International Journal of Reconfigurable and Embedded Systems
Vol.8, No.1, March 2019, pp. 36~50
ISSN: 2089-4864, DOI: 10.11591/ijres.v8.i1.pp36-50

Performance evaluation of embedded ethernet and Controller Area Network (CAN) in real time control communication system

Ching Chia Leong, Mohamad Khairi Ishak
School of Electrical and Electronic Engineering, Engineering Campus, UniversitiSains Malaysia, Malaysia

Article Info

Article history:

Received Dec 5, 2018
Revised Jan 27, 2019
Accepted Feb 9, 2019

Keywords:

Communication
Embedded system
Ethernet
Real time

ABSTRACT

Real-time communication is important in control network. In real-time communication, message need to be delivered from source to destination within specification. Embedded Ethernet and Controller Area Network (CAN) protocol can be used in control network to achieve hard real-time communication. For embedded Ethernet protocol, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is the media access control (MAC) used to control data transmission between nodes in network. Back-off algorithm in CSMA/CD is used to handle packet collisions and retransmission. For CAN protocol, it is communication protocol developed mainly for automotive application. It has priority arbitration to handle collisions and retransmission. In this project, embedded Ethernet network models and CAN network models are developed and simulated in MATLAB Simulink software. Several back-off algorithms, which are Binary Exponential Backoff (BEB), Linear Back-off Algorithm, Exponential-Linear back-off Algorithm and Logarithm Back-off Algorithm are proposed and implemented into Embedded Ethernet network model to evaluate the performance. Both embedded Ethernet and CAN network models are extended to 3 nodes, 10 nodes, and 15 nodes to evaluate performance at different network condition. The performance criteria evaluated and discussed are average delay and jitter of packets. The results show that in network with high number of nodes, Linear Back-off Algorithm and Exponential-Linear back-off Algorithm shows improvement in packets delay and jitter. For CAN network, the packet jitter is relatively low.

Corresponding Author:

Mohamad Khairi Ishak,
School of Electrical and Electronic Engineering,
UniversitiSains Malaysia,
Engineering Campus, NibongTebal, 14300, Penang, Malaysia.
Email: khairiishak@usm.my

1. INTRODUCTION

Real-time control communication is critical in many industries and applications due to expansion of system physical setup and functionality. For instance, medical device, automotive system, and robotic. Real-time control communication provides easy implementation by reducing complexity of wiring and provides flexibility and maintainability of system. Real-time control communication helps to transmit message within specific bounded time delay. The application of real-time control communication mostly found in network system where safety is involved. For instance, Anti-lock Braking System (ABS) is one of automobile safety features that have high requirement in fast response time [1-3].

The characteristics of real-time control communication are small but frequency packet transmission among large number of nodes and meet the time critical requirement [4]. Several communication protocols

have been proposed to meet the requirement of real-time control communication system. For instance, Ether CAT, Control Net, TT Ethernet and CAN open. EtherCAT uses master and slave architecture based on original Ethernet and exclusive hardware is needed. ControlNet uses token-passing approach in control message transmission. The node that hold the token can transmit message until time out. TTEthernet adds hardware in Ethernet to implement time triggered and event triggered traffic in the network. Custom switches are needed for node synchronization. CAN open uses a synchronization signal to trigger node transmission simultaneously [5-7].

Ethernet IEEE Std 802.3 published in 1985, is a half-duplex protocol that use Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as the media access control (MAC) to control data transmission between nodes in network. Back-off algorithm in CSMA/CD is used to handle packet collisions and retransmission. Ethernet has advantage of being the leading network technology for long times. Thus, it is practical in implementation, expendability, maintainability and configurable. It is also low cost due to commonly available and simple hardware [6]. Since Ethernet has many advantages on real-time system, this thesis will investigate on Ethernet and CSMA/CD. Back-off algorithm also studied deeply to propose different modified algorithms to enhance the performance especially in real-time control communication system.

Controller Area Network (CAN), protocol invented by Bosch in 1986, is deterministic communication protocol developed mainly for automotive application. It has priority arbitration to handle collisions and retransmission [2]. Since CAN protocol is designed for control system such as car ABS system, it is deterministic due to priority and has bounded time delay. CAN protocol only need simple low-cost hardware to implement, that is twisted pair wires and resistors. Therefore, this thesis also spend effort to investigate the performance of CAN being use as real-time communication control system.

2. MODEL SIMULATION

The software used in this project is MATLAB. MATLAB is numerical computing software, developed by MathWorks and has lots of toolbox and library ready to use. The version used is MATLAB R2015a. Simulink is a graphical programming environment that has well integration with MATLAB. It is well known for multidomain system simulation and model design in field of automation control and digital signal processing. The graphical block programming tool and block library in Simulink are used intensively to develop the simulation model. SimEvents is also necessary in this project. SimEvents is a discrete event simulation tool that add event-based simulation engine and building blocks to Simulink. MATLAB, Simulink and SimEvents form a powerful environment to create a network simulation model needed in this project.

The research starts with Ethernet model development. The initial Ethernet model consist default back-off algorithm, that is BEB algorithm and it has only three nodes. Then evaluation and validation take place to ensure the model is working as intended. The next step is run simulation and collect data. To compare performance of different back-off algorithm, the back-off algorithm in the model is replaced to linear back-off, exponential linear back-off, and logarithm back-off. To simulate the performance when higher number of nodes are present, ten nodes and fifteen nodes models are developed, evaluated, validated and simulated. Finally, whole data collected is analyzed and report and is generated. Besides Ethernet model, CAN model is also developed using similar flow.

2.1. Embedded Ethernet

Three nodes Ethernet model is shown in Figure 1. There are three important blocks in the model, which are Application, MAC Controller, and T-Junction. Application Block is used to model the generation and consumption of Ethernet data packet. MAC Controller block is used to control the node's use of shared channel. T-Junction block is used to connect node to the network There are other blocks for linking and termination purpose, such as Cable and Terminator. Figure 1 is only the basic Ethernet model with 3 nodes. The Ethernet model is developed in a way that it can be extended. In this project, it is extended to 10 nodes, and 15 nodes. The aim is to evaluate and measure the performance when number of nodes increases.

The main function of Application Block is to model the generation and consumption of Ethernet data packet. Figure 2 shows the blocks that handle packet generation. Event-Based Random Number2 gets packet rate and random seed as input parameter, then generates intergeneration time randomly. The intergeneration time will be used by Time-Based Entity generator to generate entity.

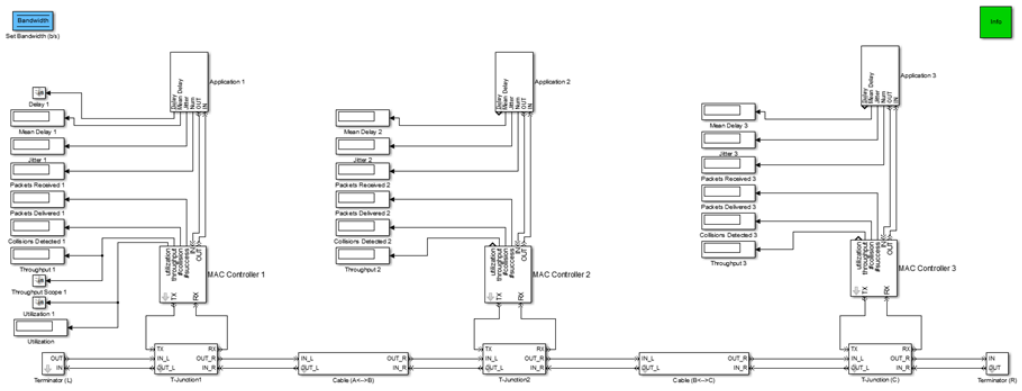

Figure 1. Three nodes embedded ethernet model

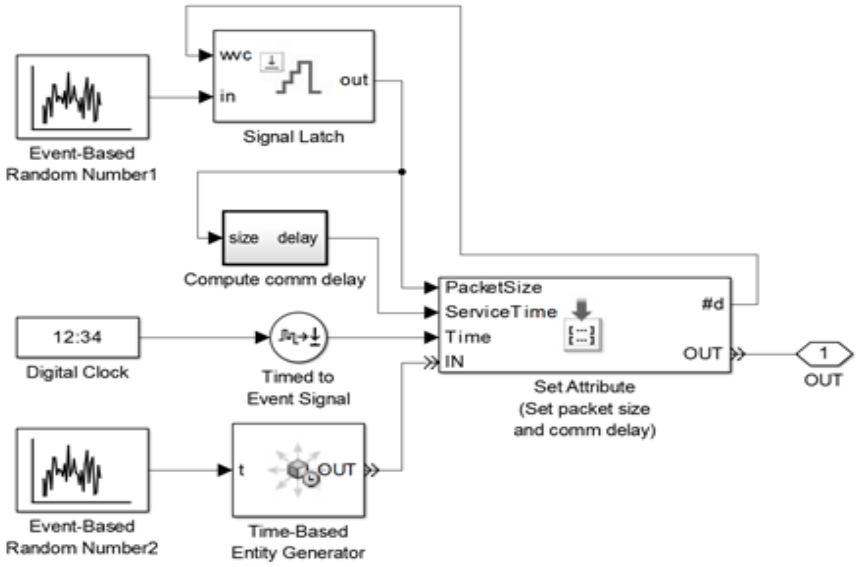

Figure 2. Packet generation blocks

Set Attribute block sets attributes to the entity. Attributes is numerical data attached to the entity. The entity here is the packet and the attributes of packet are PacketSize, ServiceTime, and Time. For Packetsize, it is the size of packet in bit. Event-Based Random Number1 is used to generate random number of bit from uniform distribution and Signal Latch block is used to write to and read from memory. In this project, the packet size is fixed at minimum, that is 64 bit for consistency purpose. The Service Time is the time needed to service the packet. The Compute Comm Delay block is used to calculate the service time using the formula in(1)

$$\text{ServiceTime} = \frac{\text{Packet Size}}{\text{Bandwidth}} \quad (1)$$

Digital clock block is used to output the time when source node begins transmitting packet, then converted to event signal and set as attribute Time in packet. Its purpose is to calculate the delay of the packet, when packet reaches the destination node. After attributes of packet are set, the packet is sent to output port, which will then send to MAC Controller block.Figure 3 shows the blocks that handle packet consumption. These blocks main function is to retrieve the attributes of the packet for delay and jitter calculation. #d is the number of entities departed, represent the number of packet received. Time is the time when source node begins transmitting packet.

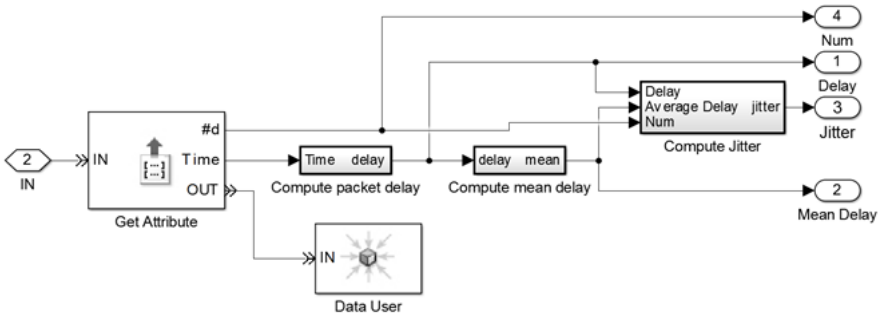

Figure 3. Packet consumption blocks

Compute Packet Delay block in the Figure 3 is used to compute delay. In this project, delay is defined as time difference between the time when the source node begins transmitting packet and the time when destination node complete receiving packet. Equation shows how to compute delay of a packet,

$$T_{delay} = T_{dest} - T_{src} \quad (2)$$

where $T_{src}$is the time when source node begins transmitting packet and T_{dest} is the time when destination node complete receiving packet. Digital clock block is used here to output T_{dest}. Get Attribute block is used to retrieve the T_{src}.

The Compute Mean Delay block is used to compute average delay. Get Attribute block is used to retrieve the number of packet received. Equation below shows the formula to calculate average delay.

$$T_{avg\ delay} = \frac{1}{N}\sum_{i=0}^{N} T_{delay}^{i} \quad (3)$$

where N is the total number of nodes and $T_{avg\ delay}$ is the delay of individual packet. The Compute Jitter block is used to compute jitter of packets. Jitter is the standard deviation of delay. It shows whether delay widely spread out or closely distributed around the average delay. The code behind Compute Jitter block is shown in Figure 4.

```
function jitter = compute(delay, average, num)

persistent sum;

% Initialize persistent variables in the first iteration.
if isempty(sum)
sum = 0;
end

dif = delay - average;
dif_power = dif.^2;
sum = sum + dif_power;
standard = sum /(num-1);
standard_sqrt = sqrt(standard);

% Update the jitter of current node.
jitter = standard_sqrt;
```

Figure 4. Code of compute Jitter block

The Compute Jitter block has three inputs, that are delay, average delay and number of packets. The code is developed according to standard deviation formula in Equation.

$$T_{Jitter} = \sqrt{\frac{\sum_{i=0}^{N}(T_{delay}^{i} - T_{avg\ delay})^2}{N-1}} \quad (4)$$

where N is the total number of nodes, T_{delay}^{i} is the delay of individual packet, and $T_{avg\ delay}$ is the average delay of packets.

Figure 5 shows the structure of T-Junction block. The function of T-Junction block is to connect node to the network. There are few Path Combiner and Replicate blocks. The packet is duplicated by Duplicate block. One packet is directed upward to MAC Controller block of the node. Another packet is directed to Cable block then to T-Junction block of other nodes. The Path Combiner and Replicate blocks are connected in a way that packet from left side can transmit to right side and vice-versa.

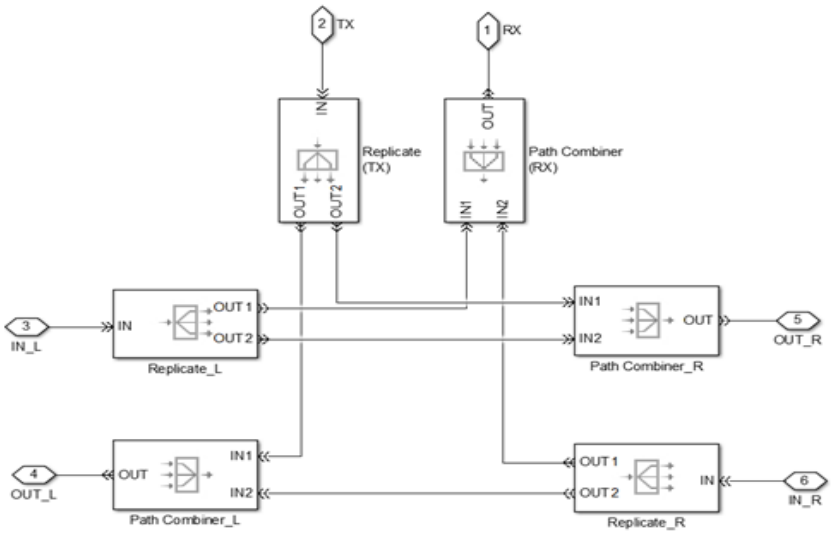

Figure 5. Structure of T-Junction block

MAC Controller block is used to control the node's use of shared channel by imitate and simulate the CSMA/CD protocol and back-off algorithm. Next, explanation will be made on how the blocks of MAC Controller structure in Figure 6 imitate and simulate characteristics and behaviours of CSMA/CD protocol. Figure 7 shows the flow chart of CSMA/CD protocol. The flow chart consists of different process and decision making. For listen to network process and network idle decision making, it is imitated by CSMA/CD block and Media Access Control block. For transmitting packet process, it is imitated by Transmission Buffer, Control and Controller block. For collision detection decision making, it is imitated by Infinite Server block and CSMA/CD block. For Back-off algorithm process, it is imitated Single Server block and Backoff Controller Block. These are the key blocks that imitate the CSMA/CD protocol in Ethernet. Next, the function of each block used in MAC Controller will be explained in order.

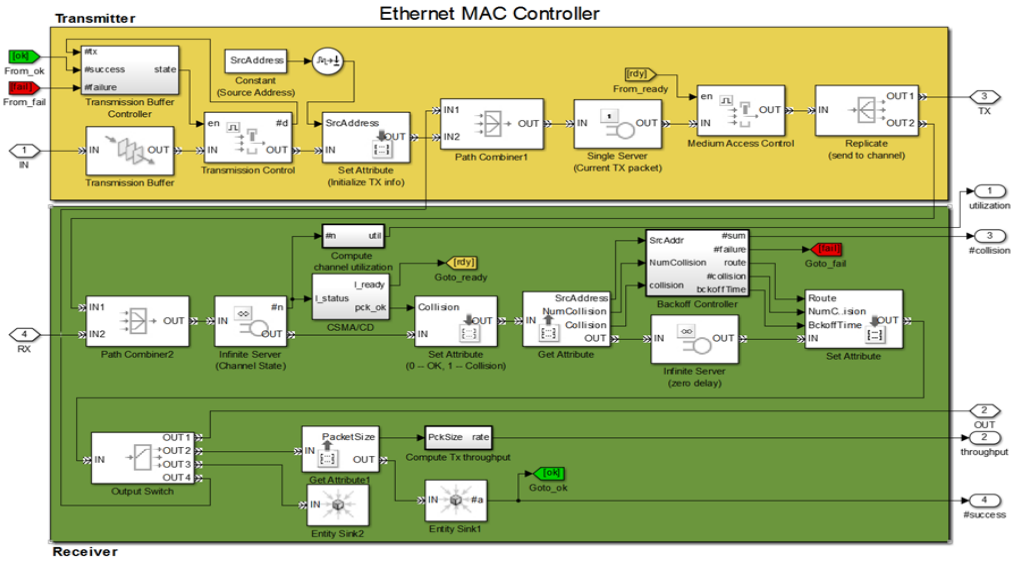

Figure 6. Structure of MAC controller block

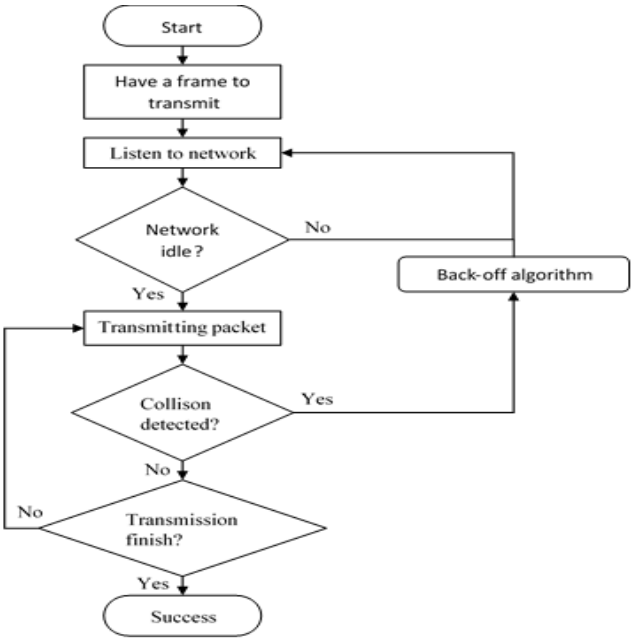

Figure 7. Flow chart of CSMA/CD protocol

The MAC Controller is divided to two regions, transmitter and receiver. For transmitter region, the function is to transmit the packet if the channel is idle. There are many blocks work together to achieve this. For starter, Transmission Buffer block stores packets from Application block in first-in-first-out sequence. The Transmission Control block is used to allow or forbit the arrival of packets from Transmission Buffer block, based on enable signal. The enable signal is computed from Transmission Buffer Controller block.

Figure 8 shows the structure of Transmission Buffer Controller block. The output state is the sum of constant 1, number of departed packets from Transmission Control block, number of packets failed to transmit, and number of packets success to transmit. In other words, Transmission Control block will only accept the packet if the output state will be 1, which mean all departed packets is completely processed, the packets either fail or success to transmit. These three blocks control the flow of packets, making sure the packets are processed one after one.

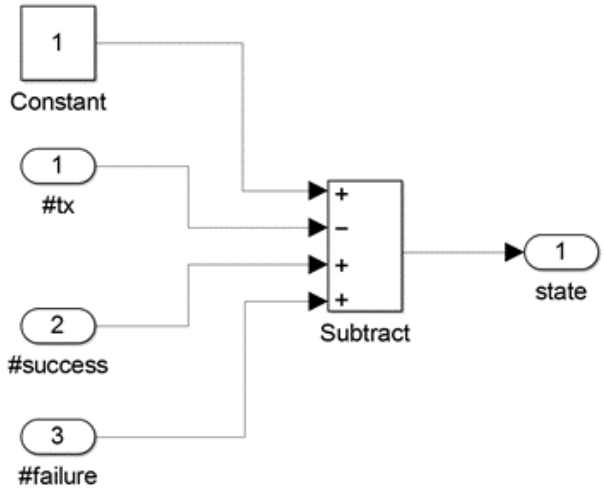

Figure 8. Structure of transmission buffer controller block

Next, Set Priority block set three attributes to the packets, that are SrcAddress, BckoffTime and NumCollision. Each node has own SrcAddress preset manually. BackoffTime and NumCollision is set to 0 in this initial stage. Path Combiner1 is used to merge path, that is accepting packet from two input port. One input accepts packet from transmission region, and one input accepts packet from receiver region, in case of packet retransmission when collision occur. Single server block put the packet on hold for a duration of time, based on value of attribute BckoffTime. For first transmission, packet from transmission region, BckoffTime of the packet will be 0. For retransmission, packet from receiver region, BckoffTime of the packet will be vary, based on back-off algorithm.

After that, Medium Access Control block allows or forbit the arrival of packet from Single Server block, based on rdy signal. If rdy signal is positive, it accepts the packet, and put the packet on hold if otherwise. The rdy signal is computed from CSMA/CD block. Basically, rdy signal will be positive if shared channel is idle, and negative if shared channel is busy. Replicate block accepts a packet, duplicate it, and output to two output ports. One packet sends to the T-Junction block, which distribute it to shared channel. One packet sends to receiver region, to check for collision, and retransmit if collision happen.

For receiver region, the function is to check for collision and retransmit using back-off algorithm. The Path Combine2 block is used to merge path, that is accepting packet from two input port. One input accepts packet from T-Junction block of the shared channel. One input accepts packet from transmitter region, to check for collision in here, receiver region and send back to transmitter region to retransmit, if collision occur.

After that, Infinite Server block put the packet on hold for a duration of time, based on value of attribute ServiceTime. This represent the time needed to send the bits of packet to the channel. Infinite Server block has output #n, number of packets in the block, which means number of packets transmitting in the channel. If more than one packets are transmitting in the channel, collision occurs.

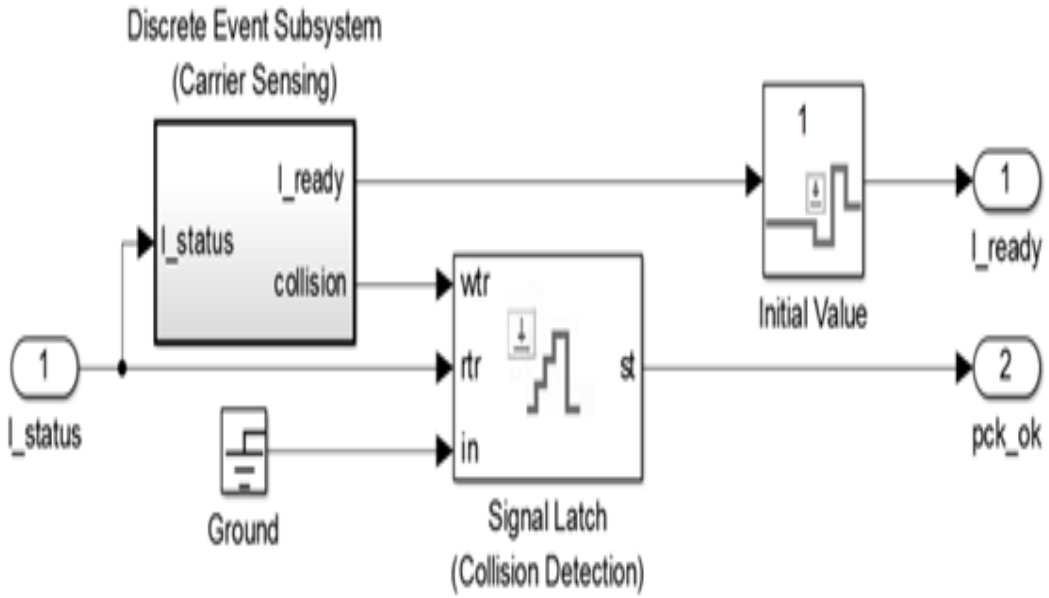

Figure 9. Structure of CSMA/CD block

The #n of Infinite Server block is used by next block, CSMA/CD block to compute l_ready and pck_ok. l_ready tells whether the channel is idle and pck_ok tells whether collision occurs. Figure 9 shows the structure of CSMA/CD block. For Discrete Event Subsystem block, the input is I_Status, the number of packet transmitting in the channel. It has two output, l_ready and collision signal.

Figure 10 shows the structure of Discrete Event Subsystem block. l_ready is equal to 1-l_status. l_ready will be 1 only if no packet is transmitting in the channel, and will be 0 or negative otherwise. Thus, l_ready tells whether the channel is idle. Collision signal is equal to l_status -1. Collision signal will have rising edge from 0 to 1, when l_status, the number of packet transmitting in the channel, change from 1 to 2. Thus, collision signal creates rising edge when collision occur.

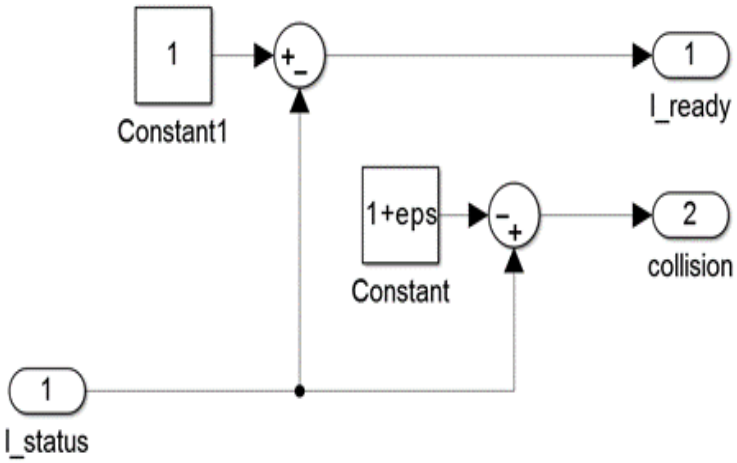

Figure 10. Structure of discrete event Subsystem block

The rising and failing edge characteristic works with the next block, Signal Latch block. The Signal Latch block responds to two events, l_status increases from 0 to 1, and l_status increases from 1 to 2. When l_status increases from 0 to 1, rtr has rising edge, trigger a read event and output st is 0. When l_status increases from 1 to 2, wtr has rising edge, trigger a write event and output st is 1. It is the nature of the block that the state of the block is 0 if the last event was a read event, and 1 if the last event was a write event. Output st is 0 only if one packet transmits in the channel for entire duration. This outcome means the packet transmits successfully and other outcome means collision between two or more packets occurs.

The Set Attribute block is used to set attribute collision to the signal. This attribute represents whether collision occur or not for this packet in this transmission. Then Get Attribute block is used to retrieve attributes of the packets, SrcAddr, NumCollision, and collision for the back-off algorithm processing.

Figure 11 shows the state flow diagram of back-off algorithm. Basically, there are transmitting state and receiving state. First, the flow checks the attribute SrcAddr of packet with SrcAddr of node. If the SrcAddr of packet and node are same, it is understood the packet is transmitting from own node. In this case, the flow transition to transmitting state. Then, the flow checks the attribute collision of packet. If no collision occurs, the flow transition to TxOK state. The route is set to 2. Route is an important attribute of packet and Table 1 shows where the packet is directed by Output Switch block, based on the route of packet.

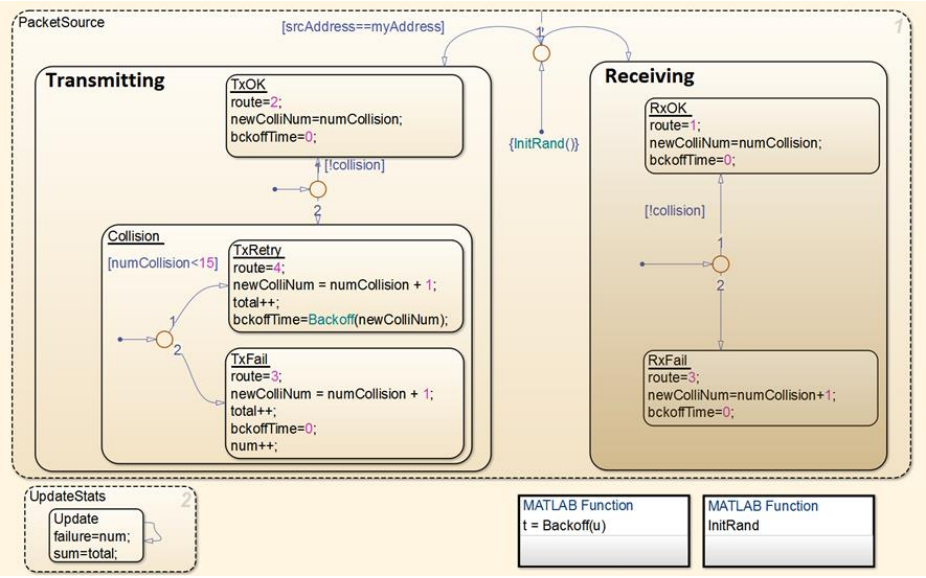

Figure 11. State flow diagram of back-off algorithm

Table 1. Packet Direction of Attribute Route

Route	Direction
1	Received by Application block for processing
2	Increment number of packet transmit successfully and contribute to the throughput
3	Reject and discard the packet using Entity Sink block
4	Send to transmission region for retransmission

For other attributes, NumCollision remain the same and BckoffTime is set to zero. If collision occurs, the flow transition to Collision state. The flow checks for NumCollision. If NumCollision is less than 16, the flow transition to TxOK. The route is set to 4, NumCollision increment by one and BckoffTime is computed using selected back-off algorithm. If NumCollision is more than 16, the flow transition to TxFail and route attribute is set to 3.

If the SrcAddr of packet and node are different, it is understood that the packet is received from other nodes. In this case, the flow transition receiving state. Then, the flow checks the attribute collision of packet. If no collision occurs, the flow transition to RxOK. The route is set to 1, NumCollision remain the same and BckoffTime is set to zero. If collision occurs, the flow transition to RxFail. The route is set to 3, NumCollision increment by one and BckoffTime is set to zero. It is the transmitting node responsibility to compute for the back-off time, receiving node only have to discard the packet.

The function Backoff in structure of MAC Controller block is used to imitate the back-off algorithm in CSMA/CD protocol. The default in Ethernet protocol is Exponential Back-off Algorithm. Based on literature review, other back-off algorithms are developed and implemented. The back-off algorithms developed are Linear Back-off Algorithm, Exponential-Linear Back-off Algorithm and Logarithm Back-off Algorithm.

The default back-off algorithm in Ethernet protocol is Exponential Back-off Algorithm. Exponential Back-off Algorithm increases back-off time exponentially. Figure 12 flow chart of Exponential Back-offAlgorithm in the function. C is collision and ST is slot times. When a collision happens, it checks the collision that had occurred to the packet. If collision is less than 15, collision increment by one. Then it checks if number of collision is more than 10. If collision is less than 10, delay time is computed by choosing random value below slot times, exponential it, and multiply it by minimum slot time 51.2μs. If collision is more than 10, collision value used in delay time computation is always 10. If collision is more than 15, transmission fail and packet is discarded.

Linear Back-off Algorithm increases back-off time linearly. Figure 13 flow chart of Linear Back-off Algorithm in the function. C is collision and ST is slot times. When a collision happens, it checks the collision that had occurred to the packet. If collision is less than 15, collision increment by one. Then delay time is computed by choosing random value below slot times, and multiply it by minimum slot time 51.2μs second. If collision is more than 15, transmission fail and packet is discarded.

Exponential-Linear Back-off Algorithm increases back-off time exponentially or linearly depend on number of collision. Figure 14 shows flow chart of Exponential-Linear Back-off Algorithm in the function. C is collision and ST is slot times. When a collision happens, it checks the collision that had occurred to the packet. If collision is less than 15, collision increment by one. Then it checks if number of collision is more than 10. If collision is less than 10, delay time is computed by choosing random value below slot times, exponential it, and multiply it by minimum slot time 51.2μs second. If collision is more than 10, linear multiplication is used instead of exponential. If collision is more than 15, transmission fail and packet is discarded.

Logarithm Back-off Algorithm increases back-off time using logarithm. Figure 15 flow chart of Logarithm Back-off Algorithm in the function. C is collision and ST is slot times. When a collision happens, it checks the collision that had occurred to the packet. If collision is less than 15, collision increment by one. Then delay time is computed by choosing random value below slot times, logarithm it, and multiply it by minimum slot time 51.2μs second. If collision is more than 15, transmission fail and packet is discarded.

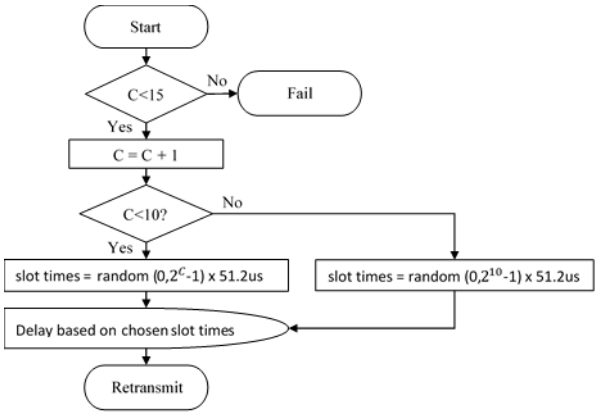

Figure 12. Beb algorithm

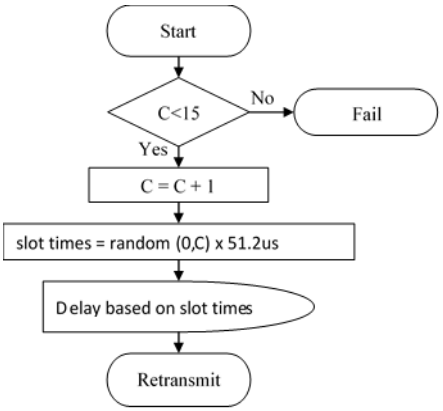

Figure 13. Linear back-off algorithm

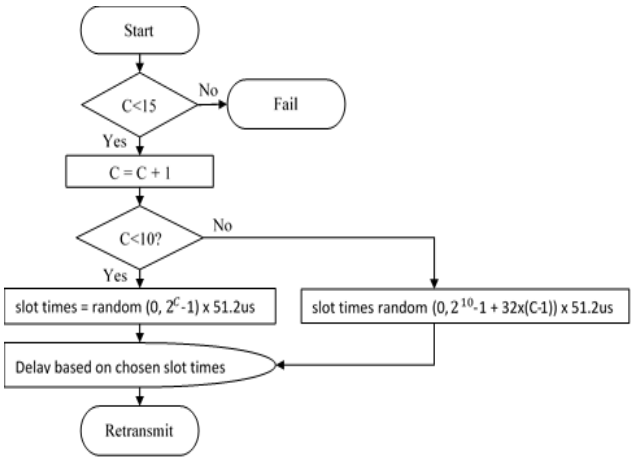

Figure 14. Exponential-linear back-off algorithm

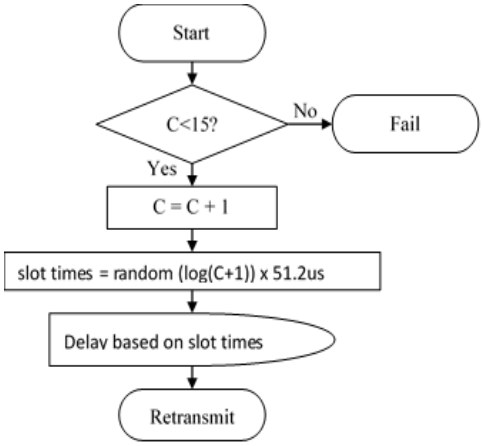

Figure 15. Logarithm back-off algorithm

2.2. Controller Area Network Model

Figure 16 shows the structure of CAN model. There are three important blocks in the model, which are Application, CAN Node, and CAN Bus. The Application block models the generation and consumption of 8 bytes CAN data packet. The CAN Bus block connects the CAN Nodes to the network. The critical block is CAN Node block, which handle the nodes use of the share channel.

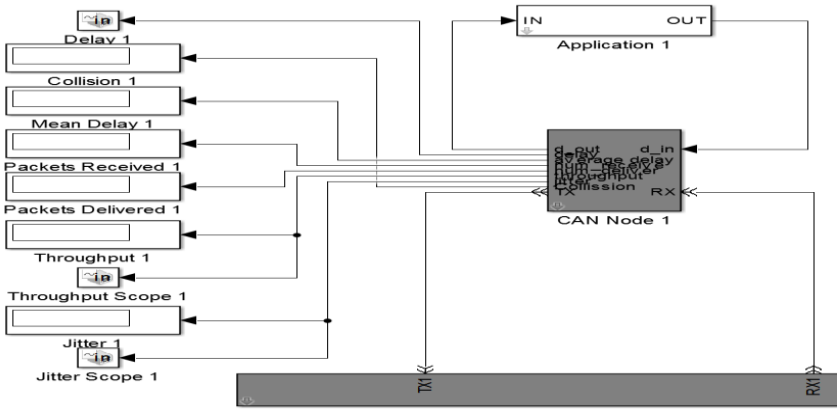

Figure 16. Structure of CAN model

The CAN Node block has several outputs. d_in and d_out are used to transmit pactet to and from Application block. TX and RX are used for transmit packets to and from the CAN Bus block. The rest outputs are used for data collection, such as average delay, jitter, number of collision, and number of packets.

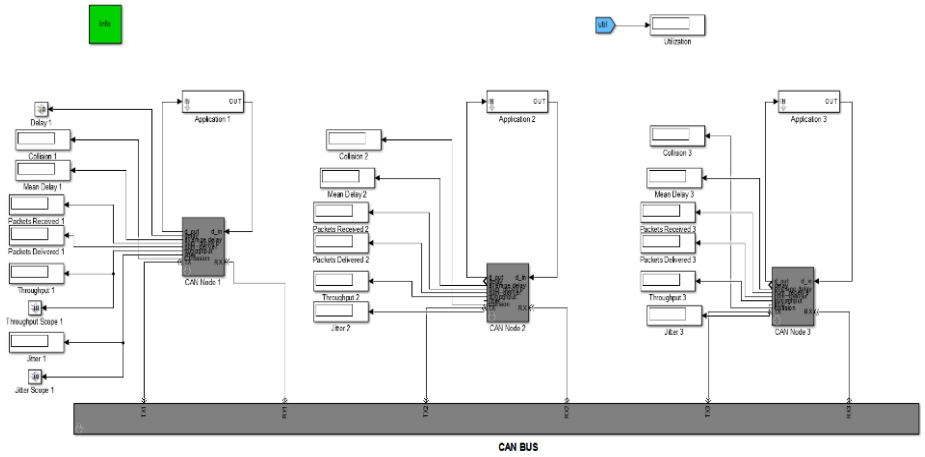

Figure 17. CAN network model with 3 nodes

Figure 17 is CAN network model with 3 nodes. The CAN model is developed in a way that it can be extended. In this project, it is extended to 10 nodes, and 15 nodes. This is to evaluate and measure the performance when number of nodes increases. Each node is assigned unique ID and has different priority. The purpose of CAN Node block is to imitate and simulate the characteristic and behavior of CAN protocol. There are few key blocks in CAN Node block. They are Transmission Buffer block, Re-transmission block, and Error detection block.

Transmission Buffer block stores the packets in sorted sequence. The sorted sequence is based on the ID attribute of packet. ID attribute is the identifier of the packet. It is identity of the node that transmit the packet. However, more importantly it carries priority information of the node. The smaller value of node ID, the higher the priority of the node. In overall, the block sort packets according the value of ID. Thus, the higher priority packet will pass to the output first.

Figure 18 shows the structure of Retransmission Buffer block. It checks for acknowledgement signal. It will receive acknowledgement signal if a packet transmits successfully. If acknowledgment signal increment, there is a rising edge and triggers the Discard Save Frame upon Success block to processes the packet. If acknowledgment signal decrement, there is a falling edge and triggers the Discard Save Frame upon Success block to processes the packet. ReTX output port will Path Combiner block to attempt retransmission.

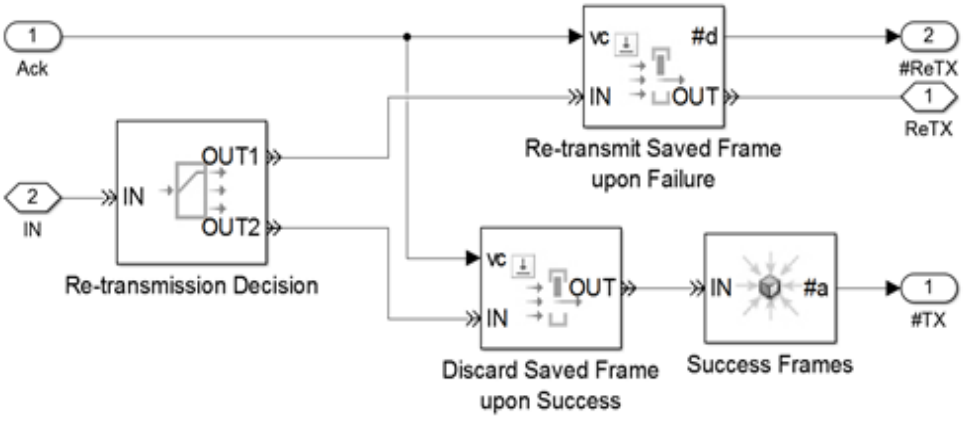

Figure 18. Structure of retransmission buffer block

Error detection block is critical block in CAN Node structure. It has two output port, Status and Ack. Status represents the status of the receiving packet, whether collision occurs with higher priority packet. Ack represent the status of the transmitting packet, whether collision occurs with higher priority packet. Only the packet with higher priority can proceed and the packet with lower priority packet need to retransmit or discard.

After the model is developed, model evaluation takes place. Model evaluation is used to ensure the model can transmit and receive packet and hence imitate the Ethernet protocol. Trial and error method is used for model evaluation. Several test runs are performed and any error and bug is addressed and solved. After model evaluation, the project transition to model validation stage. Model validation is done using Model Advisor in Figure 3.20. Model Advisor is a tool in Simulink to check for common mistakes and best practices in the model. It produces warning and error that not fulfill model correctness and best practices.

3. RESULTS AND ANALYSIS

Ethernet network model and CAN network model are simulated. For Ethernet network, different back-off algorithms mentioned earlier are implemented and simulated. Then, all the network models are extended from 3 nodes, to 10 nodes and 15 nodes to do simulation. The results are recorded and tabulated. The results are presented in chart form and analysis and discussion are made. From the discussion, it can be observed that different network protocol, different back-off algorithm, and different number of nodes affect the performance in certain ways.

3.1 Assessment of the MAC Controller and Back-off Scheme of Embedded Ethernet Model

There are three embedded Ethernet model with different number of nodes, that are 3 nodes, 10 nodes and 15 nodes. The parameter of Ethernet model is set up as below for simulation and data collection. For Application block, rate of packet generation is set to 100 packets/s. Packet size is set to 64 bytes. For MAC Controller block, transmission buffer size is set to 25 packets. The channel bandwidth is set to 10M bits/s. For cable, length is set to 100m. For these configurations, simulation is run for 300s to get a comprehensive result. There are three cases of simulation, that are 3 nodes, 10 nodes, and 15 nodes. In all cases, four different back-off algorithms are considered. The purpose is to discuss the effect of different back-off algorithm in different number of nodes environment.

The three nodes Ethernet model with different back-off algorithm is simulated. Figure 19 shows overall average delay and overall jitter. When average delay and jitter are drawn together, it is understood that the different in average delay between different algorithms is low. The different in jitter between different algorithms is larger. The jitter of Linear and Logarithm Back-off Algorithm is much higher compared to BEB Algorithm and Exponential-Linear Back-off Algorithm. The reason is they increase back-off time more rapid during the first few collisions.

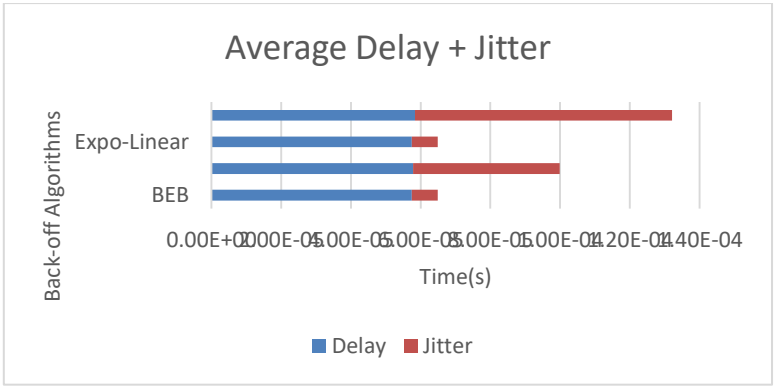

Figure 19. Overall average delay and overall jitter in 3 nodes Ethernet model

The Ethernet model is extended to 10 nodes. Figure 20 shows overall average delay and overall jitter. It can be observed that all modified back-off algorithm has slightly improved in term of average delay, but has significantly improved in term of jitter, compared to BEB Algorithm. The most obvious instance is the difference between jitter of BEB Algorithm and Exponential-Linear Back-off Algorithm. BEB Algorithm has highest jitter of 3.95ms and Exponential-Linear Back-off Algorithm has lowest jitter of 0.211ms.

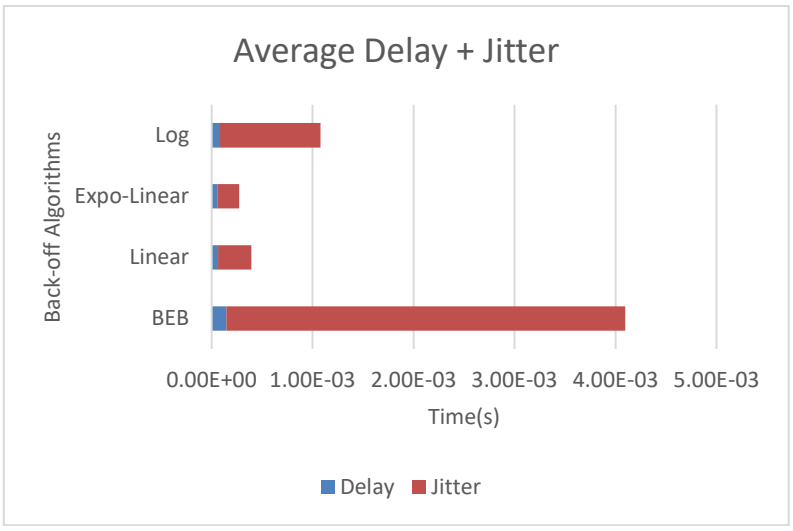

Figure 20. Overall average delay and overall jitter in 10 nodes ethernetmodel

The Ethernet model is extended to 15 nodes. Figure 21 shows overall average delay and overall jitter in a chart. It can be observed that when there are 15 nodes in network, jitter is more important criteria as it is much higher than average delay. BEB Algorithm has extremely high jitter, 5.71ms while other algorithms have relatively low jitter, less than 1ms. It is observed that when number of nodes increases, both average delay and jitter increase, as number of collisions increase. It is interesting to note that for 3 node Ethernet model, the jitter is extremely low, 0.0327ms as only very few collisions occur.

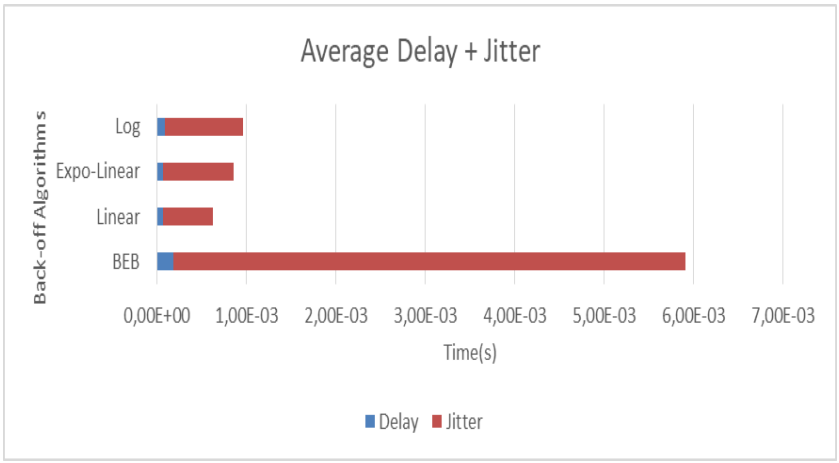

Figure 21. Overall average delay and overall jitter in 15 nodes Ethernet model

3.2 Assessment of CAN Model

The CAN model with 3 nodes, 10 nodes and 15 nodes is simulated. The parameter of CAN model is set up as below for simulation and data collection. For Application block, rate of packet generation is set to 100 packets/s. For CAN Node block, packet size is set to 8 bytes. For CAN Bus block, the channel capacity is set to 1M bits/s and loss probability is set to 0.001. For these configurations, simulation is run for 300s to get a comprehensive result. There are three cases of simulation, that are 3 nodes, 10 nodes, and 15 nodes. The purpose is to discuss the performance of CAN protocol in different number of nodes environment. Figure 22 shows chart with both overall average delay and overall jitter. It is interesting to note that in all cases, jitter is always lower than average delay. It can be understood that CAN model has good result in term of jitter.It shows that CAN model has advantage in bounded delay, the delay does not deviate far from average.

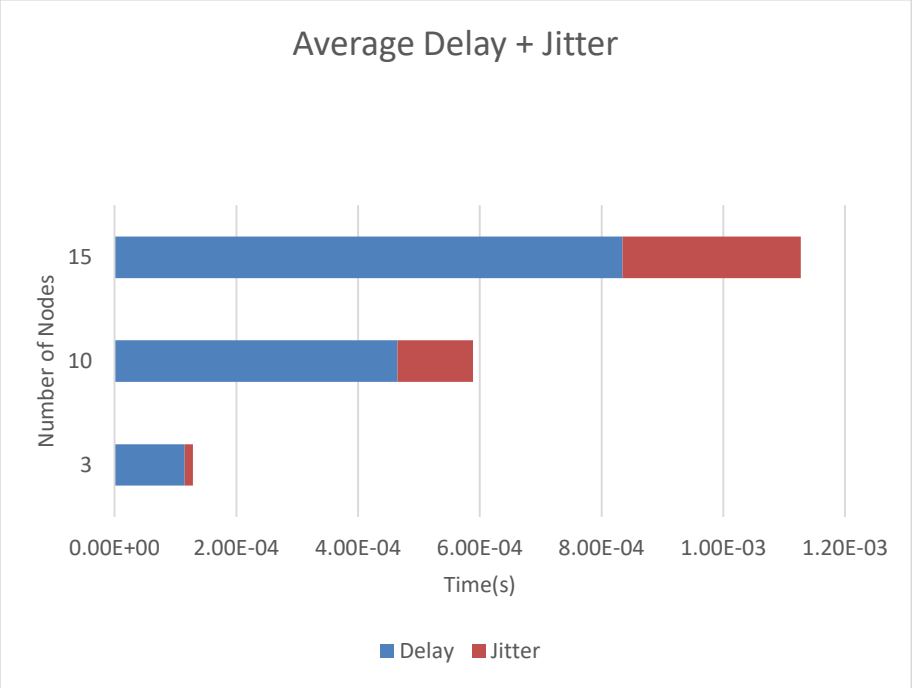

Figure 22. Overall average delay and overall jitter of CAN model

4. CONCLUSION

CSMA/CD Ethernet network and CAN network are modeled using MATLAB and Simulink and they are able to imitates the behaviour of the original protocols. Different back-off algorithms are studied and implemented in Ethernet network model. The second objective is achieved. All the network models are extended from 3 nodes, to 10 nodes and 15 nodes. The key findings in this project are BEB Algorithm has lowest average delay and jitter in 3 nodes network. Exponential-Linear Back-off Algorithm has lowest average delay and jitter in 10 nodes network. Linear Back-off Algorithm has lowest average delay and jitter in 15 nodes network. However, the difference between results of Exponential-Linear and Linear Back-off Algorithm are very low. For CAN network, jitter is much lower than average delay due to its deterministic nature.

ACKNOWLEDGEMENTS

The work was supported by the Short Term Grant UniversitiSains Malaysia with grant No.PELECT 60313037.

REFERENCES

[1] J. A. Stankovic, T. E. Abdelzaher, L. Chenyang, S. Lui, and J. C. Hou, "Real-time communication and coordination in embedded sensor networks," *Proceedings of the IEEE*, vol. 91, pp. 1002-1022, 2003.
[2] Z. King and Y. Shucheng, "Investigating and securing communications in the Controller Area Network (CAN)," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 814-818.
[3] A. Aly, E.-S. Zeidan, A. Hamed, and F. Salem, *An Antilock-Braking Systems (ABS) Control: A Technical Review* vol. 02, 2011.
[4] L. Feng-Li, J. R. Moyne, and D. M. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE Control Systems*, vol. 21, pp. 66-83, 2001.
[5] K. Seok-Kyu and K. G. Shin, "Statistical real-time communication over Ethernet," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 322-335, 2003.
[6] M. K. Ishak, G. Herrmann, and M. Pearson, "Reducing delay and jitter for real-time control communication in Ethernet," in *2013 15th International Conference on Advanced Communications Technology (ICACT)*, 2013, pp. 162-168.
[7] M. K. Ishak, G. Herrmann, and M. Pearson, "Performance evaluation using Markov model for a novel approach in Ethernet based embedded networked control communication," in *2016 Annual IEEE Systems Conference (SysCon)*, 2016, pp. 1-7.

BIOGRAPHIES OF AUTHORS

Ching Chia Leong was born in Pulau Pinang, Malaysia. He received his Bachelor of Electrical and Electronic Engineering from UniversitiSains Malaysia Engineering Campus in 2007. He is currently a Master student at UniversitiSains Malaysia Engineering Campus.

Dr Mohamad Khairiishak received his Ph.D. degree from the University of Bristol,UK, in 2015. He joined the UniversitiSains Malaysia, as a Senior Lecturer in April 2015 at the School of Electrical and Electronic Engineering at USM. He is a Member of the IEEE and a member of the IET. His research interest covers the development and application of embedded system and real-time control communication systems.