A Compact Approach to Clustered Master-Slave Ethernet Networks

Mohammad Ashjaei, Moris Behnam, Thomas Nolte
Mälardalen University, Västerås, Sweden
mohammad.ashjaei@mdh.se

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

Abstract

Ethernet switches are increasingly used in real-time distributed systems as a technical solution to guarantee the timeliness in communications. However, there are still limitations related to real-time behavior caused by the limited number of priority levels and the possibility of memory overruns with consequent message losses. These limitations can be eliminated using a master/slave technique such as proposed by the FTT paradigm. This led to the FTT-SE protocol that schedules transmissions centrally in a master node. While this protocol has already been well studied and investigated for small networks with a single switch, its extension to larger networks is still an open issue. In this paper we propose a compact clustered solution to scale the FTT-SE protocol to networks of multiple switches by organizing the network in sub-networks composed of one master and one switch each and which can be connected directly, without bridges. This paper also shows how the timeliness of the traffic can still be enforced. The response time analysis, implementing a prototype and developing a simulator is currently on-going.

1 Introduction

Communication within embedded systems is substantially increased when the systems are distributed, large scale and complex. To support a higher number of nodes and functionality, several protocols have been proposed to achieve real-time behavior while exchanging higher amounts of data. One possible technique based on switched Ethernet is using a master/slave scheme in which a master node controls all transactions between the slave nodes. Also, the master node is responsible to schedule all communication considering the real-time and QoS constraints. This technique is used, for example, in the EtherCAT [1] and the Ethernet Powerlink (EPL) [2], where a master node dispatches transmissions according to scheduling tables.

The FTT-SE protocol (Flexible Time Triggered Switched Ethernet) [3] is a master-slave technique that supports run-time adaptation of the network traffic enforcing an explicit synchronization. Several methods have also been investigated to handle asynchronous traffic in the FTT-SE protocol [4]. However, until recently this protocol considered single switch networks, only. Different alternatives to handle the limitation of a single switch have been suggested in [5]. Furthermore, scheduling and response time analysis of a multi-hop FTT-SE network configuration using a single master with multiple switches has been studied in [6]. Nevertheless, a solution based on a single master is suitable for small and medium size networks, only, and contains a single point of failure.

In this paper, we propose a solution to support large scale FTT-SE networks with multiple switches, by defining sub-networks (clusters) composed of one switch and one master each, and connecting them together directly, as needed. We show how this can be achieved keeping the timeliness of the communications. The proposed solution covers both synchronous and asynchronous message transmissions.

2 FTT-SE Protocol

FTT-SE is a real-time communication protocol that implements the flexible time triggered paradigm to guarantee the traffic timeliness. A dedicated node called master controls transmissions in the network at every Elementary Cycle (EC). For each EC, the master node schedules all ready messages according to an on-line policy, such as fixed priority, and encodes the schedule into a Trigger Message (TM). The master node broadcasts the TM to all slave nodes at the beginning of the EC [3], slave nodes receive the TM, decode it and transmit the messages that were scheduled for transmission. During transmission of the TM by the master, slave nodes send the status of their ready asynchronous traffic to the master. As illustrated in Figure 1, the time it takes for a slave node to decode the TM and start its transmissions of messages is called the turn around time. Moreover, scheduler considers an idle time in transmission bandwidth to prevent overrun of messages. Figure 1 shows the communications within each EC assuming a simple example of a network consisting of one master node and two slave nodes (A and B). As shown in the figure, the data communication slot within each EC is divided in two sub-slots, where one dedicated to synchronous traffic (periodic) called the synchronous window (SW), and the other to asynchronous (aperiodic) traffic (ASW). The input and output ports of switches are called uplinks and downlinks respectively.

The master node considers the asynchronous requests from the slaves (for example A and B for two slaves in Figure 1) and schedules ready aperiodic messages for the upcoming EC [4]. In the best case, this method takes at least one EC to schedule the aperiodic traffic [4] (one or two additional ECs are required to send the request to the master).

Furthermore, the FTT-SE protocol fragments large messages into several packets which are scheduled separately, and selecting the optimum packet size has been studied in [7] to maximize the schedulability of traffic.

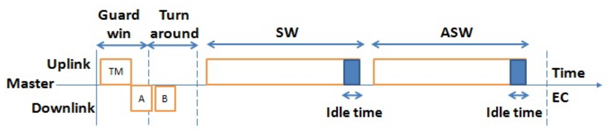

Figure 1. FTT-SE Elementary Cycle

3 System Model

In this paper we assume that the network is composed of several switches connected in a tree topology in which one switch on the top of the hierarchy is connected to one or more switches in the second level of hierarchy. This may continue to deeper levels in the hierarchical architecture. Moreover, each switch has one master node attached, together with several nodes, forming a sub-network. Also, each switch, master and sub-network of the network hierarchy is a parent switch, master and sub-networks for the lower hierarchy level connected to it. The level on the top of the network hierarchy is called the root level. Figure 2 shows an example of the mentioned architecture in which five sub-networks are connected together. Sub-network 1 contains SW1, master M1 and nodes A and B. SW1 is the parent switch for SW2 and SW3 and, in turn, SW2 is the parent switch for SW4 and SW5. Furthermore, the switches are considered as store-and-forward type.

Synchronous message streams are modeled using the periodic real-time model $m_i(C_i, Mmax, D_i, T_i, O_i, S_i, Ds_i, R_i)$, where C_i is the total transmission time of the message, D_i and T_i are relative deadline and period of the message, respectively, O_i is the offset, $Mmax$ is the maximum packet transmission time for the large messages that are fragmented by the protocol. Also, R_i denotes the set of switches in the route of a message from the source node S_i to the destination node Ds_i. For asynchronous message streams, the model is similar to the periodic model except that there is no offset O_i, and T_i is the minimum inter-arrival time of the message [6].

Moreover, when using multiple switches in a network, we define two kinds of messages. If the sender and receiver of a message are connected to the same switch, i.e., they belong to the same sub-network, the message is called local. Otherwise, a message is called global. Finally, note that we are considering a restricted model with unicast messages, only. The extension to multicast messages will be done in future work. This extension is more critical to the analysis than to the operational aspects, which raise no specific difficulty. Substantial experience with this issue has already been developed in the current FTT-SE implementation [3].

4 Problem Formulation

Simply connecting the sub-networks as defined previously is not enough to achieve an efficient bandwidth utilization and an analyzable network. This is due to following reasons:

Confinement of broadcast domains. Within each sub-network, the respective master controls the traffic using broadcasting one TM per EC. By simply connecting sub-networks together, the broadcast nature of each TM will make them propagate through the entire network, generating unwanted interference.

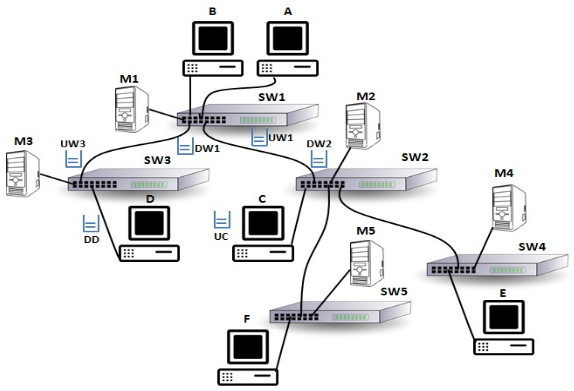

Figure 2. An example of network with five switches

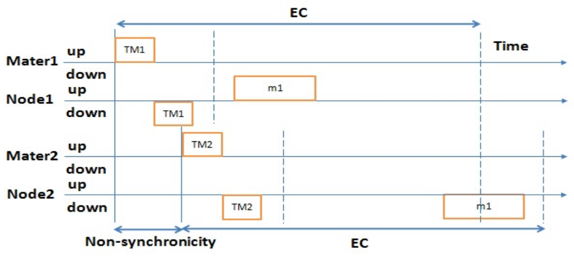

Figure 3. Overrun caused by EC non synchronicity

Time synchronization. According to FTT-SE, all transmissions between nodes scheduled in each EC should be finished within the same EC. This should also apply to the traffic that travels across sub-networks, which requires that the ECs in all sub-networks are synchronized. Therefore, all masters should broadcast their TMs approximately at the same absolute time. Otherwise, transmissions can overrun the EC causing undesired interference (Figure 3).

Scheduling synchronization. All masters that schedule global messages should do it consistently, i.e., they should be scheduled in the same ECs, in order to limit the interference between global messages sharing communication paths. Otherwise, global messages scheduled in one sub-network but not in another one may suffer from unwanted interference and miss their deadlines (Figure 4).

5 Confinement of broadcast domains

In order to prevent TMs to cross the sub-network boundaries we propose using multicast groups. Each sub-network has, thus, its own multicast address that includes all the local nodes and excludes the connections to other sub-networks. Thus, sending a TM to such an address naturally confines its distribution to the associated local nodes. Nevertheless, the nodes transmissions are carried out in the usual way, with direct addressing. Another alternative would be to use one VLAN per cluster but we believe this would be more complex to implement due to the number of situations of traffic that needs to cross the VLANs boundaries, i.e., the global traffic. A deeper comparison of both approaches will be carried out in future work.

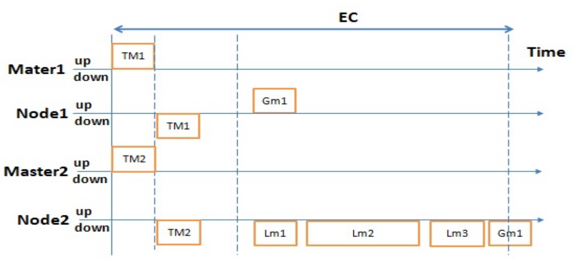

Figure 4. Overrun caused by message interference

6 Time Synchronization

Among the diverse ways of synchronizing the ECs in all sub-networks we propose using a simple explicit synchronization mechanism based on a specific message that is broadcast to the entire network by the root master. We call this message the Global Trigger Message (GTM) and it is a minimum size Ethernet packet. All masters in all sub-networks are waiting for the GTM to initiate their local ECs by sending their own local TM (Figure 5). This mechanism generates a staggered start of the ECs in sub-networks at different depths in the hierarchy but we consider this delay acceptable given the minimum length of the GTM.

In order to recover from GTM losses, we define time-out intervals for each master according to its position in the network hierarchy. If a master does not receive a GTM after its time-out interval, it will generate one and sent to its children. Nevertheless, other fault-tolerance mechanisms related to tree reconfiguration have implications on the timeliness of the traffic and will be addressed in future work.

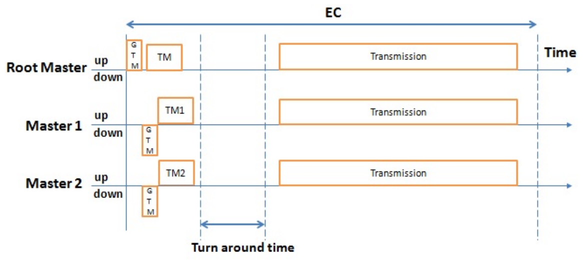

Figure 5. Global synchronization using a Global Trigger Message

7 Global Scheduling

7.1 Periodic Message Scheduling

According to the FTT-SE protocol, each EC has two scheduling windows (SW and ASW) to handle synchronous and asynchronous traffic in the network. Our goal is to schedule the local and global traffic within those windows efficiently.

For this purpose we have divided each scheduling window in global and local sub-windows to separate local transmissions from global ones. For each asynchronous and synchronous window, specific bandwidth for local and global messages is considered as shown in Figure 6. By having a dedicated bandwidth for local message transmissions, the same message scheduling approach used for a single switch network in [4] can be reused for the local message scheduling. The optimal selection of bandwidth reservation for the local and global messages can be studied as an optimization problem.

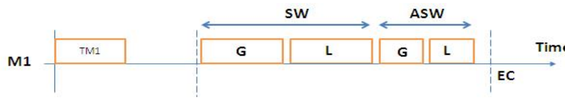

Figure 6. EC windows considering local and global traffic

As for synchronous global messages, we schedule them in parallel in all masters, with similar parameters. For this purpose, we make sure that all synchronous global schedulers are synchronized, i.e., scheduling the global synchronous messages in the same ECs in all sub-networks. The synchronization of the schedules is facilitated with a cyclic EC counter that is broadcasted to the whole network within the GTM. Since each global synchronous message is scheduled in all sub-networks, the schedulers ensure that there is enough bandwidth for its timely transmission along its path. However, note that it will be effectively triggered in one of the sub-networks, only, in which the respective producer is.

Looking at the network shown in Figure 2, consider that there is a synchronous global message that must be sent from C to D. It must cross the following links UC, DW2=UW1, DW1=UW3 and DD in which UC is the uplink connected to node C and DD is the downlink connected to node D and DW2 and UW3 are the downlink of SW2 and uplink of SW3, respectively. To make sure that all links in the path have adequate bandwidth reserved for the message transmission in a given EC, all masters involved directly and indirectly in the path must schedule it consistently, namely masters M2, M1 and M3. Nevertheless, to ensure about the consistency of the schedulers, all masters schedule the message within the global synchronous window.

7.2 Aperiodic Message Scheduling

Similarly to the synchronous window for handling periodic messages, we have also divided the asynchronous window into two parts for local and global aperiodic messages, respectively. Therefore, local aperiodic messages have their own bandwidth for transmission. Thus, they can be handled exactly as in the case of single switch networks presented in [4].

Concerning global asynchronous messages, the solution proposed for global synchronous scheduling does not work. In fact, unlike synchronous messages, the time at which asynchronous messages become ready to be transmitted is not known in advance. Therefore, we propose two possible solutions to perform the global asynchronous message scheduling such that it is analyzable and support timeliness guarantees.

7.2.1 Bandwidth reservation per sub-network

In this solution we further divide the bandwidth allocated to the global asynchronous sub-window among all sub-networks, i.e., we allocate a dedicated bandwidth to each sub-network. Considering the network architecture in Figure 2 with 5 sub-networks, we divide the global asynchronous sub-window in 5 sub-windows such that each sub-network has its dedicated sub-window for scheduling the global aperiodic messages that are requested by its slave nodes. Since there is an exclusive window for each sub-network taken into account by all masters, there will be no interference between the global aperiodic messages produced in different sub-networks. Moreover, this method is again similar to the single switch FTT-SE case. A similar approach was also proposed in [8] but with bandwidth being reserved for each node instead. The main disadvantages regarding this method are the following:

- Similarly to the single switch case, idle time must be considered in both analysis and implementation of the protocol for each sub-window dedicated to each sub-network to avoid overrun of aperiodic messages. Therefore, the accumulation of all idle times decreases the bandwidth utilization and efficiency.
- Whenever there are no global asynchronous requests in one EC in a given sub-network, the respective bandwidth will be wasted and cannot be used by other sub-networks, further reducing the protocol bandwidth efficiency.

7.2.2 Bandwidth reservation per sn-cluster

In this case we reorganize the network in clusters of sub-networks, which we call sn-clusters. Basically, a sn-cluster merges all sub-networks that have the same parent sub-network. Using again the example in Figure 2, SW4 and SW5 are grouped as one sn-cluster while SW2 and SW3 are grouped as another sn-cluster together with the root sub-network SW1. The global asynchronous bandwidth is now divided among the sn-clusters, only, consequently decreasing the number of global asynchronous sub-windows (Figure 7) and the total idle time.

The rationale of this solution is to let each parent master manage in an integrated way the global asynchronous requests of all its children sub-networks. In this case, the nodes send their global aperiodic message requests to the parent master in a specific signaling message, sent immediately after the signaling message of the local requests. The parent master schedules those requests together considering the sn-cluster assigned bandwidth and prepares a special TM, called asyncTM, just with the triggers for such messages, to be broadcast in the children sub-networks, after their regular local TM. This is achieved sending the asyncTM to each child sub-network multicast address.

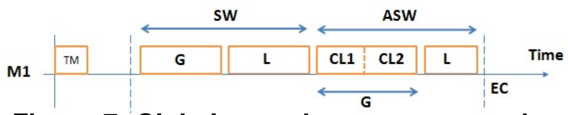

Figure 7. Global asynchronous reservation per cluster

Considering again Figure 2, if there is a ready global aperiodic message from node E, then node E sends the request to master M2 (parent master). Then, M2 will schedule this request together with any other requests coming from its children sub-networks, using the sn-cluster assigned bandwidth (sub-window), and send an asyncTM with the respective triggers to sub-networks SW4 and SW5. The advantages and drawbacks of this solution are:

- There is less idle time associated to the global asynchronous sub-windows and thus any bandwidth slack can be used more efficiently than in the first solution.
- It involves extra complexity, not only in forming the sn-clusters but also in assigning the respective bandwidth and in the extra signaling and the trigger messages that are needed.

Finally, note that the bandwidth distribution among sn-clusters is another optimization problem that needs further investigation.

8 Conclusions and Future Work

In this paper we propose a solution to scale the FTT-SE protocol to large networks using a compact clustered approach in which the clusters are sub-networks composed of one master, one switch and the nodes connected to it. The sub-networks are connected directly, thus in a compact way, without any specific interconnection equipment such as gateways. We addressed the design issues to make this an efficient solution that provides timely communications, covering both synchronous and asynchronous traffic.

Currently, we are refining the design of the current solution as well as implementing a prototype and developing a modular simulator based on Matlab/Simulink. Simultaneously, we are developing a response time analysis suitable for the protocol presented in this paper. Once implementation and simulator are available, we will validate the proposed approach and we will also apply it to general master-slave networks.

References

[1] "Real-time PROFINET IRT, http://us.profibus.com /profinet/07".

[2] "Ethernet Powerlink, available at http://www.ethernet -powerlink.org".

[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, pp. 295 –302.

[4] 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.

[5] F. Yekeh, M. Pordel, L. Almeida, M. Behnam, and P. Portugal, "Exploring alternatives to scale FTT-SE to large networks", in *6th IEEE International Symposium on Industrial Embedded Systems (SIES'11)*, june 2011, pp. 107 –110.

[6] M. Behnam, Z. Iqbal, P. Silva, R. Marau, L. Almeida, and P. Portugal, "Engineering and Analyzing Multi-Switch Networks with Single Point of Control", in *International Workshop on Worst-case Traversal Time (WCTT'11)*, November 2011.

[7] 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)*, september 2011, pp. 1 –7.

[8] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance Analysis of a Master/Slave Switched Ethernet for Military Embedded Applications", *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 534 –547, november 2010.