# The Design and Implementation of a Simulator for Switched Ethernet Networks*

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

***Abstract*—In the context of Switched Ethernet, the Flexible Time-Triggered Switched Ethernet protocol (FTT-SE) was proposed to overcome the limitations and related problems of using COTS switches in real-time networks, such as overflow of switch queues due to uncontrolled arrival of packets. Although the FTT-SE protocol has been validated by several experiments on real applications, evaluation of different architectures as well as evaluation of large scale networks is not straightforward. Therefore, a simulator to evaluate different network architectures based on the FTT-SE protocol is useful. In this paper we present such a simulator. We address the extended FTT-SE protocol using multiple switches and we present a modular simulator based on Simulink/Matlab that allows us to visualize message transmissions and to evaluate end-to-end delay bounds of messages.**

## I. Introduction

Recently, there has been a growing interest in using the Switched Ethernet technology even for hard real-time distributed systems as it provides means to improve global throughput compared with other technologies. Moreover, Switched Ethernet also provides traffic isolation and it eliminates the impact of the non-determinism due to CSMA/CD arbitration that the original Ethernet was suffering from. Among several different switch types that have been proposed to support real-time traffic communication, Commercial Off The Shelf (COTS) technologies are becoming more attractive as they reduce the development costs and simplify the maintenance process compared with solutions that use dedicated switches. However, using COTS switches in real-time applications is challenging due to the following limitations; the size of the memory of COTS Ethernet switches is limited, hence it may not be possible to buffer unsynchronized simultaneous traffic from different sources which may cause packet drops that in turn affects the timeliness behavior of the network. In addition, the COTS Ethernet switches have a limited number of priorities to schedule the traffic inside switches.

One solution for the mentioned problem with respect to the COTS switch is to use a master/slave approach where one certain node will be responsible for control over the traffic communication across the network and thereby guaranteeing the real-time requirements of the traffic. In this paper, we focus on the FTT-SE protocol [1] that uses the master/slave approach to enforce global coordination among streams by using a dedicated node called the master, thus controlling the load submitted to the switch at each instant in time and thereby avoiding the potential queue overflow problem. Moreover, the FTT-SE protocol handles different traffic types including real-time periodic, real-time aperiodic and non-real-time messages by defining a specific bandwidth for each type of message. This protocol was investigated and validated by several experimental results in [1]. Moreover, a method is proposed in [2] to deal with the scalability of the protocol, using multiple switches along with multiple master nodes.

In this paper, we focus on the design and implementation of a modular simulator that can be used to simulate different design options of the FTT-SE protocol for both small and large scale networks. This simulator is based on Simulink/Matlab which makes it modular and allows us to create several models according to arbitrary architectures based on the protocol.

The rest of the paper is organized in the following manner. The next section discusses some related work on modeling and simulation of protocols, Section III describes the FTT-SE protocol and the extended solution. Then, Section IV presents the simulator design while Section V validate the simulator using some experiments. Finally, Section VI concludes the paper and presents the future work.

## II. Related work

Several techniques have been proposed to model and simulate the Ethernet protocol using different tools and modeling algorithms. In [3], models are proposed for nodes, switches and traffic according to the Switched Ethernet protocol. Moreover, an evaluation is performed to validate the performance of the modeling method by comparing the simulation results with the collected data from a specific network application.

In the area of embedded avionics networks, a simulation model considering the Avionics Full Duplex Switched Ethernet (AFDX) is proposed in [4]. The end systems (nodes), switch, different queues for switch and end systems, and the measurement unit were modeled. Moreover, the validation of the modeling algorithm is performed for the specific architecture and the performance of that is compared with the results supplemented by Network Calculus. However, the simulator was developed only for a particular application and it is not implemented as a general simulator.

Furthermore, a simulation algorithm was proposed in [5] to evaluate the end-to-end upper bound delay in AFDX networks. Finding an upper bound end-to-end delay for each message

*This work is supported by the Swedish Foundation for Strategic Research, via Mälardalen Real-time Research Center (MRTC) at Mälardalen University.

using simulation requires us to investigate a huge number of possible scenarios. Thus, an approach to reduce the number of possible scenarios was proposed in [5].

In addition, different network simulation systems were designed based on available simulation tools. For instance, a network simulator system for AFDX networks was designed and implemented in [6], in which Network Simulation (NS2) as a tool to simulate TCP, routing over wired and wireless networks, was considered for the main platform, however, NS2 supports limited protocols.

Furthermore, there are many tools which have been developed for network simulation, such as TrueTime [7], OMNET++ [8] and OPNET [9]. TrueTime is a toolbox developed for Simulink/ Matlab. The switched Ethernet protocol as a network block has been supported by the TrueTime toolbox. However, adding new protocols such as FTT-SE need a lot of modifications and changes on the kernel of the tool which is not easy. Moreover, the output results that can be generated from the TrueTime blocks are limited and they need to be modified to allow for calculation of response times of messages. Another tool called OPNET is used to evaluate the performance of a network, specially for evaluation of Internet, however, this tool is a commercial tool. OMNET++ is another component-based and modular simulator which is mainly used for sensor networks, internet protocols and performance modeling. As a result, neither of them can be used directly to include the FTT-SE protocol.

## III. FTT-SE BASICS

The FTT-SE protocol [1] is a real-time communication protocol that combines the master/slave technique with the Flexible Time-Triggered (FTT) paradigm. A dedicated node, called the master node is used to control the traffic in the network by broadcasting a specific message called the Trigger Message (TM). The master node schedules the ready messages according to an on-line scheduling policy, and encodes the scheduled messages into the TM. The scheduling is performed every predefined time interval called an Elementary Cycle (EC), in which the master broadcasts the TM to all slave nodes at the beginning of each EC. Then, the slave nodes receive the TM, encode it and send the scheduled messages for transmission in the current EC.

According to the FTT-SE protocol, the data transmission bandwidth in each EC is divided into two sub-bandwidths (windows) to handle synchronous (periodic) traffic within the Synchronous Window (SW) and asynchronous (aperiodic) traffic, within the Asynchronous Window (AW), as depicted in Figure 1. The time that the slave nodes need to decode the TM is called the turn around time. Moreover, the input and output ports of the switches are called the uplinks and the downlinks respectively.

Furthermore, to handle the asynchronous traffic, each slave node sends a request message, which is called signaling message (SIG), to the master node whenever an asynchronous message is activated. The master node then schedules the asynchronous traffic for upcoming ECs [10]. As illustrated

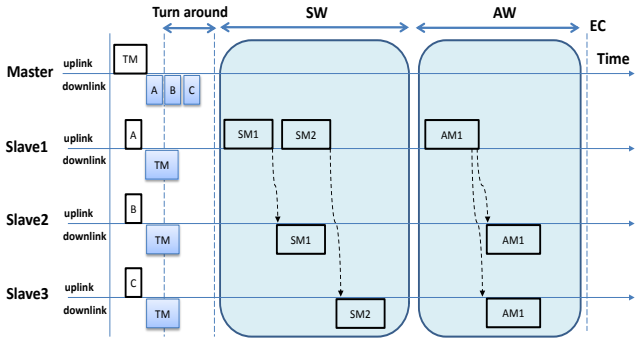


Fig. 1. The FTT-SE Elementary Cycle

in Figure 1, A, B and C are the aperiodic requests from the slave nodes. Moreover, aperiodic messages can be activated at any time during the EC. The worst case scenario occurs when an aperiodic message is activated exactly after a request has been sent to the master node. In such a scenario, the aperiodic signaling request will be sent to the master in the next EC and the master node will schedule that request at earliest in the following EC, i.e. within three ECs.

The scalability of the FTT-SE protocol using multiple switches was investigated in [11] and [2] based on two approaches. In both solutions, multiple switches are connected together directly forming a tree shaped topology.

In the first solution, a single master node is used to coordinate the traffic transmission in the network. The bandwidth assigned for synchronous and asynchronous traffic is similar to the single switch FTT-SE protocol as depicted in Figure 1. Moreover, the TM is generated and broadcasted to all nodes in the network. Also, to deal with asynchronous messages, slave nodes send the request messages to the master node.

In the second approach, a network architecture consisting of multiple switches with a master node connected to each switch is considered. An example of such an architecture is depicted in Figure 2, where the switch SW1, the master node M1 and nodes A and B are grouped into one sub-network (SN1). The sub-network SN1 is a parent sub-network for SN2 and SN3. Moreover, a cluster is defined such that it contains all sub-networks with the same parent sub-network. For instance, in Figure 2, SN4 and SN5 are grouped as one cluster for which SN2 is the parent sub-network.

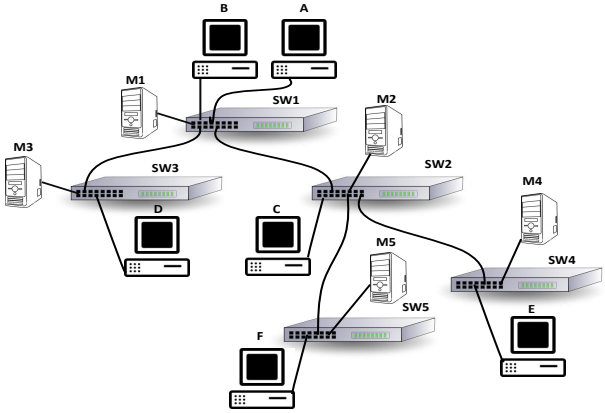


Fig. 2. An example of network

To handle the traffic in such a network, two categories of traffic are defined as local and global. The traffic which is transmitted between the nodes inside one sub-network is called local, otherwise the traffic is called global. To handle the synchronous and asynchronous traffic, the data transmission within an EC is divided among the traffic types as depicted in Figure 3, where SW is the synchronous window and AW is the asynchronous window. The local synchronous and asynchronous traffic handling is carried out similar to the single switch FTT-SE protocol within their specified bandwidth. However, all master nodes schedule all global messages in the network in parallel, based on the allocated bandwidth for such types of traffic. To schedule the global asynchronous messages, the global asynchronous window is divided per cluster and the parent master node of the cluster is responsible to schedule the global aperiodic messages inside that cluster.

Furthermore, the ECs of all master nodes are time synchronized using a particular message which is called the Global Trigger Message (GTM). The root master sends the GTM to all master nodes and they will wait to receive this message before broadcasting their local TM.

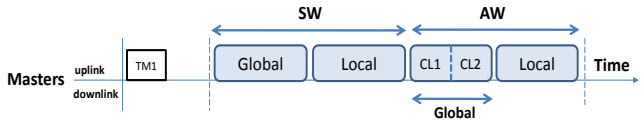


Fig. 3. EC considering local and global traffic

## IV. Simulator Design

Using Simulink/Matlab we have developed a simulator to evaluate the timing behavior of the messages in a network based on the FTT-SE protocol for small and large scale networks. We have used Simulink/Matlab due to its modular features along with custom blocks and graphical interfaces. The core of the simulator is a cycle-based which starts by sending TM to the slaves, i.e. the simulator is designed in several states such that each state is allowed to execute only after finishing of the previous state. However, the master block keeps track of the EC duration as well. We have developed three basic models using the S-Function block in Simulink to simulate the functionality of the master node, slave nodes and switch models respectively. These blocks are stored in a Simulink library file. To simulate the parallel execution of blocks in a model, we have divided each EC into 100 time slots and in each of them all the functions are executed. The number of time slots shows the resolution of simulation which is not fixed and can be changed in the configuration of the simulator. However, by increasing the number of time slots, the simulation time will increase due to the number of functions that need to be execute in each time slot. On the other hand, decreasing the time slots number can affect on the accuracy of the results. Therefore, we have chosen 100 time slots in this trade-off as an engineering experiment. Note that, the function of blocks in Simulink executes in sequential order which is automatically specified by Matlab in advance. Therefore, the input data of each block is guaranteed to be available before the execution of that block.

### A. Ready Queues Management

For the case of multiple master nodes, each master contains four ready queues to support local and global periodic as well as local and global aperiodic messages. Scheduling of global messages is performed in parallel in all master nodes at the same time. The ready queues are sorted based on the priority of messages in which the highest priority messages are inserted at the head of the queues, we used the Fixed Priority/Highest Priority First Scheduling Policy for on-line scheduling in the simulator. Messages with the same priority are sorted in the queues based on the First Come First Serve (FCFS) policy for local messages, whereas for global messages, messages having the same priority are sorted based on the id number of messages.

For management of the ready queues, we have developed three functions to handle queue updating before scheduling the messages by master nodes. These functions, which are implemented in Matlab m-files, are the following:

**Get head message.** This function returns the first message in the ready queue which is always the highest priority message among all messages in the ready queue in this simulation.

**Remove a message.** If the scheduler checks a message and it selects that message to be transmitted in the current EC, the message should be removed from the ready queue. Therefore, this function removes a message defined by its id together with the ready queue in which the message is residing. The output of this function is the updated ready queue sorted according to the priorities of all messages in the queue.

**Insert a message and sort in ascending format.** Whenever a message becomes ready, it should be inserted in the correct ready queue, which is performed by this function. This function inserts a message in a queue according to its priority and it re-sorts the queue according to the priorities of messages in which the highest priority message is assigned at the head of the queue.

For the single master case, two queues are used to schedule both synchronous and asynchronous messages, one specific for each type of messages. Therefore, the queue management functions are applicable for these two ready queues.

### B. Master Block Design

The master block is divided into two sub-blocks dealing with sender and receiver functions. We have defined an array structure for the master node to store its variables and parameters. In the solution with multiple master nodes, each master node in the network may have different parameters depending on its local configuration such as local message numbers and local slave node properties. All master blocks in the model are connected to a single m-file function which is distinguished with a mask block parameter number.

For each master input (receiver) and output (sender) blocks two separate functions are implemented, however the master function is developed based on a state flow that each of them should run in order. The master function has three states which are depicted in Figure 4. Each state, is allowed to run when the previous state has executed only. The first state is broadcasting

the TM to all slaves. The next state is receiving aperiodic requests from the slave nodes during the TM window. The last state is performing the scheduling function for all kind of messages, including local and global aperiodic messages and generating a TM for the upcoming EC. State 1 and 3 are executed in the master sender function, whereas state 2 is executed in the receiver function. For input and output signals, we implemented separate scope output functions due to the flexibility of the graphical interface.

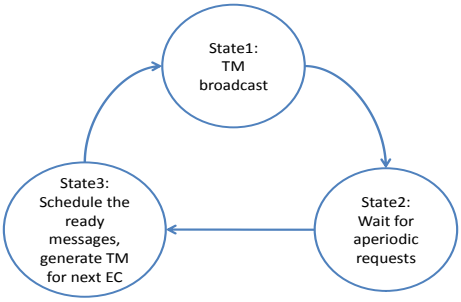


Fig. 4. Master function state flow

In state 2 of the master function, which is executed in the master receiver sub-block, the master polls the aperiodic requests from the slave nodes. This state finishes and moves to the next state when all signal messages are received. The aperiodic requests which are received during this state are from local slave nodes or from children slaves for global aperiodic requests. In both cases, the requests are stored and the master schedules them for the next EC. The local aperiodic scheduled messages are encoded inside the TM along with other periodic messages, however the global scheduled aperiodic messages are encoded in a different trigger message called the asynchTM, which is sent to children slave nodes directly.

In state 3, the master function looks up the message structure and checks whether any periodic message becomes ready. The scheduler inserts all ready messages, including aperiodic messages, which are requested from the slaves into the related ready queues. Moreover, the ready time of each message is stored in the related variable of message. Consequently, scheduler function checks bandwidth for the ready message transmission according to their destination and rout. The scheduled messages are encoded inside the TM to be sent at the beginning of the next EC.

To support the single master FTT-SE network, the master block is developed such that it schedules the traffic and broadcasts the TM to the entire network. Moreover, each slave node sends the aperiodic request to the master node which is connected to the top switch.

### C. Switch Model Design

Similar to the master block, the switch is modeled with a Matlab S-Function associated using a particular m-file. Four kinds of connections are defined for the switch models i) the master connection identified in port 1, ii) the parent switch connection is dedicated to port 2, iii) two children connections for the children sub-networks as port 3 and 4, and finally iv) five connections for the slave nodes. Therefore, 9 ports are assumed for switches in this version of the simulator. Moreover, for each input and output link a specific buffer is assigned to store receiving and sending data. The buffers are sorted according to the First In First Out (FIFO) policy. The general structure of the switch model is to poll the input data and to process the destination address of them, and in turn, to insert into the related output buffer. The switch parameters including the input and the output buffers are stored in an array structure which is shown in Table I.

| switch structure | description |
|---|---|
| port_nbr | The number of ports for switches. |
| inBuffer | The input buffer for all input ports individually. |
| outBuffer | The output buffer for all output ports individually. |

TABLE I
THE SWITCH STRUCTURE

### D. Slave Block Design

Similar to the master block, the slave block is divided into two sub-blocks denoted sender and receiver sub-blocks. The slave function is executed based on the state flow which indicates the current state of each slave node. The slave function composes of four individual states started by the TM reception. The state flow of the slave function is depicted in Figure 5.

After receiving the asynchTM and TM messages from the parent master and the main master respectively, the slave checks if any local aperiodic messages are ready to be transmitted. The same check is performed for the global aperiodic messages. For aperiodic messages, the sporadic model is used to model this type of traffic in which the minimum inter-arrival time is defined for each message, however in this simulator a dynamic activation for aperiodic messages is developed to simulate the unpredictable arrival time of aperiodic messages. Moreover, the aperiodic message may become ready at anytime during the EC window. To simulate this behavior of aperiodic messages, we assume the worst case in which the message always becomes ready after the TM broadcasting window.

The third state of the slave function, which is executed in the sender block, is decoding the TM and transmitting the messages which are scheduled by master node. This state includes all local/global periodic and aperiodic messages. However, the message transmission starts with local and global periodic messages and continues with local and global aperiodic messages.

After sending the scheduled messages, the last state is to wait for message receiving. The slaves read their inputs until the EC time window is finished. When a message is received from the slave node, the receiving time is stored in the related message variable. The time interval between the ready time and the receiving time of the messages shows the end-to-end transmission time. For setting the receiving time, the store-and-forward switch delay and order of messages are considered to simulate as accurately as possible. Since the bandwidth capacity was checked in the scheduler, then all scheduled

messages should be received in the current EC without any deadlines being missed. In case of a deadline miss or a failure in receiving of message, the output report of the simulator will show it.

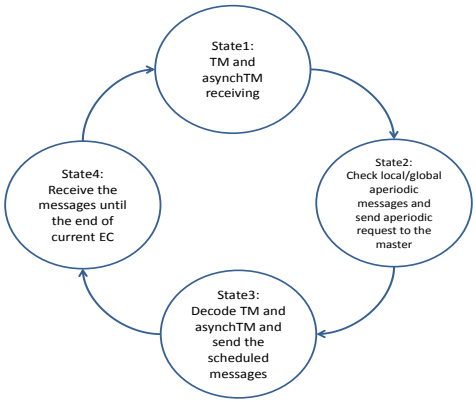


Fig. 5. Slave function state flow

For generating the scope output signals to cover both receiving and sending messages, the scope functions store the messages which are sent and received. The messages are scoped according to their transmitting time. Each message is indicated by its identification number which is unique in the entire network.

### *E. Settings and Report*

In order to set the configuration of network example, such as the EC size and the bandwidth allocations, a database which is developed in Matlab m-file is prepared. For each network model, different configuration can be determined to assess the performance of the example in different bandwidth assignments.

Moreover, to present the end-to-end delay of messages, after finishing the simulation, a function is developed to generate the output reports. These reports provide information including the network parameters such as number of switches and masters, EC configuration and end-to-end delays including minimum, average and maximum that has been measured during the simulation.

## V. Examples

In this section, we present an example of a network that consists of 5 switches (sub-networks) as depicted in the Simulink model in Figure 6. This network composes of 16 slave nodes in which 2-4 nodes are connected to each sub-network. The network parameters for this example are $EC = 4ms$, $TM = 12\mu s$, $SIG = 6\mu s$ and the transmission speed of the Ethernet network is considered as $100Mbps$ and 40 messages are generated randomly. The Fixed Priority Scheduling Policy is assumed in this example and the priority of messages is selected according to the Rate Monotonic priority assignment. Note that, the simulator can support higher amount of messages if all messages are schedulable (meet their deadlines) in the network architecture. In this example we have experimented 40 messages to present the results considering the space limit.

### *A. Multiple Masters Network*

In the multiple masters architecture, each switch is connected to a single master, i.e., five master nodes are created in Simulink, for illustration purpose we explain the root sub-network model in Figure 7. Moreover, the transmission bandwidth in each EC is divided as follows. The synchronous local and global scheduling windows are selected to have $1ms$ equally, the asynchronous local scheduling window is $800\mu s$ and finally the asynchronous global scheduling window is $700\mu s$. In the example, the network is composed of two clusters and the bandwidth of the asynchronous global scheduling window is further divided equally among them, i.e., $350\mu s$.

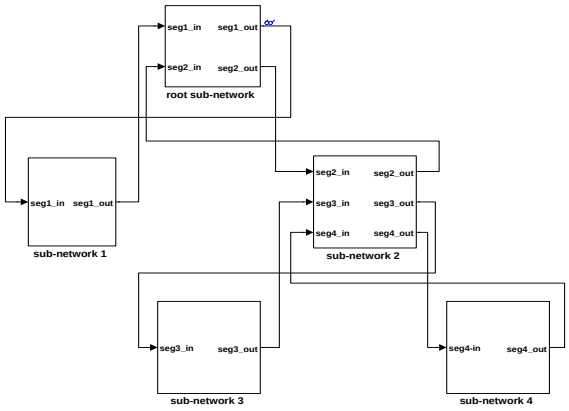


Fig. 6. Evaluation example

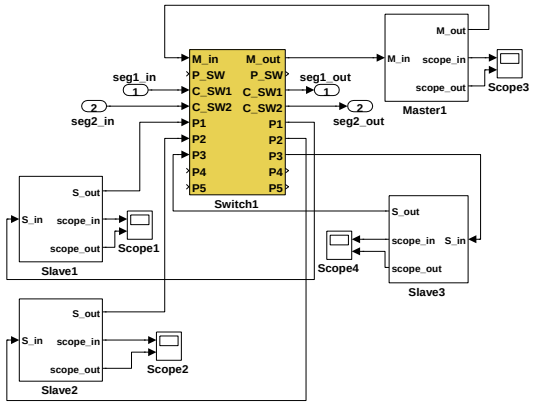


Fig. 7. Root sub-network model

The simulation for this example is performed for time duration of 500 ECs of simulation time. To visualize the message transmission, an ordinary Scope block of Simulink is attached to the respective scope ports of the master and slave blocks. For instance, the messages transmitted and received by slave node 6 in the root sub-network are illustrated in Figure 8, where the x-axis presents time and y-axis shows the message id. In this example the messages $m21$, $m23$ and $m33$, which are transmitted from slave node 6, are periodic with the priority equal to 4, 4 and 1 respectively (higher numbers represent higher priority). Also, this slave node receives the messages $m20$ and $m22$ which are periodic as well. The

| message id | Min. RT | Avg. RT | Max. RT |
|---|---|---|---|
| 20 | 2 | 2 | 2 |
| 21 | 2 | 2 | 2 |
| 22 | 2 | 2 | 2 |
| 23 | 2 | 2 | 2 |
| 33 | 2 | 2 | 2 |

TABLE II
MESSAGE RESPONSE TIMES

transmission time of each message is depicted in the scope with respect to the declaration of the messages. Moreover, the end-to-end delay of all messages in the model can be reported after the simulation time is finished. The minimum, average and maximum response time, which are measured during the simulation, for the mentioned messages are shown in Table II (the unit used is multiples of ECs).

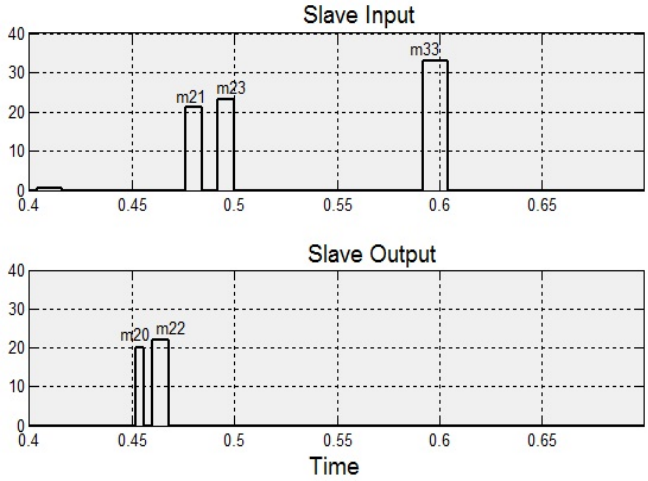


Fig. 8. Scope input/output of slave number 6 in example 1

### B. Single Master Network

In this section, we present the results of applying the approach of a single master on the example presented in the previous section. The master node is connected to the Switch 1. The synchronous window and the asynchronous window are selected to have $2ms$ and $1.5ms$ respectively. Similar to the previous example, the simulator is executed for 500 ECs of simulation time. Figure 9 shows the message transmissions in the slave number 7.

## VI. Conclusion and Future Work

In this paper, we addressed the FTT-SE protocol and the extended FTT-SE protocol to support large scale networks using multiple switches along with multiple master nodes. Moreover, we presented the design of a simulator based on Simulink/Matlab. The simulator contains three basic models representing master, switch and slaves, all implemented as S-Function blocks in Simulink to make the simulator modular.

Moreover, two examples consisting of five sub-networks are created in the simulator to evaluate both the FTT-SE approaches. Different output scopes of message transmissions are presented along with end-to-end delay reports using both a single master as well as multiple masters. We have presented this tool that allows us to perform detailed analysis of the protocols in a way that before needed implementation with

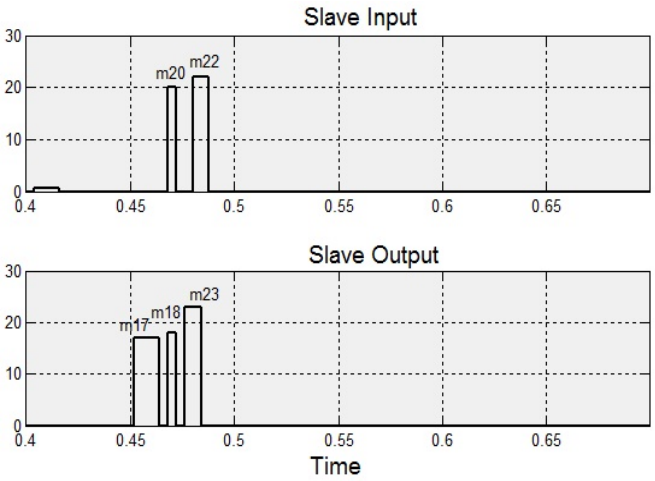


Fig. 9. Scope input/output of slave number 7 in example 2

all its complexity. This version of the simulator is designed and implemented considering some restriction assumptions, e.g. the switch model is limited to support two children sub-networks. However, the extension of the simulator is currently ongoing.

Furthermore, this tool is extensible in the sense that we can easily accommodate other Ethernet protocols for comparison, since the core of the simulator is already implemented. Moreover, we are planning on making the tool available for public as a downloadable plug in to Simulink/Matlab.

## References


[1] 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.

[2] M. Ashjaei, M. Behnam, T. Nolte, L. Almeida, and R. Marau, "A compact approach to clustered master-slave ethernet networks," *9th IEEE International Workshop on Factory Communication Systems (WFCS'12)*, May 2012.

[3] Z. Huang, Y. Zhang, and H. Xiong, "Modeling and simulation of switched ethernet," in *2nd International Conference on Computer Modeling and Simulation (ICCMS'10)*, vol. 3, January 2010.

[4] H. Charara and C. Fraboul, "Modeling and simulation of an avionics full duplex switched ethernet," in *Advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop*, July 2005.

[5] J.-L. Scharbarg and C. Fraboul, "Simulation for end-to-end delays distribution on a switched ethernet," in *12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07)*, September 2007.

[6] S. Dong, Z. Xingxing, D. Lina, and H. Qiong, "The design and implementation of the afdx network simulation system," in *International Conference on Multimedia Technology (ICMT'10)*, October 2010.

[7] D. Henriksson, A. Cervin, and K.-E. Årzén, "TrueTime: Real-time control system simulation with MATLAB/Simulink," in *Proceedings of the Nordic MATLAB Conference*, Copenhagen, Denmark, October 2003.

[8] "Omnet++: Component-based c++ simulation library, available at http://www.omnetpp.org."

[9] "Opnet: Application and network performance, available at http://www.opnet.com."

[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] R. Marau, M. Behnam, Z. Iqbal, P. Silva, L. Almeida, and P. Portugal, "Controlling multi-switch networks for prompt reconfiguration," in *Proc. of 9th Int. Workshop on Factory Communication Systems (WFCS'12)*, May 2012.