MTU Assignment in a Master-Slave Switched Ethernet Network

Mohammad Ashjaei, Moris Behnam, Thomas Nolte
Mälardalen University, Västerås, Sweden
{mohammad.ashjaei, moris.behnam, thomas.nolte}@mdh.se

Luis Almeida
IT / DEEC, University of Porto, Portugal
lda@fe.up.pt

Abstract—**In this paper, we investigate the problem of selecting the Maximal Transmission Unit (MTU) size that maximizes the schedulability of real-time messages. We focus on a bandwidth-efficient master-slave switched Ethernet protocol, namely the FTT-SE protocol. We propose an algorithm to find the MTU for each message in order to maximize the schedulability of the messages. Moreover, we evaluate our proposed algorithm and we show that setting the MTU for messages using the algorithm increases the schedulability of messages compared with assigning the MTU to the maximum value that the protocol can support.**

I. Introduction

Nowadays, there is an increasing demand towards using high performance network solutions for real-time Networked Embedded Systems (NES) due to the growth of the number of nodes in such systems, their increased amount of functionalities and the high amount of information being transmitted between the nodes. Ethernet has been proposed as an interesting technology for such systems as it provides high throughput, low cost, wide availability and general maturity. To overcome the limitation of Ethernet with respect to real-time guarantees, it has been complemented with suitable transmission control mechanisms, being the base for several real-time communication protocols currently used in NES, such as PROFINET, Ethernet POWERLINK, TTEthernet and FTT-SE. To keep the high performance of the Ethernet based protocols, the network should be configured properly and one of the configuration parameters that has a significant effect on the performance of the protocols is the selection of the maximum packet size of messages [1].

In the area of Ethernet protocols, a packet is defined to hold up to 1500 data bytes which is relatively high compared with other communication technologies. As a configuration parameter, the maximum data size that a packet can hold in Ethernet is called the Maximal Transmission Unit (MTU), which has a big impact on the performance of the network and the bandwidth utilization. For instance, the MTU size affects the minimum slot time in TTEthernet [2], while in cyclic base protocols, such as the FTT-SE protocol [3], it affects the size of the idle time, which is considered to prevent overruns between cycles. In this paper, we mainly focus on the FTT-SE protocol which is based on a master-slave switched Ethernet technology.

Considering industrial real-time applications, the amount of data to be transmitted can vary from small to very large, for instance, the flow size might be rather large for automation applications based on video streams or machine vision. Therefore, the data of such applications should be fragmented to several packets to be transmitted sequentially. However, selecting the best MTU that guarantees the real-time requirements for all messages is challenging. On one hand, a larger MTU will reduce the number of packets needed to transmit messages which in turn reduces the total transmission time of messages due to a lower amount of overhead associated with Ethernet packets. On the other hand, the large MTU increases the idle time used in every cycle to prevent overruns which in turn decreases the efficiency of the protocol. This contradicting effect has been discussed in [1] for the FTT-SE protocol and two algorithms (optimal and simplified) have been proposed to find the optimum MTU for all messages. The algorithms are based on the utilization bound schedulability test and they only consider the effect of messages that share the same destination node, while the impact of other messages that might delay the transmission of messages has not been included in the analysis.

In this paper, we generalize the solution presented in [1] by including the effect of all messages that can delay the transmission of messages. In addition, we propose an algorithm based on the response-time schedulability analysis to find a proper MTU for each message to increase the schedulability of systems. We show that the proposed algorithm increases the schedulability compared with the case when the MTU is set (configured) to the maximum Ethernet packet in the network.

The rest of the paper is structured in the following way. Section II presents related work. Section III outlines the basics of the FTT-SE protocol. Section IV presents the system model and Section V sketches the schedulability analysis. Moreover, Section VI proposes the heuristic algorithm, while Section VII shows the evaluation of the algorithm. Finally, Section VIII concludes the paper and presents the future work.

II. Related Work

The problem of fragmenting messages into smaller packets transmitted over large heterogeneous networks has been discussed in [4]. In such networks, some routes can carry limited packet size and large messages should be fragmented leading to a higher protocol overhead and a lower throughput. For such a problem, optimal routing techniques are developed to avoid message fragmentation as much as possible.

In the context of wireless networks, having several small packets degrade the throughput of the network due to the protocol overhead inherent to the transmission of each packet.

On the other hand, using a large packet size may also affect the efficiency due to retransmission of faulty packets. Therefore, optimal solutions have been proposed in [5] and [6] in the area of wireless networks. Moreover, in [7] an algorithm to dynamically adjust the packet size in multi-level security wireless networks is proposed in which the goal is to minimize the overhead in each packet. The same goal as finding the optimized packet length for wireless sensor networks is presented in [8], where the criterion for optimization is energy efficiency rather than the bandwidth efficiency.

However, the above proposed solutions are not applicable in this paper as the source of the problem and the goals are different where we focus mostly on real time guarantees.

The work presented in [9] proposed an algorithm to select optimal preemption points in order to increase the schedulability of real-time tasks. The criteria to select the optimum preemption points is based on decreasing the overhead of task preemption and the blocking from lower priority tasks on the higher priority tasks. Adding preemption points inside the execution of tasks can be modeled as fragmenting the tasks into a set of subtasks which is similar to fragment messages into a set of packets. Nevertheless, the proposed algorithm is not suitable for our case as it tries to optimize the non-preemptive regions (between two preemption points) of lower priority tasks that block the execution of higher priority tasks. While in our case selecting the MTU of higher priority message can contribute to their transmission time and also the idle time included on every scheduling cycle for each message affecting the schedulability of all messages.

The work presented in [1] is the most related work, where two algorithms were proposed to find one optimum MTU for all messages in the scope of the FTT-SE protocol. However, the presented algorithms are based on the utilization bound schedulability and do not consider all messages that can delay the considered messages. In this paper, we use a tighter schedulability test based on response time analysis and we consider all messages that can interfere with the messages under consideration. In addition and to improve the efficiency, we assign an individual MTU for each message unlike the previous work where only one MTU is assigned for all messages. Note that the algorithms presented in [1] can only give optimal results for very simple cases assuming that all messages are forwarded to the same destination. Otherwise, a very high computational complexity algorithm is required to find the optimal solution. In this paper, we propose a heuristic algorithm based on the response time analysis to find MTUs for all messages that keep the system schedulable, i.e., all messages meet their deadlines.

III. The FTT-SE Basics

The FTT-SE protocol [3] is an Ethernet real-time communication protocol that uses a master-slave technique to coordinate all traffic in the network. This protocol supports both synchronous and asynchronous traffic. The former is time-triggered and activated by the scheduler according to its period, whereas the latter traffic is issued by applications in the nodes.

The master node organizes the traffic in fixed time slots called Elementary Cycles (EC) and broadcasts a specific message, which is called the Trigger Message (TM), at the beginning of the EC. The scheduling of messages is carried out on-line according to some suitable scheduling policy, and the scheduled messages are encoded into the TM. The network nodes receive the TM, decode it and initiate the transmission of messages.

As depicted in Figure 1, the data communication in each EC is divided into two specific windows to handle synchronous and asynchronous traffic, which is called the synchronous window and asynchronous window respectively. Once the nodes in the system receive the TM, the time they need to decode it and initiate the transmissions is called turn around time (TRD).

The asynchronous messages make use of a signaling mechanism that allows the master to become aware of them and consider them in its internal traffic scheduling [10]. The signaling mechanism is based on so-called signaling messages (SIG) sent by the nodes to the master node, informing it of the status of the nodes queues. Whenever an asynchronous message becomes active in one node, this node informs the master in the next SIG message that it sends to schedule the asynchronous messages in the upcoming ECs, e.g., the messages A and B in Figure 1.

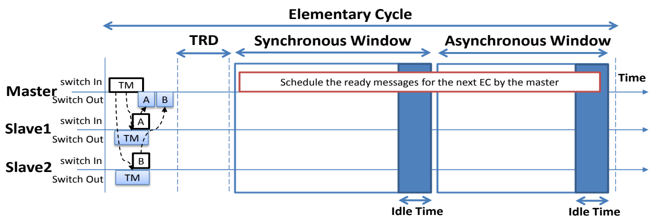

Fig. 1. The FTT-SE Elementary Cycle

The FTT-SE protocol automatically fragments large messages into several packets that are scheduled sequentially by the master node.

IV. System Model

In this paper, we consider the real-time periodic model to describe both synchronous and asynchronous messages as presented in the following set:

$$\Gamma = \{m_i(C_i, D_i, T_i, S_i, Ds_i, MTU_i, nP_i), i = 1..N\} \qquad (1)$$

In this set, C_i is the total transmission time of the message including all physical layer overheads such as inter-frame gap, D_i and T_i are the relative deadline and period of messages respectively, which are presented as integer number of ECs. Moreover, S_i is the source node and Ds_i is the destination node of the message (we assume unicast streams in this paper).

Also, MTU_i is the maximum packet size among the packets that compose m_i and nP_i is the number of packets. We model both, synchronous and asynchronous messages, with the same set in which T_i presents the minimum inter-arrival time for asynchronous messages. In this paper we assume that Γ is sorted by the descending priority of the messages. Finally, the fixed priority scheduling algorithm is used to schedule messages and the priorities of messages are assigned according to the Rate-Monotonic (RM) algorithm.

The switches are assumed to be Commercial Off-The-Shelf (COTS) and cut-through switching and ready messages in switches are scheduled using the First In First Out (FIFO) approach. We also consider the switch relaying latency (Δ) in the schedulability analysis.

According to the FTT-SE protocol, all messages which are scheduled to be transmitted in one EC should be received by the end of the EC. In order to prevent any overrun of the traffic, a message that cannot be fully transmitted within the transmission window is suspended for the next EC, e.g., m_1 in Figure 2.

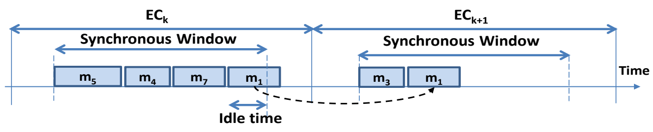

Fig. 2. The Idle Time Presentation

This property introduces an idle time in each transmission window that should be taken into account in the schedulability analysis.

V. Schedulability Analysis

In the FTT-SE protocol, the system is schedulable when all messages meet their deadlines. The schedulability is investigated by computing the response time (RT) for all messages. The system is guaranteed to be schedulable if $\forall m_i : RT(m_i) \leq D_i$. In addition, the FTT-SE scheduling is based on reserving a bandwidth every EC for each type of messages, both synchronous and asynchronous, which is similar to the periodic resource model presented in [11]. Therefore, in order to calculate the response time analysis, we perform the analysis based on a *request bound function (rbf)* and a *supply bound function (sbf)*.

The $rbf_i(t)$ represents the maximum load generated by m_i and all higher priority messages that can delay m_i within the time interval $[0,t]$. Therefore, the $rbf_i(t)$ is calculated by summing the total transmission time of the message itself, the interfering messages and the remote load interference denoted by $W_i(t)$ which will be discussed later in this section. The $rbf_i(t)$ computation is presented in (2), where $hp(m_i)$ is the set of messages with priority higher than that of m_i.

$$rbf_i(t) = C_i + \Delta + \sum_{\substack{\forall m_j \in hp(m_i) \wedge \\ (S_j = S_i \vee Ds_j = Ds_i)}} \lceil \frac{t}{T_j} \rceil C_j + W_i(t) \qquad (2)$$

Besides the interference of the messages that share links with the message under analysis m_i (i.e., $\forall m_j \in hp(m_i) \wedge (S_j = S_i \vee Ds_j = Ds_i)$), the message m_i may still be delayed indirectly through other messages. To show this effect let us consider the example illustrated in Figure 3.

In Figure 3, m_1 is transmitted from Node A to Node B, m_4 is sent from Node A to Node C and m_3 is transmitted from Node B to Node C and m_2 is sent from Node B to Node A. In this example we focus on m_4 and we assume that it is the lowest priority among the other messages. In this scenario, m_1 is delaying m_4 which cause delay in m_3 reception. If m_1 was not scheduled for this EC, it would be possible for m_3 to be transmitted in the EC. Therefore, m_1 delays m_3 even though they do not share links. We can call this effect remote load delay. Since m_4 has the lowest priority, the scheduler in the master node will suspend the transmission of m_4 to the later EC.

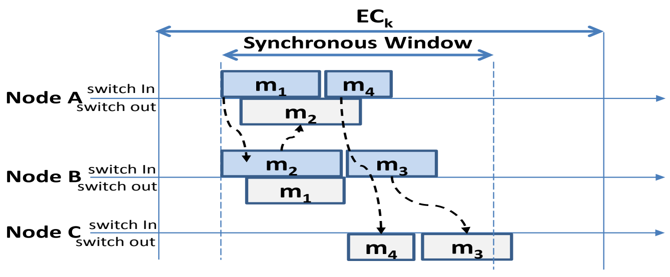

Fig. 3. The Remote Load Interference

To consider this delay in the analysis, all higher priority messages that share source node with the interfering messages are considered as higher priority interfering messages in the response time analysis as shown in (3).

$$W_i(t) = \sum_{\substack{\forall m_k \in hp(m_j) \wedge S_k = S_j \\ \wedge \forall m_j \in hp(m_i)}} \lceil \frac{t}{T_k} \rceil C_k \qquad (3)$$

The sbf(t) is the minimum effective communication capacity that the network supplies within the time interval $[0,t]$. Note that in each EC, a particular bandwidth is provided for transmitting each type of message which is imposed by $LSW - I$, where LSW is the length of the synchronous window and I is the idle time in that window. The idle time is upper bounded by the maximum packet size among the higher priority messages and the message under analysis. Thus, for the message m_i the supply bound function $sbf_i(t)$ is computed in (4).

$$\begin{aligned} sbf_i(t) &= (\tfrac{LSW - I_i}{EC}) \times t \\ I_i &= \max_{\forall m_j \in hp(m_i)} (MTU_i, MTU_j) \end{aligned} \qquad (4)$$

The response time of m_i is computed based on (5).

$$t^* = min(t > 0) : sbf_i(t) \geq rbf_i(t) \tag{5}$$

In order to determine t^*, the inequality should be checked in all instants that $rbf_i(t)$ changes due to interference of other messages up to D_i. Therefore, a set of check points is given by (6).

$$\begin{aligned} &CP_{rbf_i} = [\cup cp_{m_a}, \forall_{m_a \in hp(m_i)}] \cup D_i \\ &where, cp_{m_a} = T_a, 2T_a, ..., nT_a, n = \lfloor \tfrac{D_i}{T_a} \rfloor \end{aligned} \tag{6}$$

Finally, we compute the response time in number of ECs which is given by (7).

$$RT(m_i) = \lceil \frac{t^*}{EC} \rceil \tag{7}$$

The analyses explained above are suitable for the synchronous messages and the asynchronous messages. However, for asynchronous messages additional 2 EC delay should be added to the RT. The reason for this is that the request for asynchronous messages may have to wait 1 EC before the node signals it in the next SIG and the master then executes the scheduling one EC before the respective dispatching.

VI. MTU Assignment Algorithm

The maximum and minimum possible packet transmission times are limited to MTU_{max} and MTU_{min} which are defined according to the protocol specification. In this section we present an algorithm to find the MTU_i within $[MTU_{min}, MTU_{max}]$ such that the system becomes schedulable.

According to the schedulability analysis presented in Section V, the selection of MTU_i affects the response time analysis as it influences the bandwidth utilization in $sbf_i(t)$ through the idle time. Increasing MTU_i might increase I_i in (4) for message m_i and the other lower priority messages that share the same destination node. As a result, increasing I_i will decrease the $sbf_i(t)$ and hence decreasing the schedulability of the message. Moreover, increasing MTU_i will require less packets to transmit the data which in turn will decrease the total transmission time of the message and it will decrease $rbf_i(t)$ and as a result it will increase the schedulability of the system. Considering these two contradicting effects in the schedulability analysis, we may conclude that there is a tradeoff between decreasing the effect of idle time and the protocol overhead when changing the MTUs of messages.

Looking at (4, 2), we can conclude that selecting the MTU for a message not only affects the response time of that message itself but it affects the schedulability of the lower priority messages through the higher priority interference and remote load interference delay. In order to find the optimum solution a combination of all possible MTU ranges for all messages should be checked which requires an algorithm with an exponential computational complexity. Thus, in this section we present a heuristic algorithm to find the MTU_i.

In order to present the effect of MTU_i in the request bound function, the total transmission time of the message is formulated based on the MTU_i. The total message transmission time C_i includes the actual data transmission time C_i^* and the protocol overhead O. The protocol overhead O is a constant value which is added to each packet separately and includes Ethernet overhead, the FTT-SE protocol overhead and the inter-frame gap between the packets. Therefore, the total message transmission time equals to $C_i^* + nP_i \times O$. Note that, we consider the actual transmission time of a message equally split among its packets. This helps avoiding residual short packets and leads to increase the schedulability as described before. Thus, the MTU_i is evaluated in (8).

$$MTU_i = \lceil \frac{C_i^*}{nP_i} \rceil \tag{8}$$

The number of packets for each message can be expressed as in (9).

$$nP_i = \lceil \frac{C_i^*}{MTU_i} \rceil \tag{9}$$

We can reformulate the total message transmission time C_i to be a function of MTU_i by considering nP_i from (9). Moreover, we can approximate the equation by removing the ceiling in the equation which is presented in (10).

$$C_i = C_i^* + (\frac{C_i^*}{MTU_i} + 1) \times O \tag{10}$$

We expand the inequality (5) by substituting the $sbf_i(t)$ from (4) and the $rbf_i(t)$ from (2). Also, the transmission time C_i in $rbf_i(t)$ can be replaced with (10). Due to the max function in the $sbf_i(t)$, we need to evaluate the inequality in two different conditions.

In the first condition, we assume that the MTU_i is greater or equal to the maximum MTU of all higher priority messages than that of m_i. Therefore, a quadratic equation is derived as a function of MTU_i as it is presented in (11).

$$\tfrac{t}{EC} MTU_i^2 + (C_i^* + M(t)) \times MTU_i + (C_i^* \times O) \leq 0 \tag{11}$$

$$\begin{aligned} M(t) = \quad & O + \Delta - \tfrac{LSW \times t}{EC} + \sum_{\substack{\forall m_j \in hp(m_i) \, \wedge \\ (S_j = S_i \vee Ds_j = Ds_i)}} \lceil \frac{t}{T_j} \rceil C_j \\ & + \sum_{\substack{\forall m_k \in hp(m_j) \, \wedge \, S_k = S_j \\ \wedge \forall m_j \in hp(m_i)}} \lceil \frac{t}{T_k} \rceil C_k \end{aligned} \tag{12}$$

Note that, the quadratic equation (11) has two solutions which shows a range of solutions that satisfies the inequality. Moreover, the coefficient of MTU_i^2 is always positive as t and EC are always positive integers. Therefore, the parabola opens upwards in this case, i.e., the quadratic has a minimum value. As a result, given $MTU_i[lo]$ and $MTU_i[hi]$ as lower value and higher value of the solutions respectively, that make the left side of the equation equal to zero, all the values within the range $[MTU_i[lo], MTU_i[hi]]$ guarantee the schedulability of that message.

If the primitive condition is not satisfied, i.e., MTU_i is less than the maximum MTU of all higher priority messages, we need to evaluate MTU_i using (13) and (14).

$$MTU_i \leq \frac{-C_i^* \times O}{C_i^* + L(t)} \tag{13}$$

$$L(t) = M(t) + \frac{t}{EC} \times \max_{\forall m_j \in hp(m_i)} (MTU_j) \tag{14}$$

If the evaluated MTU_i from (13) satisfies the assumed condition, i.e., $\forall m_j \in hp(m_i) : MTU_i < max(MTU_j)$, then the solution is accepted, otherwise there is no solution to make the message schedulable.

Algorithm 1 shows an algorithm to find the MTU for all messages in order to make the system schedulable. As we have seen above, selecting MTU for a message always affects itself and the lower priority messages. Therefore, the algorithm starts from the highest priority message and it continues to the lowest priority messages. Algorithm 1 starts with calculating the MTU range for m_1 based on (11). As m_1 is assumed to be the highest priority message in the set, the set of MTUs (MTU_{hp}) from messages with priority higher than m_1 is set to zero. Moreover, for all messages the MTU_i is calculated when t is assigned to the deadline of the message, i.e., $t = D_i$.

Algorithm 1 MTU Assignment Algorithm

```
1: //Find the range of $MTU_1$ according to (11)
2: $t = D_1, MTU_{hp} = 0$
3: $sched = -1$
4: $MTU_1[hi,lo] = MTUcalc(t, MTU_{hp})$
5: $MTU_1[hi,lo] = CheckRange(MTU_1[hi,lo])$
6: //Change MTU to nP according to (9)
7: $nP[min,max] = translate(MTU_1[hi,lo])$
8: for $i = nP[min] \to nP[max]$ do
9:     for $m_j = m_2 \to m_N$ do
10:        $t = D_j, MTU_j = 0$
11:        //Find the range of $MTU_j$ according to (11), (13)
12:        $MTU_j[hi,lo] = MTUcalc(t, MTU_{hp}[hi])$
13:        $MTU_j[hi,lo] = CheckRange(MTU_j[hi,lo])$
14:        //If there is a solution, set the flag
15:        if $MTU_j[hi] > 0$ then
16:            $sched = 1$
17:            $update(MTU_{hp})$
18:        else
19:            $sched = -1$
20:            break
21:        end if
22:    end for
23:    //If there is a solution, no need to check other nP
24:    if $sched == 1$ then
25:        break
26:    end if
27: end for
28: return $MTU_j, sched$
```

The $[MTU_1]$ range is evaluated and the algorithm checks the range with the maximum possible protocol range of MTU, i.e., the range should be within $[MTU_{min}, MTU_{max}]$ (lines 4 and the following). Afterwards, we need to find the value within the evaluated MTU range such that all other lower priority messages are schedulable. Therefore, the algorithm iterates for all possible number of packets (only for m_1) from the evaluated range of the MTU_1 according to (9) (line 7) starting from the highest down to the lowest value of MTU. Given the value of MTU_1 the algorithm checks the schedulability of other messages starting from message m_2 the second highest priority to evaluate its MTU and then continue with the other messages.

In the message iteration, the algorithm calculates the range of the MTU_j for each message according to (11, 13) considering the highest evaluated MTU values of higher priority which is denoted by $MTU_{hp}[hi]$ (lines 12 and the following). This value is already computed for m_1 and will be calculated for other messages in the loop, then it is updated to be used later in the calculations of MTU for the other lower priority messages(line 17). For instance, in the iteration of m_3, the $MTU_{hp}[hi]$ includes both $MTU_2[hi]$ and MTU_1 which are calculated in the previous iterations. The loop continues for other messages unless the solution is not found with the evaluated values of the MTU_{hp}.

Whenever the algorithm does not find a range for at least one of the messages, it breaks the inner loop and it continues for the next number of packets in the outer loop (lines 18 and the following). Otherwise if the range is evaluated for all messages, the algorithm stops the outer iteration and returns the MTU_j for all messages (lines 24). When computing MTU_j the algorithm considers the highest evaluated values of the MTUs ($MTU_{hp}[hi]$) of the higher priority messages $m_2, ..m_{j-1}$. The reason for this is that the higher value of MTU requires lower number of packets which may lead to better response times of messages.

The complexity of the algorithm is $O(N \times nP)$ where nP is a function of MTU which is between $[MTU_{min}, MTU_{max}]$. Note that the presented algorithm is not an optimal algorithm and it is sufficient but not necessary meaning that if it does not find a solution for a system it does not mean than there is no solution for that system.

VII. Evaluation

In this section, we evaluate the improvements that can be achieved by the presented algorithm in terms of increasing the schedulability of messages and we compare the results of the algorithm with the results of using the maximum protocol's MTU (MTU_{max}) for all messages. The evaluation is carried out using four different simulation studies. In each study, the algorithm is applied on a number of randomly generated message sets given the following parameters as input to the message sets generation program. The range of the message period is defined $[T_i^{min}, T_i^{max}]$, the number of messages is denoted by N and the transmission time of the messages is selected within $[C_i^{min}, C_i^{max}]$.

For each study, 100000 message sets are randomly generated given the range of the above mentioned input parameters.

In all studies the following assumptions are made: the network capacity is set to $100Mbps$, number of slave nodes in the network is 5, the elementary cycle duration is $EC = 1.5ms$, the protocol overhead is 44 bytes including Ethernet overhead and the FTT-SE overhead, and the Ethernet inter-frame gap is $96bits$ which makes $O = 3.96\mu s$, and the switch latency is considered $\Delta = 5\mu s$. Finally, the minimum and maximum packet size is $[100, 1500]$ bytes. Moreover, in all studies only synchronous messages are considered and for each study 20 different synchronous window durations LSW are selected from; $LSW = [100, 1000]\mu s$ in steps of $50\mu s$, where the 100000 sets are tested for every value of LSW.

The different settings in each study are:

- **Study 1** is specified to have $N = 10$, $[T_i^{min}, T_i^{max}] = [2 \times EC, 50 \times EC]$ and $[C_i^{min}, C_i^{max}] = [150, 200]\mu s$.
- **Study 2** is done with the same parameters as Study 1, except the number of messages which is $N = 30$.
- **Study 3** is specified having similar parameters in Study 1, except that the message transmission times are increased within $[C_i^{min}, C_i^{max}] = [200, 500]\mu s$.
- **Study 4** is performed having the same range for transmission time of the messages in Study 3, but changing the range of the periods within $[T_i^{min}, T_i^{max}] = [5 \times EC, 80 \times EC]$ and $N = 30$.

In all studies we count the number of scheduled sets out of the 100000 randomly generated and using the MTU values evaluated from the proposed algorithm and the same study is repeated assuming the MTU_{max} for all messages.

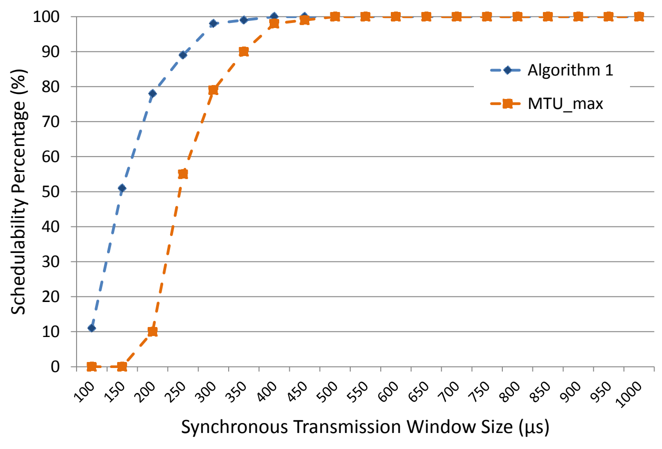

Fig. 4. The Result of Study 1

Figure 4 shows the percentage of schedulable systems in Study 1 as a function of LSW. It is clear from the figure that the results of using the proposed algorithm increase the schedulability of sets significantly compared with the case of assigning the maximum MTU. To guarantee the schedulability of all sets we need $LSW \geq 500\mu s$ for the case of maximum MTU while using the MTU from the proposed algorithm reduces the required length to $LSW \geq 400\mu s$.

In the second study, we changed the number of messages to 30 in each set to investigate the effect of the number of messages on the results. The results of Study 2 are depicted in Figure 5. The result illustrates that using the maximum MTU for the messages cannot reach to 100% meaning that the dedicated synchronous window is not enough. However, assigning MTUs according to the proposed algorithm, makes all sets schedulable even with $LSW \geq 850\mu s$. Note that increasing the number of messages in general requires more bandwidth to be dedicated to the messages to guarantee their schedulability since it increases the number of interfering messages for lower priority messages which is clear when comparing the results of Study 1 and Study 2.

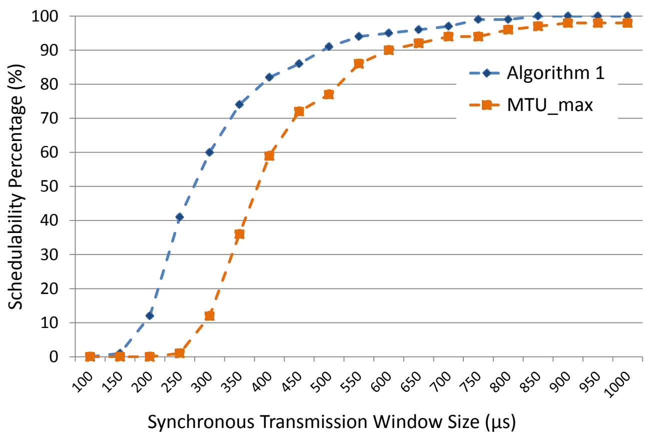

Fig. 5. The Result of Study 2

The results of Study 3 are presented in Figure 6 and it shows that increasing the transmission time of the messages highly affects the schedulability of the sets, yet setting the MTUs based on the presented algorithm makes 100% of the generated sets schedulable by using a synchronous window duration larger than $750\mu s$.

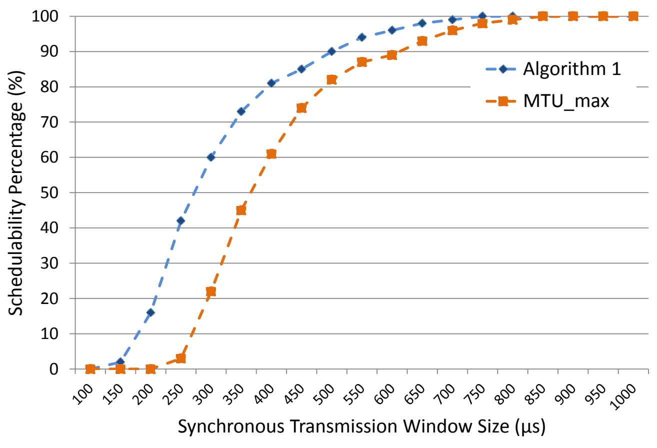

Fig. 6. The Result of Study 3

In the last study, we increase both the number of messages and the period of the generated messages. Figure 7 shows the result of Study 4. As explained previously, increasing the number of messages affects the percentage of schedulable sets. Also increasing the difference between the minimum and maximum periods of messages have the same negative effect on the schedulability since the shorter period messages may activate several times while scheduling the lower priority larger periods which increases the interference from higher priority messages. Applying the algorithm on the message sets makes 100% of the message sets schedulable when the dedicated synchronous window is more than $550\mu s$, whereas using MTU_{max} requires the synchronous window more than $750\mu s$.

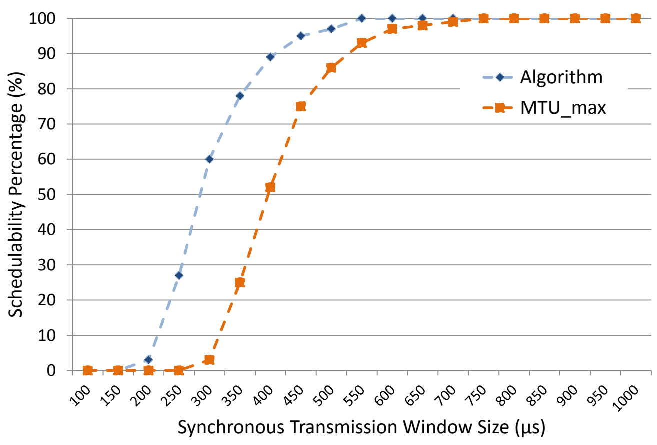

Fig. 7. The Result of Study 5

VIII. Conclusion and Future Work

In this paper, we proposed a heuristic algorithm to find a MTU for each message in order to increase the schedulability of the messages in the FTT-SE protocol. We evaluated the proposed algorithm using 4 different studies and we showed that assigning the MTU of the messages according to the proposed algorithm increases the percentage of schedulable sets compared with using the protocol maximum MTU for all messages. However, this algorithm does not evaluate the optimum MTU for the messages as the complexity is high. The future work aims at finding the MTU in the multi-hop FTT-SE protocol.

Acknowledgments

This work is supported by the Swedish Foundation for Strategic Research, via Mälardalen Real-time Research Center (MRTC) at Mälardalen University. Also, it is partially supported by the Portuguese Government through FCT grant CodeStream PTDC/EEI-TEL/3006/2012.

References

[1] M. Behnam, R. Marau, and P. Pedreiras, "Analysis and optimization of the mtu in real-time communications over switched ethernet," in *16th IEEE International Conference on Emerging Technologies Factory Automation (ETFA'11)*, sept. 2011.

[2] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (tte) design," in *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, may 2005.

[3] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over cots ethernet switches," in *6th IEEE International Workshop on Factory Communication Systems (WFCS'06)*, June 2006.

[4] C. A. Kent and J. C. Mogul, "Fragmentation considered harmful," *SIGCOMM Comput. Commun. Rev.*, Jan. 1987.

[5] J. Chen, L. Gong, Y. Yang, and P. Zeng, "Average performance of packet network," in *6th International Conference on ITS Telecommunications Proceedings*, 2006.

[6] C. K. Kodikara, S. Worrall, and A. Kondoz, "Optimal settings of maximum transfer unit (mtu) for efficient wireless video communications," *IEE Proceedings of Communications*, 2005.

[7] M. Younis, O. Farrag, and W. D'Amico, "Packet size optimization for increased throughput in multi-level security wireless networks," in *IEEE Military Communications Conference (MILCOM'09)*, 2009.

[8] Y. Sankarasubramaniam, I. Akyildiz, and S. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[9] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito, and M. Caccamo, "Preemption points placement for sporadic task sets," in *22nd Euromicro Conference on Real-Time Systems (ECRTS'10)*, 2010.

[10] R. Marau, P. Pedreiras, and L. Almeida, "Asynchronous traffic signaling over master-slave switched ethernet protocols," in *6th International Workshop on Real Time Networks (RTN'07)*, July 2007.

[11] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *24th IEEE International Real-Time Systems Symposium (RTSS'03)*, 2003.