This regular paper was presented as part of the main technical program at IEEE ETFA'2011

Analysis and Optimization of the MTU in Real-Time Communications over Switched Ethernet

Moris Behnam, Ricardo Marau
IT / DEEC
University of Porto, Portugal
{morisb,marau}@fe.up.pt

Paulo Pedreiras
IT / DETI
University of Aveiro, Portugal
pbrp@ua.pt

***Abstract*—The Flexible Time-Triggered communication over Switched Ethernet protocol (FTT-SE) was proposed to overcome the limitation of guaranteeing the real-time communication requirements of conventional switches, and at the same time to support reconfiguration of dynamic adaptive systems. The protocol fragments large messages into a sequence of packets that are individually scheduled. The maximum transmission unit (MTU), that restricts the packets size, has a significant effect on the schedulability of the packets. In this paper, we investigate the problem of selecting the optimal MTU size that maximizes the schedulability of real-time messages. We propose two algorithms to find optimal/sub-optimal values of MTU; the first one finds an optimal solution but exhibits high computational complexity, while the second one is sub-optimal but exhibits a lower computational complexity. Finally, we evaluate our proposed algorithms by means of simulation studies and compare their results with the results of assigning MTU to the maximum packet size that the protocol can allow.**

I. Introduction

Nowadays Networked Embedded Systems (NES) are pervasive, being present in almost all aspects of our daily life. In early days, NES were typically based on highly constrained hardware resources and performed simple functions, typically digital IO and analog data interface. Additionally, specific network protocols, commonly designated by fieldbus, have been developed to address the specificities of NES. Fielbuses normally have a very limited bandwidth, being optimized to support frequent exchanges of small data. The data frequency could reach the KHz range and the maximum packet size could be as low as 8 bytes per packet, e.g. as in the CAN bus.

However, the quantity, complexity and functionality of NES systems has been increasing consistently. Due to this evolution, in certain application domains the amount of information exchanged over the network reached the limits achievable using traditional fieldbuses. Consequently, new protocols have been investigated, many of them based on high bandwidth general-purpose networks. In particular, Ethernet has been complemented with suitable transmission control mechanisms to provide real-time services, being the base for several real-time communication protocols currently used in NES, such as PROFINET, Ethernet POWERLINK, TTEthernet and FTT-SE.

[1]This work was partially supported by the iLAND project, call 2008-1 of the EU ARTEMIS JU Programme an by the HaRTES project, funded by COMPETE/FCT under the name FCOMP-01-0124-FEDER-007220.

Unlike classical fieldbuses, originally optimized for short data transmissions, Ethernet provides a relatively longer packet size, holding up to 1500 data bytes. The maximum data size a packet can hold is commonly known as Maximal Transmission Unit (MTU) and may have a rather noticeable impact in the performance of real-time communication protocols. For instance, in time-slot based protocols such as TTEthernet [11], the MTU constrains the minimum slot duration. Whilst in cyclic-based real-time protocols, such as Ethernet POWERLINK [1] or FTT-SE [10], it constrains the amount of idle time that may be included on every cycle to prevent overruns between cycles.

The dissemination of NES in areas such as machine vision, automated inspection or object tracking based on video streams, brought to the foreground the MTU optimization problem. This class of applications produces large amounts of data. For instance, an MJPG compressed image may easily reach tens or even hundreds of KiB per frame and it requires multiple packets to be transmitted. For this type of data, it would be desirable to use an MTU as large as possible, in order to minimize the overhead associated to the transmission of each Ethernet packet (38 bytes). However, as discussed above, the use of a large MTU penalizes the efficiency of some real-time protocols, raising an optimization problem.

This paper investigates the problem of determining the packet size that optimizes the protocol data-throughput efficiency. This work is developed in the scope of the FTT-SE protocol, but the results obtained should be easily generalizable to other cyclic-based real-time protocols.

Two different algorithms are proposed. One is optimum, but exhibits high temporal complexity, thus may not be suitable for online usage. The other is sub-optimal but exhibits a lower computational complexity, thus being more suitable to online usage.

The remaining of the paper is organized in the following way. Section II outlines related work. Section III presents a brief overview of the FTT-SE protocol. Section IV introduces the system model. Section V presents the schedulability analysis. Section VI the problem formulation. Section VII describes the MTU optimization algorithms proposed in this paper. Finally, Section VIII presents experimental results and Section IX concludes the paper.

978-1-4577-0018-7/11/$26.00

II. Related work

The problem of fragmenting messages into smaller packets transmitted over large networks has been discussed in [5]. 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 problem, optimal routing techniques are developed to avoid message fragmentation as much as possible. A similar problem can happen when transmitting messages through high bit-error rate links such as wireless. Retransmitting too large packet size degrades the efficiency of protocols and the same happens when transmitting small packet size because of the high overhead. Adaptive and optimal solutions have been proposed in [4] and [6]. However, the solutions proposed to solve the fragmentation problems mentioned above are not applicable to our case as the source of the problem and the goals are different from ours.

The most related work from real-time scheduling approaches is the limited preemption approach presented in [3], where an optimal algorithm is presented to select optimal preemption points in each task in order to increase the schedulability of tasks, taking into account the overhead of task preemption and the generated blocking from lower priority tasks on the higher priority tasks. The problem of selecting the optimal preemption points to increase the schedulability of the tasks is similar to the problem of selecting the optimal MTU size. However, the proposed algorithm in [3] is not suitable for our case, as the effect of non-preemptive regions (between two preemption points) from lower priority tasks block higher priority tasks. In our case any message can contribute to the idle time included on every scheduling cycle affecting the schedulability of all messages.

III. FTT-SE protocol

FTT-SE [10] is an academic real-time communication protocol that exploits the Master/Multi-slave paradigm and the advantages brought by Ethernet micro-segmentation, namely the ability to forward packets in parallel and the absence of collisions on a one node per switch-port basis. This protocol uses a master node to coordinate the transmissions of other nodes. The communication is organized in fixed duration slots called Elementary Cycles (ECs), which are triggered by a periodic message (Trigger Message - TM), issued by the Master, containing the schedule for each EC.

The traffic scheduling activity is carried out on-line in the Master, invoked once per EC and disseminated by means of the TM. The EC duration is tunable and can be designed to suit the application needs. Typical EC durations range from 1ms to tens of ms, depending on the application dynamics. Traffic scheduling is centralized, allowing an easy enforcement of any scheduling policy as well as performing atomic changes to the communication requirements. This last feature facilitates the deployment of mechanisms that allow admitting and removing message streams, on-line, under guaranteed timeliness, and mechanisms for dynamic bandwidth management.

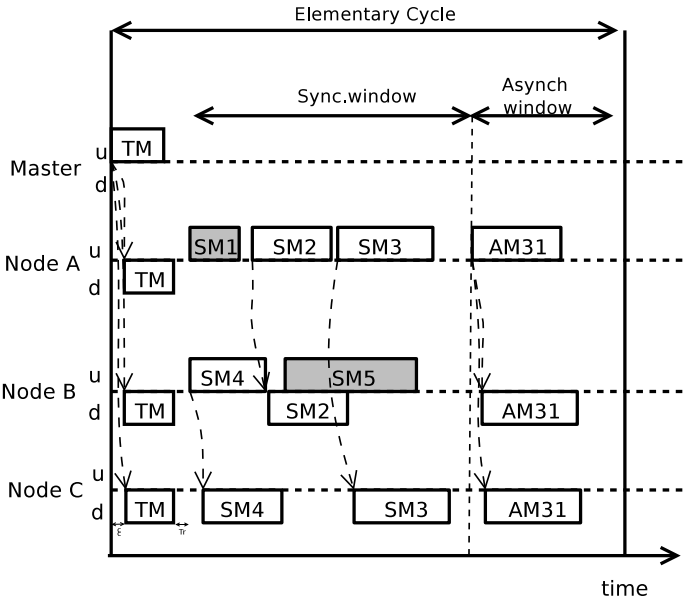

Fig. 1: The FTT-EC structure.

The protocol supports real-time synchronous and asynchronous messages, as well as non real-time traffic. The synchronous traffic is time-driven, being activated autonomously by the scheduler. The aperiodic traffic is event-driven, driven from the distributed application at arbitrary time instants. Messages are locally queued, awaiting transmission while a periodic message updates the activation requests on the Master's scheduler.

As depicted in Figure 1, the EC is organized in three independent time windows, Synchronous, Asynchronous and the slot for the TM. The synchronous and asynchronous traffic is scheduled on the respective time windows, whereas the non real-time traffic takes any transmission slot remaining from both windows in a best-effort policy. The windows have a user-defined maximum duration, which is enforced at run-time by the scheduler. Consequently, no window overruns should occur, providing strict traffic isolation among traffic classes.

Further details about the FTT-SE protocol can be found in [10] and [8].

IV. System model and assumptions

This paper considers a simplified model that describes cyclic-based real-time protocols such as the FTT-SE protocol. The system comprises a set (SM) of N synchronous message streams, each stream (m_i) modeled using the periodic real-time model (Equation (1)).

$$SM \equiv m_i(C_i, D_i, T_i, S_i, R_i, Mmax_i) \quad (1)$$

For each m_i, C_i represents the total transmission time that includes message fragmentation and overheads. We define $Mmax_i$ as the maximum transmission time among the packets that compose m_i. $Mmax_i$ is defined per message and it cannot be greater than the size of a packet holding MTU data bytes. D_i and T_i are the relative deadline and message

period, represented in multiples of ECs. In this paper, we assume implicit deadlines, i.e, $T_i = D_i$ and that streams can be scheduled using Earlier Deadline First EDF or Rate Monotonic RM. Finally, S_i is the source port (i.e. the switch port where the data source is attached to the network) and R_i denotes the receiving port of a unicast transmission.

The above model is also applicable to sporadic messages, with the only difference being that T_i represents the minimum inter-arrival time, i.e., the minimum time between consecutive instances of message m_i. These models represent accurately the real-time synchronous and asynchronous messaging system of the FTT-SE protocol.

The system model considered in this paper also assumes the nonexistence of cycle overruns, which is usual for cyclic-based protocols. Overruns are prevented deferring the dispatch of messages that may exceed the cycle (or time window) capacity. Figure (2) illustrates this scenario, in which message $m48$ does not fit inside the k^{th} elementary cycle and remains in the ready queue. The message is eventually transmitted in the following cycle along with other higher priority messages that activate in the meantime.

The idle-time that results from deferring the message transmission cannot be neglected from the system analysis. This aspect is referred in [2], where it is shown that the maximum amount of idle time that can be inserted in an elementary cycle or window is upper bounded by the transmission time of the largest packet. It is also observed that the relative penalization due to the idle time is related to the ratio between the elementary cycle (or window duration) and the transmission time of the largest packet. This last observation is particularly relevant, since it constrains the design space. The cycle duration is typically imposed at the application design, based on the system dynamics, thus reducing the optimization space to the MTU parameter.

V. Schedulability analysis

The FTT-SE protocol does not allow window overruns, using the inserted idle-time based scheme described in Section IV. In the schedulability analysis, to model the possible scheduling suspension in each scheduling window, an idle time equal to the maximum packet size of all messages that use the link has to be considered, which decreases the effective size of the window. [9] presented a schedulability analysis based on utilization bound for FTT-SE synchronous message scheduling. Considering only the scheduling of a downlink of a switch, the schedulability condition of the link is given by Equation (2).

$$\sum_{\forall i} \frac{C_i}{T_i} \leq U^{lub} \times \frac{LSW - I_{max}}{E} \tag{2}$$

where U^{lub} is the utilization bound assuming either RM or EDF, E is the elementary cycle duration and I_{max} is the maximum idle-time, computed as follows:

$$I_{max} = \max_{\forall i}(Mmax_i) \tag{3}$$

Let us define O as the overhead of the protocol added to each packet. Let us also define nP_i as the number of packets required to transmit message m_i. From the definition of total message transmission time, C_i already includes the data transmission time C_i^* and the protocol overhead, i.e., equal to $nP_i \times O$. Given C_i^*, $Mmax_i$ is evaluated depending on the number of packets nP_i and the overhead O as follows,

$$Mmax_i = \left\lceil \frac{C_i^*}{nP_i} \right\rceil + O \tag{4}$$

Note that, from the definition of MTU, the transmission time of all packets should be lower than the MTU, i.e., $Mmax_i \leq MTU$. Considering the protocol overhead O, nP_i should satisfy the condition $nP_i \geq \lceil \frac{C_i^*}{MTU - O} \rceil$. However, since the overall protocol overhead is directly proportional to the number of packets necessary to transmit a message, the condition is minimized (Equation 5).

$$nP_j = \left\lceil \frac{C_j^*}{MTU - O} \right\rceil \tag{5}$$

Taking into account the protocol overhead O in the analysis, Equation (2) can be rewritten as follows:

$$\sum_{\forall j} \frac{C_j^* + nP_j \times O}{T_j} \leq U^{lub} \times \frac{LSW - I_{max}}{E} \tag{6}$$

VI. Problem formulation

Equation (6) indicates that the schedulability of the messages can be increased either by decreasing the effective load (left hand side of Equation (6)) and/or by increasing the schedulability bound (right hand side of Equation (6)). Furthermore, all parameters of Equation (6), except the number of packets and the inserted idle time, are fixed, thus the schedulability of the message set depends on both these parameters. However, Equations (3), (4) and (5) indicate that the number of packets and inserted idle-time both depend on MTU. Selecting a smaller MTU will increase the number of packets of the messages, and consequently the effective load, while decreasing the idle time. Conversely, selecting a larger MTU decreases the effective load at expenses of an higher idle time. Considering these two contradicting effects in the schedulability analysis, we may conclude that there is a trade-off between decreasing the effect of idle time and the protocol overhead when changing the protocol's MTU.

We can formulate our problem as follows: Given a set of stream messages SM transmitted though the same downlink and scheduled using EDF or RM scheduling algorithm, find the optimal MTU such that the schedulability bound in Equation (6) is maximized. To do so, we will rewrite Equation (6) as a function of the number of packets, as shown in Equation (7)

$$\sum_{\forall j} \frac{C_j^*}{T_j} \leq \frac{U^{lub} \times LSW}{E} - \left(\frac{U^{lub} \times I_{max}}{E} + \sum_{\forall j} \frac{nP_j \times O}{T_j} \right) \tag{7}$$

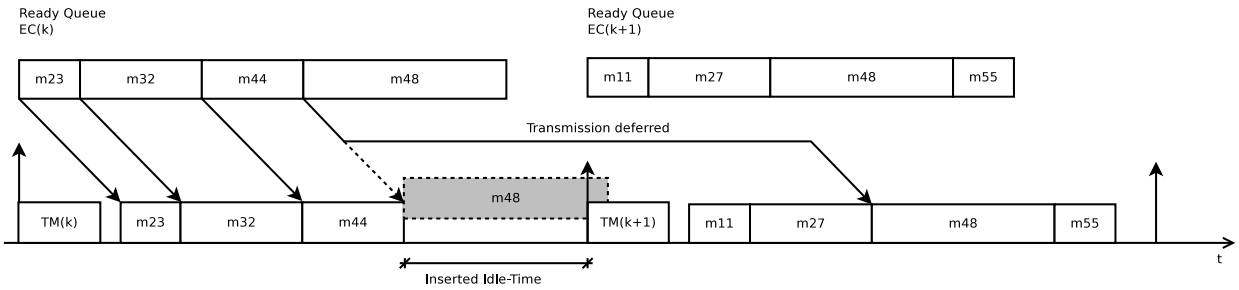

Fig. 2: Inserted idle-time illustration

Then the optimization goal is to minimize function $f(nP)$ in Equation (8), which is as a function of the message packet number nP. Notice that I_{max} is a function of nP, see Equations (3) and (4).

$$f(nP) = \frac{U^{lub} \times I_{max}}{E} + \sum_{\forall j} \frac{nP_j \times O}{T_j} \quad (8)$$

A. Example

For illustration purposes, consider an example with $O = 0.5$ time units, $E = 20$ time units, $LSW = 18$ time units, scheduled with the Rate Monotonic scheduling policy. Consider also the message set shown in the table below

C	2	4	10	20	30
T	20	60	120	200	240

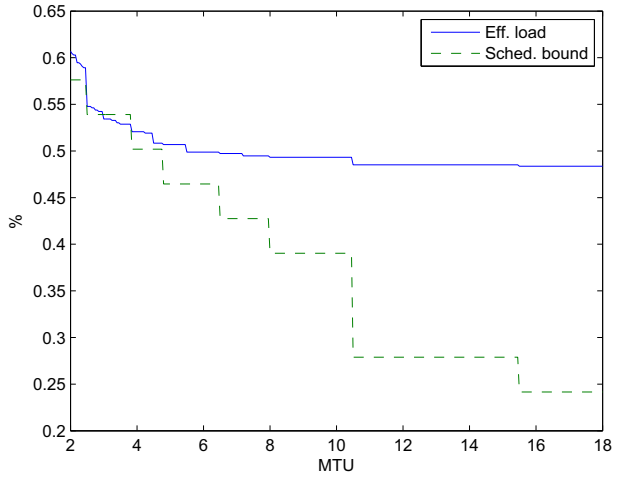

Fig. 3: Effective load and schedulability bound as a function fo the MTU

Figure 3 presents the effective load and the effective schedulability bound as a function of the MTU. As expected, as the MTU increases both the effective load and the schedulability level are reduced. However, the reduction is not strictly monotonic and happens at different rates. In the present case, the system is only schedulable for MTUs in the range [3...3.8].

VII. Optimization algorithms

Depending on the protocol specification, requirements and constraints, the minimum and maximum packet transmission times are restricted to certain limits MTU_{min} and MTU_{max}. Note that $MTU_{max} \leq LSW$ to be able to send at least one packet every EC. This range of MTU will be used by the algorithms that will be explained in the following section.

A. Searching algorithm

To find an optimal MTU that increases the schedulability of messages, we can consider a very simple algorithm based on iterating MTU from MTU_{min} to MTU_{max}. In each iteration, it finds the number of packets for all messages using Equation (5) and then it evaluates $f(nP)$ using Equation (8). Finally, the MTU value that minimizes $f(nP)$ will be the optimal solution. However, the computational complexity of this algorithm can be high. It requires $MTU_{max} - MTU_{min}$ iterations, which may not be suitable for online usage.

To decrease the search space, another algorithm based on the number of packets can be used. Given MTU_{min} and MTU_{max}, the minimum nP_i^{min} and maximum nP_i^{max} number of packets for each message can be specified using Equation (5). Then, for each possible number of packets within this range, MTU is evaluated as explained in Algorithm VII.1.

Algorithm VII.1 shows an algorithm that finds the optimal MTU based on the number of packets. The algorithm iterates for each message m_i (Line 2) and then it goes through all possible number of packets nP_i, from nP_i^{min} to nP_i^{max} (Line 4) and for each iteration it enforces $I_{max} = Mmax_i$ (Line 5). This can be done by assigning the number of packets for all other messages $m_j | i \neq j$ to values that make their packet size less or equal to $Mmax_i$ (Find_nP_All function in Line 6). Find_nP_All function uses Equation (5) to evaluate the number of packets for all messages (considering $MTU = I_{max}$). The algorithm starts by calculating $f(np)$ (Line 7) and then the local optimal MTU that minimizes $f(np)$ based on m_i (Lines 8-11). Finally, considering all local optimal values of MTU, the one that makes the minimum $f(np)$ is the optimal solution (Lines 13-16).

This algorithm iterates $N \times (nP_i^{max} - nP_i^{min})$ times and since nP_i^{max} is proportional to C_i^* (see Equation (5)), the number of iterations depends on the transmission times of the messages. Note that, if the transmission times of messages are not high, then some values of MTU within the

range $[MTU_{min}, MTU_{max}]$ will not be considered (discarded from the search space). The discarded values of MTU are $\forall MTU \neq Mmax_i | \forall m_i \in SM \& \forall nP_i \in [nP_i^{min}, nP_i^{max}]$. All discarded MTUs will not give better results than the considered ones in the algorithm. The reason is that I_{max} is evaluated based on $Mmax_i$ which is in its turn evaluated based on the given nP_i independently of MTU, hence MTU does not affect Equation (8). The only condition to satisfy concerning MTU is $I_{max} \leq MTU$ and we assign $MTU = I_{max}$ to enforce packet transmission time of all messages to be less than I_{max}.

Algorithm VII.1: SEARCHING ALGORITHM()

```
1 f = high value
2 for (m_i = m_1 tom_N)
3    f_i = high value
4   for(nP_i = nP_i^min to nP_i^max)
5     MTU_temp = I_max = ⌈C_i^*/(nP_i)⌉ + O
6     nP = Find_nP_All(I_max, O)
7     f_temp = f_nP(I_max, nP)
8     if f_temp < f_i
9       MTU_local = MTU_temp
10        f_i = f_temp
11     end if
12   end for
13   if f_i < f
14    MTU = MTU_local
15     f = f_i
16   end if
17 end for
18 return MTU
```

B. Simplified algorithm

In the previous section, we have presented two different algorithms that find optimal MTU with relatively high computational complexity which may not be suitable for adaptive systems where the algorithms are applied online. In this section we propose a lower computational complexity algorithm based on an analytical solution that finds a sub-optimal value of MTU.

Equation (8) can be reformulated to be a function of nP_i (the number of packets of one message m_i). Assuming that $Mmax_i$ is the maximum among all messages, i.e., $I_{max} = Mmax_i$, then $I_{max} = \lceil \frac{C_i^*}{nP_i} \rceil + O$ (see Equation (4)). Also, since $MTU = I_{max}$ then $nP_j = \lceil \frac{C_j^*}{I_{max} - O} \rceil$ (see Equation (5)). Substituting these two equations in Equation (8),

$$f(nP_i) = \left(\frac{U^{lub} \times \left(\left\lceil \frac{C_i^*}{nP_i} \right\rceil + O \right)}{E} \right) + \sum_{\forall j} \frac{\lceil C_j^* / (\lceil C_i^* / (nP_i) \rceil) \rceil \times O}{T_j} \quad (9)$$

Equation (9) can be linearized by removing all ceilings from the equation,

$$f(nP_i) = \frac{U^{lub} \times \left(\frac{C_i^*}{nP_i} + O \right)}{E} + \sum_{\forall j} \frac{nP_i \times C_j^* \times O}{T_j \times C_i^*} \quad (10)$$

By assigning the first derivative of $f(nP_i)$ equal to zero, we can find nP_i that minimizes $f(nP_i)$.

$$nP_i = \sqrt{\frac{C_i^* \times U^{lub}}{E \times \sum_{\forall j} \frac{C_j^* \times O}{C_i^* \times T_j}}} \quad (11)$$

Equation (11) can be used in Algorithm VII.1 instead of the inner loop to find the number of packets for each message as shown in Algorithm VII.2. In Algorithm VII.2, the function $f_nPi()$ in Line 3 uses Equation (11) to evaluate the number of packets for m_i.

The complexity of the algorithm is reduced to $O(N)$ which makes it suitable for dynamic systems. However, since we have linearized Equation (10), the result of Equation (11) may not always be optimal but still are near to optimal as will be shown in the next section.

Algorithm VII.2: SIMPLIFIED ALGORITHM()

```
1 f = high value
2 for (m_i = m_1 tom_N)
3   nP_i = f_nPi()
4   I_max = MTU_temp = ⌈C_i^*/(nP_i)⌉ + O
5   nP = Find_nP_All(I_max, O)
6   f_i = f_nPi(I_max, nP)
7   if f_i < f
8     MTU = MTU_temp
9      f = f_i
10   end if
11 end for
12 return nP
```

VIII. EVALUATION STUDIES

In this section, we evaluate the improvements that can be achieved by the algorithms presented in Algorithm VII.1 and Algorithm VII.2 in terms of increasing the schedulability of messages, compared to the case when the maximum protocol's MTU is used (MTU_{max}). The evaluation is performed through different simulation studies as will be explained in the following section.

A. Simulation settings

In each simulation study, the two algorithms are applied on synthetic message sets generated randomly to find the optimal (suboptimal) values of MTU. Message sets are generated based on the following input parameters:

- U_{set} defines a message set utilization and it is equal to $U_{set} = \sum_{\forall j} \frac{C_j^*}{T_j}$.

- $[T_i^{min}, T_i^{max}]$ defines a range of message period.
- N defines the number of messages.

Given each set of input parameters mentioned above, 100000 message sets are randomly generated. In each set, the message set utilization U_{set} is divided randomly among the N messages. Message periods are selected randomly within the range of $[T_i^{min}, T_i^{max}]$. The message transmission time for each message is derived from the desired message utilization. All randomized parameters are generated following uniform distributions.

The following assumptions are made in all simulation studies: the channel transmission rate is 100Mbps, the elementary cycle duration is $E = 1ms$, the protocol overhead is 44 bytes (including Ethernet overhead 38 bytes and FTT-SE overhead) which makes $O = 3\mu$s, and the minimum and maximum possible packet size $[100, 1500]$ bytes, i.e, $[MTU_{min}, MTU_{max}] = [8, 120]\mu$s.

B. Results

We have performed 5 different simulation studies;

- **Study 1** is specified having $N = 10$, $[T_i^{min}, T_i^{max}] = [2 \times E, 50 \times E]$, $LSW = 0.5\times$ E, EDF scheduling and $U_{set} = [0.33, 0.42]$ with 0.05 increment.
- **Study 2** changing LSW (compared to Study 1) to $0.25 \times E$.
- **Study 3** increasing the number of messages for each set (compared to Study 1) to 20 messages.
- **Study 4** changing the range of message period T_i (compared to Study 1) to $50, 100$.
- **Study 5** changing the scheduling algorithm to RM (compared to Study 1).

Table I shows the results of applying the two algorithms (optimal and simplified) on all simulation studies using the maximum U_{sub} that keeps the schedulability of all (100000) message sets. In this table, U_b^{ave}, U_b^{min} and U_b^{max} are the average, minimum and maximum utilization bounds calculated based on the right hand side of Equation (7). MTU in the table shows the range of maximum transmission times that are evaluated for the message sets.

Study/Alg.	U_b^{ave}	U_b^{min}	U_b^{ave}	MTU (μs)
Study1/Opt.	$41,92\%$	$41,60\%$	$41,98\%$	[32.6, 49.8]
Study1/Simp.	$41,90\%$	$41,59\%$	$41,98\%$	[32.9, 49.8]
Study2/Opt.	$19,36\%$	$18,95\%$	$19,43\%$	[24.4, 37.7]
Study2/Simp.	$19,35\%$	$18,93\%$	$19,43\%$	[24.5, 37.7]
Study3/Opt.	$41,73\%$	$41,02\%$	$41,95\%$	[35.2, 51.5]
Study3/Simp.	$41,72\%$	$40,98\%$	$41,94\%$	[33.9, 51.5]
Study4/Opt.	$41,98\%$	$40,97\%$	$41,99\%$	[39.5, 44.1]
Study4/Simp.	$41,98\%$	$40,97\%$	$41,99\%$	[41.7, 44.1]
Study5/Opt.	$28,97\%$	$28,56\%$	$28,98\%$	[33.6, 53.5]
Study5/Simp.	$28,90\%$	$28,52\%$	$28,98\%$	[34.0, 53.5]

TABLE I: Measured results of all studies.

Figure 4 shows the schedulability bounds of **Study 1**, when MTU is evaluated using Algorithm VII.1 (labeled Optimal), Algorithm VII.2 (labeled Simplified) and $MTU = MTU_{max}$ (labeled maxMTU). As shown in Figure 4 and Table I, the results of the two algorithms are very close and the difference between them is very small. The reason is that the error from linearizing Equation (9) can be up to $N \times O$ and the product of this is divided by the messages period, which attenuates the impact of linearizing.

Comparing the results of the algorithms and the use of MTU_{max}, we can see that the algorithms can increase the utilization bound by approximately 5.5, i.e., the utilization bound is improved by 15%. Note that the results of the figure are obtained assuming $LSW = 0.5 \times E$ and because of the effect of I_{max}, the schedulability bound of EDF is reduced to the range shown in the figure.

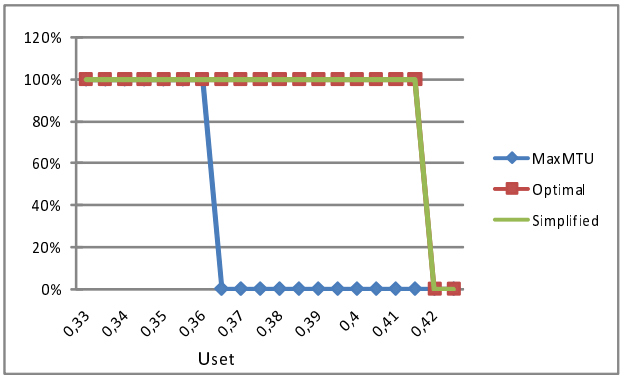

Fig. 4: The results of **Study 1**.

Figure 5 shows the schedulability bound of **Study 2** (decreasing LSW). It is clear that the improvement achieved by using the algorithm is higher when LSW is decreased (the utilization bound is improved by 69%). The reason for this big improvement is that decreasing LSW will make the contribution of I_{max} higher on Equation (7) and only the iterative procedure in the algorithms decrease I_{max}.

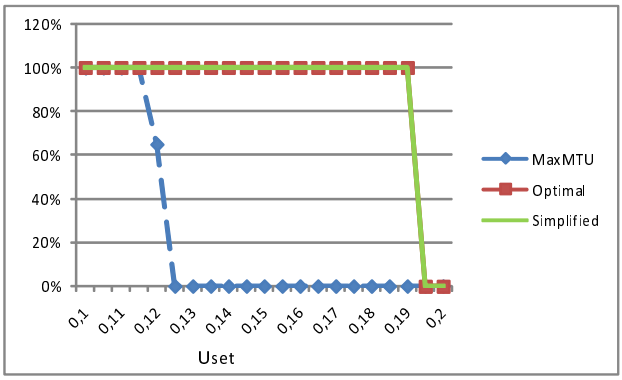

Fig. 5: The results of **Study 2**.

Comparing the results of **Study 3** and **Study 1** in Table I, it is clear that the number of messages does not have a significant effect on the results. When increasing the number of messages, the utilization bound decreases slightly because it may increase the number of packets which increases the protocol overhead effect on Equation (7). However, the additional number of

packets is bounded by the number of tasks.

To measure the effect of message periods, we increased the range of the periods in **Study 4**. As shown in Table I, this can increase the utilization bound but in very small amount compared to the results of **Study 1**. The reason is that since U_{set} is fixed, increasing the periods of generated messages will increase their transmission times, which in its turn increases the number of packets and will have negative effect on the schedulability bound. On the other hand, increasing the message periods will decrease the effect of protocol overhead on Equation (7) which will have a positive effect on the schedulability.

Finally, in **Study 5**, RM is used instead of EDF and the results are shown in Figure 6. The utilization bound is decreased because $U^{lub} = 0.69$, however, the algorithms still achieve a significant improvement compared to the case of using MTU_{max}.

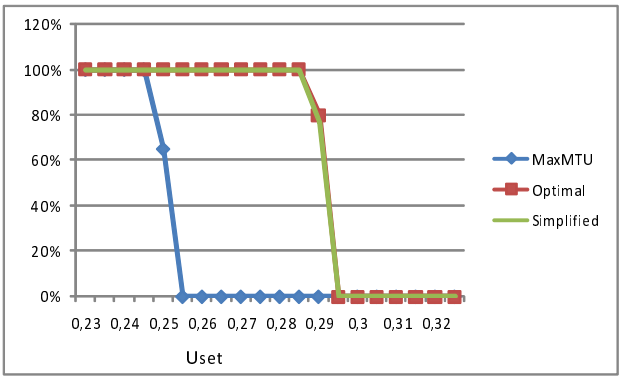

Fig. 6: The results of **Study 5**.

IX. Summary

In this paper, we have studied the effect of MTU size on the schedulability of the FTT-SE protocol. Reducing MTU on one hand reduces the idle time that is imposed to prevent scheduling windows overrun, and on the other hand it increases the number of packets that in its turn increases the protocol overhead. This paper introduced such a tradeoff and presented two algorithms to select the value of MTU. The first algorithm finds an optimal value that maximizes the schedulability of messages but exhibits relatively high computational complexity. The second algorithm finds sub-optimal solutions with lower computational complexity ($O(N)$). We have evaluated the improvements that can be achieved using the algorithms through a comprehensive simulation study. The results of the simulations show that the algorithms can significantly increase the schedulability of the real-time messages compared to the case when maximum MTU is used. In addition, the results show that the performance of the algorithm that finds sub-optimal solutions is as good as the other algorithm.

In this paper, we considered the effect of MTU on the scheduling of downlinks. however, the impact on the scheduling of uplinks is similar. As a future work we would like to include the scheduling of uplinks and downlinks when evaluating the optimal MTU. This work can also be extended to include Server-SE protocol [7] in which real-time messages are scheduled using server based techniques. Optimizing the MTU will have a significant effect on the scheduling of servers, specially for the cases when the servers capacity is small.

References

[1] Ethernet Powerlink - online information. http://www.ethernet-powerlink.org/.

[2] Luís Almeida and José Alberto Fonseca. Analysis of a Simple Model for Non-Preemptive Blocking-Free Scheduling. In *Proc. of the 13th EUROMICRO Conference on Real-Time Systems (ECRTS'01)*, pages 23–3, June 2001.

[3] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito, and M. Caccamo. Preemption points placement for sporadic task sets. In *Proc. of the 22nd Euromicro Conference on Real-Time Systems (ECRTS '10)*, pages 251–260, 2010.

[4] Jianxin Chen, Ling Gong, Yuhang Yang, and Peng Zeng. Average performance of packet network. In *Proc. of the 6th Int. Conference on ITS Telecommunications*, pages 515–518, 2006.

[5] Christopher A. Kent and Jeffrey C. Mogul. Fragmentation considered harmful. In *ACM SIGCOMM*, pages 390–401, 1987.

[6] S.T. Kondoz A.M. Kodikara, C.K. Worrall. Optimal settings of maximum transfer unit (mtu) for efficient wireless video communications. In *Proc. of IEE Communications*, pages 648–654, 2005.

[7] R. Marau, N. Figueiredo, R. Santos, P. Pedreiras, L. Almeida, and T. Nolte. Server-based Real-Time Communications on Switched Ethernet. In *Proc. of Int. Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS'08)*, December 2008.

[8] R. Marau, P. Pedreiras, and L. Almeida. Signaling asynchronous traffic over a Master-Slave Switched Ethernet protocol. In *Proc. on the 6th Int. Workshop on Real Time Networks (RTN'07)*, Pisa, Italy, 2 July 2007.

[9] Ricardo Marau, Luís Almeida, Karthik Lakshmanan, Raj Rajkumar, and Paulo Pedreiras. Utilization-based Schedulability Analysis for Switched Ethernet aiming Dynamic QoS Management. In *Proc of the 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA'10)*, pages 1 – 10, sept. 2010.

[10] Ricardo Marau, Luís Almeida, and Paulo Pedreiras. Enhancing real-time communication over COTS Ethernet switches. In *Proc. of the IEEE Int. Workshop on Factory Communication Systems (WFCS'06)*, pages 295–302, June 2006.

[11] TTTech. TTEthernet. http://www.tttech.com/solutions/ttethernet/, November 2008.