**P Thirumeni**. Date: 03.07.23

Ph.D. Student – 203085002

**Research Proposal.**

**"Advancing Real-Time Communications in CAN Networks: A Comprehensive Research, Development, and Validation of a Novel Adaptive CAN Message Arbitration and Scheduling Algorithm for Distributed Embedded Systems."**

**Literature survey.**

**Introduction.**

Real-Time Communications using Controller Area Networks (CAN) [1] remain crucial in various industrial applications, and the legacy (particularly in the industrial and aerospace domain) nature of the CAN protocol restricts certain systems from undergoing major changes. Hardware alterations and the high costs associated with alternative protocols like AFDX and FlexRay limit their adoption. CAN networks typically utilize message IDs with priorities for arbitration and message transmission.

Traditionally messages which are having important information (for example, engine firing sequence data) and messages which need to be addressed immediately (for example, alarms, safety trips, or triggering commands, etc.) are given higher priority, and regular noncritical messages (example: infotainment ON/OF control in a car, cabin temperature, etc.,) are assigned with lower priority IDs, such manual fixed ID allotment was good enough for networks with predictable behavior and time sequences.

To deal with real time deterministic message transactions, other variations of CAN was designed, like Time triggered CAN – TTCAN [2], CAN FD etc., however time sliced CAN, even though addressed some issues, requires additional synchronisation mechanism, and increased overhead, also legacy CAN networks cannot be part of this new time triggered version without major changes in hardware and software. Time-triggered CAN does not always improve the system performance, TTCAN has several drawbacks [3], some of them are, the complexity of the network compared to traditional CAN, and implementing TTCAN requires precise time synchronisation among all nodes, since TTCAN operates on a rigid time schedule, with fixed time slots allocated to each node, this fixed schedule limits the system's ability to adapt to changing communication requirements or handle sporadic events efficiently. The network is less scalable with increasing node count, and If a node fails or gets disconnected from bus, it may disrupt the entire time schedule, leading to potential network partitioning and communication issues. Due to the idle time of unused timeslots of some nodes, may result in lower overall bus utilization.

Mainly legacy systems cannot adapt to the time-triggered version with major changes, so improving traditional CAN networks with better design is one of the solutions.

A more systematic approach for the allotment of CAN message IDs, is by using a message scheduler with some procedures and rules.

**Communication model:**

A basic communications model and comparative notations of CAN message scheduling in analogous with the operating system-CPU tasks are analysed in this paper, “Analysing Real-Time Communications: Controller Area Network [6].

In the context of a CAN network, messages arbitrate or compete for access to the CAN bus to transmit among network nodes. Analogously, transmitting a message onto the CAN bus is similar to the CPU task scheduling, with the CAN bus's time slots for transmitting its frame, equating to the CPU's processing time slices.

The objective (in both CPU task scheduling and in CAN message transmission scheduling to CAN bus) is efficient resource management to achieve timely execution or transmission. The CAN bus parallels the CPU as a shared resource, where CAN messages (analogues to CPU tasks) are scheduled to meet their desired outcomes within deadlines.

These constraints are analysed in the paper [6], as worst-case response time,

*Rm* *Jm* *wm* *Cm,*

Where *Jm* is queuing jitter (maximum difference in message arrival latency) in the CAN node’s buffer, and Rm is the worst case response time, Cm is the longest time taken to transmit CAN message ‘m’. including CAN frame protocol bits (CRC,ACK etc.,) overload.

Wm is represents the worst-case queuing latency, the longest time between placing the message in a priority ordered queue, and the message commencing transmission from the Node.

The Wm is given by

*wm= Bm +*j ∈ hp(m) [(*w*m+Jj+tbit )/ Tj] Cj

However, this model is developed by comparing CAN frames as tasks. CAN has an arbitration mechanism that is based on physical voltage (LOGICAL AND OF bus voltage) and CAN messages are distributed among the CAN nodes which ae physically distributed, the dynamics of a CAN node and its jitter que length, etc., is vital information for scheduling the task, this information is confined within the node, so appropriate modifications may be needed to model CAN bus as a communication model.

Tindell et al, analysed and given useful model for CAN and methods to evaluate message response time in CAN communication. [6].

M.A. LIVANI·, J. KAISER· and W.J. JIA et al, demonstrated dynamic scheduling of CAN with

Scheduling, with constraints into hard real-time and soft real-time in nature using EDF scheduling. [7].

**Message Schedulers:**

Some designers implemented static schedulers, motivated by the concept from real time operating system(RTOS) task scheduling, but schedulable fixed message priorities from the design stage may not optimize system performance throughout its life cycle when the contributors of latency, like per node process delay, error frames, etc., change the communication dynamics.

**Fixed priority scheduler:**

In the realm of Controller Area Network (CAN) communication, similar to the principle of fixed-priority scheduling in CPUs, the Rate Monotonic (RM) and Deadline Monotonic (DM) algorithms serve as analogies. RM assigns priorities to CAN messages based on their transmission data length, analogous to shorter task periods in CPU scheduling. DM, similar to CPU scheduling, prioritizes messages based on their proximity to deadlines, aligning with the critical time constraints of CAN frames reaching their receiving nodes. These algorithms ensure efficient and deterministic real-time data exchange within the CAN network.

The scheduling algorithm, influenced by RM, assigns priorities to messages based on attributes like transmission time and deadline, ensuring messages adhere to real-time constraints.

**Rate Monotonic (RM) Scheduling:**

In CPU scheduling, tasks contend for CPU time with varying execution times and priorities. The scheduling algorithm, like Rate Monotonic (RM), allocates priorities to tasks based on attributes such as execution time and deadline, ensuring tasks meet their real-time requirements.

Chen, Zai Ping, and Yan Lei Guo Et al, implemented and simulated rate monotonic (RM) scheduling technic to CAN messages [4], this paper details the procedure to guarantee the deadlines of real-time messages in advance, RM is better compared to offline manual priority allocation, mixed priority messages can be transmitted when the message density is less but mainly periodic data is considered in this study, also the priorities are fixed and not dynamic in nature.

However, challenges arise when the size of data frames (messages) varies, akin to differing execution times or periods in CPU scheduling. Like RM's constraints with diverse task attributes, the efficiency of the CAN network can diminish when messages of varying sizes compete for bus bandwidth. Just as the upper bound on CPU utilization is calculated as…

U =Σ1 to n (Ci/Ti)

And condition for schedulability given by U < n(21/n -1)=L(n), where U is the utilization factor, L(n) is the least upper bound of the n tasks (messages) total CPU utilization. [4]

L(n)= n(21/n -1) – ref [4]

The upper bound on CAN bus utilization can be analogous, accounting for the number of messages and their transmission times. This correlation highlights the importance of adapting scheduling strategies to manage contention and differing resource needs in both domains.

With changing network behavior and message density, the variation in relative deadlines is not predictable in advance, and a simple fixed priority scheduler like RM [4] is not the solution.

Also, under “message burst” conditions the upper bound of the CAN message schedulability, which can be tentatively approximated to L(n), drastically reduces with an increase in message count ‘n’, thus the schedulability condition U =Σ1 to n (Ci/Ti), gets violated, so Rate monotonic schedulers is not the best solution for scheduling of CAN messages.

**Dynamic Scheduling:**

Changing the priority of the CAN message identifier “dynamically”, while the network behaviour and message flow behaviour in each node varies, is a good scheme to dynamically mitigate the starvation of some messages and to utilise the bus to its near full potential, Earliest Deadline First (EDF) is a scheduling algorithm used in operating systems to dynamically manage the priority and the schedule of the CPU tasks, a similar approach can be used to schedule CAN messages too, Marco Di Natale Et al [5], studied "Scheduling the CAN bus with earliest deadline techniques”, their EDF research work on CAN scheduling details an applicable method and its implementation, the EDF scheduler increases the processing overhead of the node to an additional 5% whereas improving 20% network utilisation and comparatively better than RM scheduling.

A point to be noted is that scheduling concepts of operating systems tasks cannot be mapped to one is-to-one, to CAN message scheduling.

In operating systems/RTOS, typically, scheduling algorithms like EDF will allow pre-emption of executing tasks by a high-priority task, and later when the high priority task completes its execution, the pre-empted task loads back its context to the CPU and continues from the point where it has left the processing. However, such a practice (pre-emption) is not analogous to the CAN message scheduling, once a CAN frame is started transmitting after it acquires the bus by arbitration, it should be allowed to complete its entire frame transfer till it reaches EoF - end of frame bits, any pre-emption, is meaningless since the node will retry to retransmit and arbitrate, also affecting the error counters of the native node and other nodes. Thus a simple analogous absorption of scheduling ideas from the operating system is not suitable for CAN message scheduler, the concepts of error confinement, error states and arbitration process, etc., are to be considered while designing CAN scheduler with the basic concepts of real time systems.

Also, considering the fact that the “whole frame” has to be transacted from any CAN node (as explained in the previous paragraph) once a scheduling table allows a node to initiate the transmission, necessitates the use of optimisation techniques, like linear programming/integer programming, as part of this research. Other mathematical subjects like “Queuing theory” may also be needed to optimise the sequence of messages to be placed in the CAN-Tx buffer in order to get best network performance considering diverse execution times, message types, and priorities.

Though both static scheduling [4] and dynamic [5], scheduling and time-triggered CAN versions are explored by researchers, they are not adaptive to the changes in the network condition and changes in the message density and “data burst” situation. A dynamic scheduler changes the priority only based on few conditions in the EDF[5], only the deadline of the message is the critical factor, whereas the other conditions like, node failures, bus failures(segmented bus), and other failure conditions are not considered, also such condition may multiply the problems leading to the worst performance in EDF and TT-CAN.

**Adaptive scheduling of CAN messages:**

Once the overwhelming message IDs (more message count) are competing for the bus, during “data burst” zones, the typical RM,DM,EDF etc., may give the verdict that the messages are un-schedulable due to the fact the mere large count of messages leading to, upper bound of scheduling parameter L(n), however a decision to be made which message to be neglected and which message to differ in transmission to achieve a meaningful availability of messages to the receiver nodes, from senders, as message transaction cannot stop, these decisions can not be made only with factors like utilisation factor, deadline miss or cycle time etc., which are typical approaches from CPU task scheduling algorithms and its derivatives (RM,DM,EDF), to schedule messages considering the/faults in the node, bus, and the exponential arriving message rate needs an adaptive approach, an approach which also gives intuitive priority reallocation of the message IDs, to mitigate the data burst and utilise slack bus time available due to fault condition if any.

"Adaptive is not merely dynamic", holds profound relevance in the context of CAN message scheduling. While dynamic scheduling addresses real-time changes in communication demands, adaptiveness encompasses a broader spectrum of considerations.

In the realm of CAN message scheduling, "adaptive" signifies an approach that doesn't just react to predefined parameters (RM,DM) or immediate changes (EDF), but also anticipates and accommodates varying conditions. Unlike a purely dynamic approach that adjusts timing based on real-time data, an adaptive strategy incorporates an understanding of system behavior (branch predictors in CPU architecture, which predict correct to a certain degree based on historical behavior), traffic patterns, and communication priorities.

An adaptive approach to CAN message scheduling involves:

Contextual Awareness: Rather than solely responding to the current traffic load, an adaptive scheduler considers historical data, communication patterns, and predicted future demands. It recognizes that communication needs might evolve over time.

Priority Management: An adaptive scheduler evaluates the criticality of messages and allocates bandwidth accordingly. It ensures that high-priority messages receive timely transmission even during fluctuating network conditions. Also incorporates some heuristics for the priority progression of all type of messages.

Resource Optimization: Beyond immediate bandwidth allocation, adaptiveness seeks to optimize other resources like bus latency, jitter, and system throughput. It makes intelligent decisions to maintain consistent performance levels.

Utilising fault time: There are certain bus segment or node faults which gives excess slack time to the bus ( the time earlier inherited for those faulty part of the bus), this time can be used efficiently if the algorithm can determine the segment of fault which can be exploited.

Learning and Prediction: An adaptive scheduler can learn from past behavior and predict future requirements. This enables proactive adjustments to scheduling parameters, including priority reassignment, leading to improved efficiency and responsiveness.

Balanced Trade-offs: Adaptiveness acknowledges that sometimes strict adherence to real-time dynamics can lead to inefficiencies. It aims to strike a balance between dynamic response and optimizing overall system performance. So we have to handle “baffling idiot” messages which transmit duplicated data and high priority message ids, creating starvation of other messages, also at certain conditions when the schedulability upper bound (L) is very high in the “data burst” time zone, some messages need either to be unserved ( no transmission) or differed.

In summary, while dynamic scheduling is a crucial aspect of CAN communication, adopting an adaptive approach transcends immediate adjustments. It embraces the dynamic nature of the network while incorporating intelligent decision-making that ensures continuous operation and optimal performance with correct resource utilization. An ***adaptive scheduler is forward-looking, harnessing both real-time feedback and historical context*** to create a resilient and efficient communication environment, which is not attempted yet by the researcher and an opportunity for us to explore in this area.

**Problem Statement:**

THE AIM IS TO ***MAXIMISE THE CAN BUS UTILISATION FACTOR***, MAKING IT NEARER TO UNITY, WITH THE FOLLOWING CONSTRAINTS.

1. The CAN frame SHOULD BE ALLOWED TO TRANSMIT WITHOUT PREEMPTION BY ANY TYPE OF MESSAGE *ONCE IT ACQUIRES THE BUS AFTER ARBITRATION AND STARTED TRANSMITTING*. (EVEN AN HIGH PRIORITY MESSAGE CAN NOT PRE-EMPT A TRANSMITTING FRAME OF ANY TYPE, this is in contrast with CPU task scheduling.) – whole message frame to be sent, thus research may involve linear programming for optimisation.
2. Most of the messages are to be transacted within their deadline.
3. All the nodes in the network, to get a fair chance to transmit their messages.
4. All the messages in a native node to get a fair chance to transact to the CAN bus without missing its deadline (amongst its competing messages in the same node with different priorities and types i.e. low and high priority, periodic, aperiodic).
5. The messages may be of “real time” or non real time, and in real time they can be periodic or aperiodic.
6. The nodes are distributed in a bus-based network, therefore the messages, too will have this attribute inherited, such that information of priorities among the same nodes is known to the “heuristic”, present in the native node, the information of other nodes is not readily available without a broadcasting message.
7. Lower priority messages shall get a fair chance to transmit (starvation freedom).
8. “Baffling idiot”, retransmission from a higher priority high-frequency message to be suppressed/addressed, without compromising its functional data “high priority information exchange” (avoid priority inversion at critical stage).

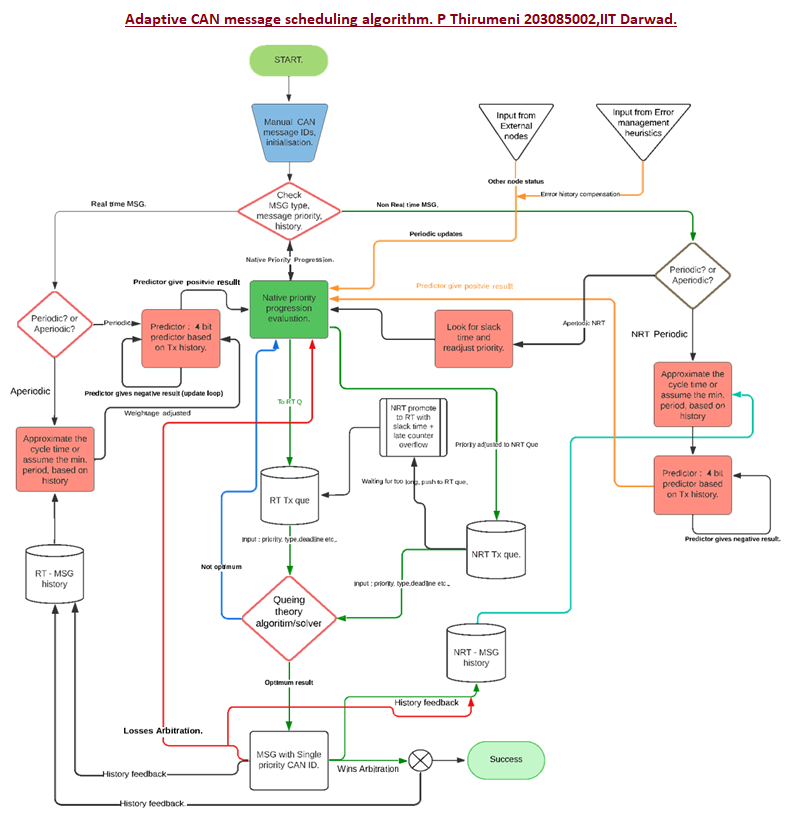
THUS WE NEED AN ADAPTIVE CAN MESSAGE SCHEDULING ALGORITHM, WITH A DISTRIBUTED SCHEDULING TABLE DESIGNED USING OPTIMISATION TECHNICS.

The proposed “Adaptive CAN Message Arbitration and Scheduling Algorithm for Distributed Embedded Systems.” Will take into account the following factors.

1. Will address the Starvation of low-priority messages by high-priority messages in the native node and, also from a non native node.
2. A Native priority progression (NPP) scheme intending to give a better opportunity to all message types, (periodic, aperiodic) and message priorities (both low and high priority will get a fair chance to utilise the bus.
3. Will address failure modes of bus and nodes and, accordingly, tune the arbitration IDs to utilise the bus idle time (slack time created by the faulty nodes or segmented bus).
4. A prediction algorithm to predict nature of the fault and adapt the network to the fault condition (fault tolerant operation).
5. Better handling of “Message burst” time zones. (message burst time zones are time periods, where multiple nodes become hyper active due to an increase in the events and tries to send multiple messages leading to starvation of some low priority messages, such a condition always happens in nodes having similar work patterns, example: Consider a solar array controlled by multiple MPPT systems each MMPT deployed for one solar panel set, which transmits the CAN messages to a central controller, BMS etc, under normal condition, each MPPT node will be less active with few periodic messages transmitting, the typical message/ information about voltage current etc.,( very few CAN message transaction in the bus) but once a large change in the irradiance of sunlight, the nodes may be in a situation to send multiple messages related to sudden “main event” ( higher sun light) like irradiance level, each of the MPPT node will send command to the battery BMS, BMS will reply its own status, and any threshold breach overload trip etc., thus suddenly the message density in CAN bus will increase drastically, and since all the MPPTs will undergo this phenomenon as the sunlight affects all the solar panel and its connected MPPT nodes, all nodes will go hyper active, thus leading to a “message burst” condition*. IN SUCH A SITUATION, ONLY AN ADAPTIVE CAN MESSAGE SCHEDULING ALGORITHM THAT CONSIDERS THE ABOVE POINTS AND ADJUSTS THE MESSAGE PRIORITY WILL BE EFFECTIVE IN MAINTAINING BUS UTILISATION.*
6. The adaptive algorithm will effectively handle priority inversion of high-priority messages even with priority modifications.
7. The design will incorporate per-node heuristics to adaptively update its message IDs, and also will have diagnostic and command messages from a master node which will update a part of the arbitration field to reflect the starvation state of the message and give a fair chance to all the nodes.
8. The CAN 2.0B frame 29 bit arbitration bits will be segmented and used for updating the Native Priority progression (NPP) scheme, in few bits, and a few other bits will be used to reflect fairness to the message IDs of the exterior node. ( the status of the other node messages will be periodically broadcasted so that all nodes will readjust to the fair need of other nodes)
9. A part of the arbitration field, particularly the LSB part, will be used to allocate manual IDs for last-bit arbitration by the CAN arbitration mechanism, which gives some manual control that may be needed to tune the network and to start the scheduling process with initial values.

10) Queuing theory and integer programming technics may be used to arrive at the best priority update for the message IDs.

11) A 4-bit Tx success predictor, to predict the success rate of a message getting transmitted based on its earlier transmission history, priority or frequency of message transmission is adjusted based on the historical transaction of the message ID, behavior of the network, or the current load. This adaptive approach can be seen as a form of prediction (similar to prediction in computer architecture of conditional branches to optimize instruction throughput using 2 bit predictor), here the system learns from the history and tries to anticipate the communication needs and adjust the message scheduling accordingly, in case of erroneous prediction the predictor will reset and readjust, this approach will be particularly effective for periodic messages and message sets which has a pattern/sequence.



Please see the pdf file for a clearer version of the above flow chart.

Segmenting the Message ID field. (this is tentative idea/motivation)

The arbitration field consisting of 29 bits in CAN 2.0B shall be segmented to carry out independent priority enhancement activities by the following streams of heuristics.

1. A scheme to take account of the improvement of priority of CAN Ids in the native CAN node, this can be called Native priority progression.
2. The first 6 MSB bits (bit 24 to 29) shall be reserved for diagnostic and command messages between nodes and a possible master node. Totally 64 such diagnostics/command messages (2^6=64) is possible.
3. The next 5 bits (20 to 24 bits) be reserved the Native priority progression (NPP) algorithm.
4. The next MSB bits of 4 bit (bits 17-20) be reserved for the history based 4-bit success predictor.
5. At this stage, it is not clear (once detailed research is done the based on effectiveness, the advantage may be realised) whether the predictor will give input to the NPP or the predictor itself will be incrementing or decrementing the ID values, if the predictor just gives its verdict as an input, to the NPP then these 4 bits are not needed.
6. The external node message sch. fairness algorithm will be handling the next 4 bits (bits 14 to 17), these will not be updated very fast since the input is from periodic broadcasting messages.
7. Another 4 bits from bit (11 to 14) are reserved for error condition heuristics. These bits will be incremented and decremented by the error conditions.
8. The priority handled by the NPP is the highest since it handles the bits at the MSB end, those are the bits which is arbitrated first in the CAN arbitration process. The error condition algorithm and the external nodes message fairness algorithm are given bits at the LSB end, since the active nature of these algorithms is slow since they depend upon data from the broadcast message, and for error diminishing bits, the occurrence of errors may be rare.
9. The remaining 10 bits (bit 0 to 10) are used for manual priority fixes, these are at the lowest LSB end of the ID field and has minimum priority influence.

The above segmentation will vary and may need to be shuffled up based on actual research results and experimental data based improvements.

**Motivation for the heuristics:**

**Heuristics for Native Priority Progression (NPP) algorithm.**

The NPP will have several heuristics, in which “Transmission Denial counter is one such mechanism”.

Each Node will maintain a transaction denial counter, this counter is incremented from the feedback of Rx MSG history or Tx MSG history database (see the algorithm flow chart), when a message fails to transmit during its arbitration or when its period is consumed/overflowed in the Tx que itself, it is considered as “the message is denied from its natural right to transmit within its period, (for aperiodic messages the adjusted minimum time is considered as period), then this “denial counter” is incremented using the threshold limit agreed and set, if the value breaches the threshold set by the network design, then the node is allowed to mark an decrement (which will increase the priority) in a special set of priority promotion bits(bit 24 to 29 - see point B, in the previous page), in its message ID field, by flipping the bit from 1 to 0.

The threshold value for the above counter can be set differently if a node has mixed basket of real time and non real time messages, then if the ratio of the type and total count of the message originating from this node will be used to decide the threshold.

For example, if the Node has 20 message IDs associated and all of them are critical then the threshold can be set by n/20 where n is the total message count in the network, if n=180, then the threshold will be 180/20=9, so for every 9 misses or denial of transmission attempt the denial counter will be decremented by 1, after 4 such decrements 4 x 9 = 36 denials the node will be allowed to modify its message id temporarily to the lowest ID value ( highest priority) to send make its message out, once this message is passed successfully without error the node will normalise this priority modification to its earlier state.

**Heuristics for Erroneous Node diminishing priority:**

A node may experience errors due to CAN-related issues (Tx/Rx error counter) or due to the native host undergoing some errors or due to the source sensor or actuator having error/ fault issues, in such situations, the CAN protocol takes care of the error handling related to the Tx/Rx error counter (CAN protocol related error). However other faults errors in the host or node is not automatically addressed until the host node is designed to transmit a message explicitly to other nodes, and such message transmission based error news is both consumes bus time and also needs to be implemented in the application design, which is not mandatory.

We can address this by using an erroneous node diminishing priority scheme, In this proposed technic, once any of the fault or unaddressed error in a node is confirmed, this node make the default dominant bit to recessive ( making all the message ID of this node less priority one) suing the reserved bit for error condition manipulation (bit 11 to 14), with a 4 bit direct operation 15 error conditions can be reflected in the message ID priority, if more error is envisaged then a software counter may be implemented which will increment or decrement this 4 bits on overflow. One drawback is the entire message set in that node to be changed In case of a common error, and a specific set of message IDs can be changed for errors associated to those messages.

These are all, initial intuition and motivation, a better mechanism to calculate the threshold value to be researched and arrived at after network hardware in loop simulations with real time non real time and periodic aperiodic tasks and learning from the experimental data.

**Possible simulation setup:**

Software to be used :

1. CAN bus analyser, to visualise, monitor CAN message frames with IDs, timing parameters and, bus utilisation values, etc.,
2. MATLAB for network simulation (Vehicle Network Toolbox) or for queuing theory calculation, integer programming etc.,
3. Wireshark for network monitoring.
4. Logic analyser (sigrok pulse view) to visualize, physical layer and datalink layer dynamics.
5. MikroC IDE, Arduino IDE, STMcube ide, etc, for programming the target boards.
6. SOCKET CAN – Linux.
7. CANoE or CANalyser (commercial software not available, if provided, is add on and will use effectively)
8. Any other free CAN network simulation software, (not known to me yet.)

Hardware:

1. Wave share dual port CAN analyser, for Bus sniffing and frame visualizing.
2. Hardware in Loop (HIL) setup using Multiple CAN nodes (microcontroller daisy-chained using SPI communication), from PC, commands will be given to these CAN nodes to “simulate and behave” as per the experimental recipe ( the recipe is CAN message ID and error simulation, rate of message simulation multiple count messages creating “message burst”, periodic aperiodic group of messages busting at specific time range contending for the bus, many such experimental conditions can be simulated with Hardware in loop and CAN bus monitor.
3. Other CAN transceivers, power supplies etc.,
4. Oscilloscope to study minute logic signals missed by logic analyser.
5. Data logger to log events and errors, if needed.

**Skills available and useful for this research:**

1. Good CAN working knowledge.
2. Embedded C.
3. Microcontroller-based system and hardware development.
4. Good understanding of real-time concepts and scheduling concepts.
5. Earlier research on CAN security aspects.
6. Ample practical experience in electrical and electronics, AFDX protocol implementation.

**Skills to be acquired:**

1. Integer programming/linear programming, for scheduler optimisation.
2. Network simulation without hardware using virtual CAN nodes.

**P Thirumeni**

Ph.D. Student – 203085002

**References.**

[1] Comparison of In-Vehicle Communication Protocols for Critical Applications. White paper on TTCAN by Dr.-Ing. A. Albert Robert et al, Bosch GmbH, Dept. FV/SLF

[2] Time triggered CAN – TTCAN, Dr. Bernd Müller et al, Robert Bosch GmbH.

[3] Comparison of In-Vehicle Communication Protocols for Critical Applications

Edward Robert Gundlach and Syed Masud Mahmud, Electrical and Computer Engineering Department, Wayne State University, Detroit.

[4] Chen, Zai Ping, and Yan Lei Guo. “Scheduling the CAN Bus with Rate Monotonic Techniques.” Advanced Materials Research, vol. 383–390, Trans Tech Publications, Ltd., Nov. 2011, pp. 4318–4322

[5] M. Di Natale, "Scheduling the CAN bus with earliest deadline techniques," Proceedings 21st IEEE Real-Time Systems Symposium, Orlando, FL, USA, 2000, pp. 259-268,

doi: 10.1109/REAL.2000.896015.

[6] Analysing Real-Time Communications: Controller Area Network (CAN), K. W. Tindell, H. Hansson A. J. Wellings Department of Computer systems, University of Uppsala, Sweden, Department of Computer Science, University of York, England.

[7] SCHEDULING HARD AND SOFT REAL-TIME COMMUNICATION IN THE CONTROLLER AREA NETWORK (CAN),M.A. LIVANI·, J. KAISER· and W.J. JIA·University of Uim, Department of Computer Structures, 89069 U/m, Germany.