:

$$T_i^{idle} = \tau^{max} - T_i^{trans} \tag{7}$$

$$T_i^{ovhd} = T_i^{idle} + T^{ifg} \tag{8}$$

If T_i^{ovhd} is the overhead time in each node slot then total overhead time ($\hat{T}^{ovhd}$) for ***n*** number of nodes in a TDMA round is:

$$\hat{T}^{ovhd} = \sum_{i=0}^{n-1} T_i^{ovhd} \tag{9}$$

If there are ***n*** number of nodes then the length of a TDMA round (T^{trad_r}) in traditional slot allocation approach is defined as:

$$T^{trad\_r} = n \cdot (\tau^{max} + T^{ifg}) \tag{10}$$

By using equation (9) and equation (10) channel utilisation (CU) in a TDMA round for traditional slot allocation approach can be calculated as:

$$CU = \frac{T^{trad\_r} - \hat{T}^{ovhd}}{T^{trad\_r}} \tag{11}$$

B. INCUS approach

In our proposed approach we customise the slot of each node on the basis of its transmission load to avoid the node slot overhead time within the allocated node slot as shown in Fig. 4. Therefore, for slot length τ^{inc} of each node, T_i^{idle} is zero and the slot length of a $node_i$ is:

$$\tau_i^{inc} = T_i^{trans} \tag{12}$$

If there are n number of nodes then length of a TDMA round (T^{inc_r}) in our proposed slot allocation approach can be defined as:

$$T^{inc\_r} = \sum_{i=0}^{n-1} (\tau_i^{inc} + T^{ifg}) \tag{13}$$

Channel utilisation in a TDMA round for our proposed approach can be calculated by using equations (9) and (13):

$$CU = \frac{T^{inc\_r} - \hat{T}^{ovhd}}{T^{inc\_r}} \tag{14}$$

As there is no node slot overhead time in our proposed approach, every node will fully utilise its allocated node slot time for transmitting data and control information over the communication channels. Hence, INCUS can transmit same amount of data in less time than the traditional approach. In the exceptional case where the payload requirements for all the nodes are the same, traditional slot allocation overhead is equal to INCUS. In all other cases, INCUS avoids the additional overhead caused by unequal payloads.

TABLE II
Brake-by-wire System Channel Utilisation using Traditional Slot Allocation

Node	τ^{max}	T_i^{ovhd}
$BBWM_{1,2}$	76μs each	$(0+4)\cdot 2 = 8\mu s$
BPN	76μs	$(38+4)\cdot 1 = 42\mu s$
$WSSN_{1\ldots4}$	76μs each	$(38+4)\cdot 4 = 168\mu s$
$BAN_{1\ldots4}$	76μs each	$(48+4)\cdot 4 = 208\mu s$
Total	880μs	426μs

C. System Level Analysis

Now we will apply our analysis regarding channel utilisation to the Brake-by-wire case study. There are a total of eleven nodes connected through a bus[1] as shown in Fig. 2. According to traditional slot allocation approach slot length τ^{max} of each node should be 76μs. Only two nodes, $BBWM_1$ and $BBWM_2$ will fully utilise their slot length for transmitting information over the communication channel and rest of the nodes will have a node slot overhead time in their allocated slot length as shown in Table II. We are using 4 μs for T^{ifg}, therefore a slot length, plus T^{ifg}, is 80 μs for each node. According to equation (10) T^{trad_r} is 880μs where $\hat{T}^{ovhd}$ by using equation (9) is 426 μs. By substituting these values in equation (11), channel utilisation for a TDMA round according to traditional slot allocation approach is **51.59**%. In the traditional slot allocation approach, the 454 bits of payload data are transmitted in a TDMA round of 880 μs.

By comparison, in INCUS, the slot length τ^{inc} of each node depends on its transmission requirements as shown in Table III. Therefore, T_i^{ovhd} is zero for all nodes. As T^{ifg} is 4 μs and T_i^{idle} is 0 μs, according to equation (9) $\hat{T}^{ovhd}$ is only a total of 44μs in INCUS. This constitutes only 10.3% of

[1]For simplicity, we ignore the replicated bus here – however, it is important to note that the timing requirements for the redundant bus are exactly the same.

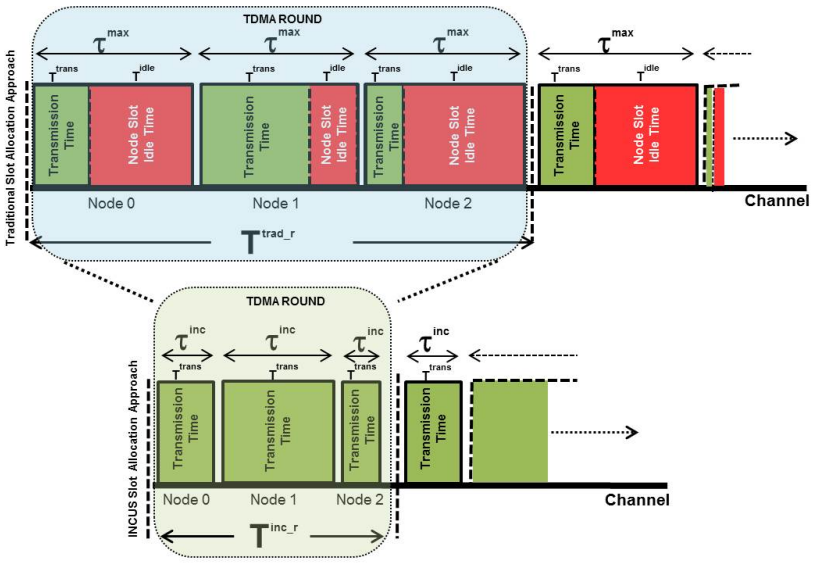

Fig. 5. INCUS vs Traditional approach

TABLE III
CHANNEL UTILISATION FOR THE BRAKE-BY-WIRE SYSTEM USING INCUS SLOT ALLOCATION

Node	τ^{inc}	T_i^{ovhd}
BBWM1-2	$76\mu s$ each	$(0+4)\cdot 2 = 8\mu s$
BPN	$38\mu s$	$(0+4)\cdot 1 = 4\mu s$
WSSN1-4	$38\mu s$ each	$(0+4)\cdot 4 = 16\mu s$
BAN1-4	$28\mu s$ each	$(0+4)\cdot 4 = 16\mu s$
Total	$498\mu s$	$44\mu s$

the overhead of the fixed maximum-length slot allocation approach shown before. Consequently, following equation (13), T^{inc_r} now totals $498\mu s$ per TDMA round (compared to $880\mu s$ in a traditional protocol such as TTP). By substituting these values in equation (14), our channel utilisation for each TDMA round is **91.16**% when using INCUS slot allocation. Fig. 5 illustrates the comparison between the traditional and INCUS slot allocation approaches.

V. CONCLUSION

In this paper, we have presented a real-time communication protocol for safety critical distributed system, called INCUS. While INCUS is based on the traditional TDMA scheme utilised in the Time Triggered Architecture, it significantly improves bandwidth utilisation over the traditional schemes. We achieve this by allowing the slot length of nodes to be configured in accordance with the actual payload requirements of these nodes. In our analysis, we have shown that INCUS reduces the gross overhead time by almost 90%, improving overall bandwidth utilisation efficiency almost twofold in a typical automotive brake-by-wire system scenario. Unlike other protocols that achieve similar efficiency gains only through the abolition of features that are vital for safety-critical systems, such as a zero acknowledgement membership and agreement protocol, we are able to retain these in our protocol. Consequently, INCUS retains the level of reliability required for safety-critical systems.

REFERENCES

[1] *ANSI / IEEE Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification*. 345 East 47th Street, New York NY 10017: The Institute of Electrical and Electronic Engineering, Inc., 1985.

[2] C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, and C. Temple, "Time-triggered architecture (TTA)," *Advances in Information Technologies: The Business Challenge*, pp. 758–765, 1997.

[3] B. Andersson, E. Tovar, and N. Pereira, "Analysing tdma with slot skipping," in *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, 2005.

[4] C. Li, M. Nicholas, and Q. Zhou, "A new real-time network protocol - node order protocol," in *Proceedings of 11th Real Time Linux Workshop*, 2009.

[5] A. E. E. Committee *et al.*, "Arinc 629: Ima multi-transmitter databus parts 1-4," *Aeronautical radio, Inc., Annapolis, Maryland.–Octobre*, 1990.

[6] H. Kopetz and G. Grunsteidl, "TTP – a protocol for fault-tolerant real-time systems," *Computer*, vol. 27, no. 1, pp. 14–23, 1994.

[7] F. Consortium *et al.*, "Flexray communications system-protocol specification," *Version*, vol. 2, no. 1, pp. 198–207, 2005.

[8] Berwanger, "et al. flexray the communication system for advanced automotive control systems," *SAE Transactions*, vol. Vol. 110(7), pp. SAE Press, pp. 303–314, 2001.

[9] J. Berwanger, M. Peller, and R. Grießbach, "Byteflight a new protocol for safety critical applications," in *Proceedings of the 28th FISITA World Automotive Congress. Seoul, Korea: FISITA*, 2000.

[10] B. Hedenetz and R. Belschner, "Brake-by-wire without mechanical backup by using a TTP-communication network," SAE Technical Paper, Tech. Rep., 1998.

[11] "Time-triggered protocol TTP/C high-level specification, document protocol version 1.1, tttech document number d-032-s-10-028," 2004.

[12] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 933–940, 1987.