Switched Real-Time Ethernet in Industrial Applications - Asymmetric Deadline Partitioning Scheme

Hoai Hoang and Magnus Jonsson

School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden, Box 823, S-301 18, Sweden. {Hoai.Hoang, Magnus.Jonsson}@ide.hh.se, http://www.hh.se/ide

Abstract

This paper presents work on a switched Ethernet network extended to allow for periodic real-time traffic, using earliest deadline first (EDF) scheduling. A scheme of asymmetrically dividing deadlines of real-time channels between the in and outgoing links to/from the switch is proposed (ADPS). The result of the simulations of setting up RT channels over a full-duplex switched Ethernet network is presented. The simulations show that the ADPS performs well when master-slave communication is assumed over the network.

1 Introduction

An important trend in the networking community is to involve more switches in the networks (e.g., LAN, Local Area Networks) and a pure switched-based network becomes more and more common. At the same time, the industrial communication community has a strong will to adapt LAN technology (e.g. Ethernet) for use in industrial systems. The involvement of switches does not only increase the performance; the possibility to offer real-time services is also improved. Now, when the cost of LAN switches has reached the level where pure switched-based networks have become affordable, the collision possibility in IEEE 802.3 (Ethernet) networks can be eliminated and methods to support real-time services can be implemented in the switches without changing the underlying widespread protocol standard.

Several protocols to support real-time communication over shared-medium Ethernet have been proposed [1] [2] [3]. However, these protocols are either changing the Ethernet standard or do not add guaranteed real-time services. Real-time communication over switched Ethernet has also been proposed (called EtheReal) [4]. The goal of the EtheReal project was to build a scaleable real-time Ethernet switch, which support bit rate reservation and guarantee over a switch without any hardware modification of the end-nodes. EtheReal is throughput oriented which means that there is no or limited support for hard real-time communication, it has no explicit support for periodic traffic so it is not suitable for industrial applications. A review of research on real-time guarantees in packet-switched networks is found in [5].

This paper presents work on a previously proposed full duplex switched Ethernet network with support for both bit rate and timing guarantees for periodic traffic [6]. Only a thin layer is needed between the Ethernet protocols and the TCP/IP suite in the end-stations. The switch is responsible for admission control, while both end-stations and the switch have EDF (Earliest Deadline First) scheduling [7]. The deadlines of messages over the network are end-to-end based, insofar as it is the maximum time to deliver, from the release time in the source node, to the arrival in the destination.

In this paper, we assume a single switch, with one node connected to each physical port. The messages originating from the source do therefore traverse two links, and we need to provide guarantees for the time to deliver over both links. We approach this problem by dividing the end-to-end deadline into two, one for the source to the switch, and one from the switch to the destination. The deadline can be partitioned in a number of ways. The method we choose affects the system. The paper is concerned with analyzing the partitioning of deadlines, and to propose a way that is more suitable for master slave communication, which is a common demand in industrial applications.

The results, and indeed the method in its current form, do not refer to a mixed topology. The network topology is confined to a star, with one centralized switch connected to one node on each physical port. A full-duplex network is assumed.

The rest of the paper is organized as follows. The network architecture is presented in Section 2. In Section 3, a feasibility analysis is introduced. In section 3, the Asymmetric Deadline Partitioning Scheme (ADPS) with the simulation results are presented. The paper is concluded in Section 4.

2 Network architecture

We consider an example of a network with full-duplex switched Ethernet. The end-nodes are occasionally referred to as master nodes and slave nodes. This is only to reflect typical industrial networks and is not a system requirement. In other words, the network can be use for arbitrary traffic situations. Both the switch and the nodes have added software (a thin RT layer) to support guarantees for real-time traffic. All nodes are connected to the switch and master nodes can communicate with slave nodes over logical real-time channels (RT channels), each being a virtual connection between two nodes in the system (Figure 1). Each master node is responsible for a number of slave nodes and RT channels carry traffic from master nodes to slave nodes. Depending on what nodes are connected to the switch, and what channels are present, we can see a

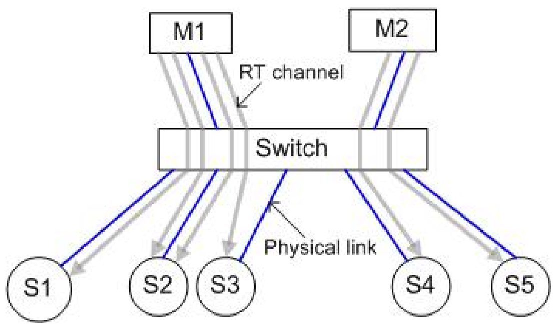


Figure 1: Network configuration, master-slaver traffic pattern with RT channels

difference of System State. We define the System State (SS) as follows:

$$SS=\{N,K\} \quad (1)$$

where N is the set of nodes in the system and K is the set of channels running on the system at present.

The network supports dynamic adding of RT channels to guarantee periodic real-time traffic. An RT channel with index i is characterized by:

$$\{Source_i, Destination_i, P_i, C_i, d_i\} \quad (2)$$

where P_i is the period of data, C_i is the amount of data per period, and d_i is the relative deadline used for the end-to-end EDF scheduling. P_i, C_i, and d_i are expressed as the number of maximal sized frames, i.e., the number of T_{frame}. $Source_i$ and $Destination_i$ are Ethernet MAC addresses for channel i's source and destination nodes respectively. It is evident that both $Source_i$ and $Destination_i$ are parts of the system, i.e. $Source_i \in N;\ Destination_i \in N$.

2 Feasibility Analysis

We proposed to use the periodic synchronization-messages that, together with real-time traffic can be viewed as multiple identical tasks from the switch to each node. In the system a synchronization frame is sent at the start of every sync-period. This affects the feasibility test. The synchronization task (ST) has:

- $C = 1$ (it consists of one frame only)
- $T = 10$ *(a sync-message is sent every ten timeslots)*
- $d = 1$

To have the capacity set to 1 and the period to 10 is straightforward, but why should the deadline be one? The reason is that the periodic sync-message is not a real channel but is viewed as one for purposes of determining feasibility only. No matter what the other channels have as deadlines, this ST will always have priority. The feasibility test will still be correct though, if the deadline is 1, because this is the lowest possible value a task with capacity 1 can have.

A channel must by definition traverse two physical links, one from the source to the switch, and one from the switch to the destination (hereafter denoted as upload and download respectively). For a given channel, it is required for the switch to provide guarantees for both the *uplink* and the *downlink* part.

Theoretically, there is a gain to be made by dividing the concept of a channel into two parts, upload and download. The reason is that one can then look upon each part of the channel as a periodic task, and the corresponding link would constitute a CPU or processing system (from a scheduling point of view). The capacity, C_i, would be the worst-case-execution-time (WCET) for the task. Furthermore, because the system is full duplex, each link would constitute two independent CPUs, one executing the download parts of all channels traversing the link, and the other executing the upload parts (hereafter we will refer to one full duplex link as two links; one upload and one download).

The duty of the download link is then to ‘execute’ the set of tasks assigned to it, in the order decided by the switch, i.e. to carry out the EDF schedule set forth by the switch. For the upload link the RT layer software handles the scheduling, but otherwise the same is true. Before we describe the feasibility test, we make the following definitions:

- *"The **Utilization** factor": According to basic EDF theory the **utilization** of periodic real-time traffic is defined as*

$$U = \sum C_i / P_i\ . \quad (3)$$

- *"The **Hyperperiod**": The **Hyperperiod** for a set of periodic tasks is defined as the length of time from when all tasks' periods start at the same time, until they start at the same time again*

- *"The BusyPeriod": a **BusyPeriod** is any interval of time in which a link is not idle.*
- *“The workload function”: **h(n, t)** is the sum of all the capacities of the tasks with absolute deadline less than or equal to t, running on link n, where t is the number of timeslots elapsed from the start of the hyperperiod. It is calculated as follows*

$$h(n,t) = \sum_{i \in K_n} (1 + \left\lfloor \frac{t - d_i}{P_i} \right\rfloor) C_i \quad (4)$$

where K_n is the set of tasks running on link n (see Figure 2).

- *A feasible link is a link with a set of channels traversing it that can be feasibly scheduled using EDF.*
- *A feasible system state is a system state with every link in the system being feasible.*

Following the discussion from above, and the new definitions, the problem for the switch to test if the channel can be added is therefore equivalent to testing if the new state is still feasible, given that the new channel has been added. The feasibility test of a link, according to [8] [9], is done in two steps, each step being a test of its own.

- *(First Constraint)*

The utilization of the link has to be less than or equal to one (100%)

- *(Second Constraint)*

For all values of t, the workload function h(n,t) has to be less than or equal to t

3 Asymmetric Deadline Partitioning Schemes (ADPS)

We have mentioned above the method of looking at the links as processing units, having tasks to perform. This method is devised in the interest of forcing the test of system feasibility, down to the level of successive tests on links. We need to derive two supposed tasks from each channel. A pair of supposed tasks for the upload and download part of a channel, T_{iu} and T_{id}, is defined as:

$$T_{iu} = \{Source_i, C_i, d_{iu}, P_i\} \qquad (5)$$

$$T_{id} = \{Destination_i, C_i, d_{id}, P_i\} \qquad (6)$$

where $Source_i$, $Destination_i$, C_i and T_i are the parameters of the channel i. d_{iu} and d_{id} are the relative deadlines for the tasks. The only new information in the tasks, compared with the channel, is the relative deadlines. Considering the relative deadlines of the tasks, d_{iu} and d_{id}, as the guaranteed worst case time to deliver from the source to the switch, and from the switch to the destination respectively, we come to the following conclusion. Creating T_{iu} and T_{id} from channel i is accomplished by partitioning the deadline of the channel into two parts: d_{iu} and d_{id} where

$$d_i = d_{iu} + d_{id} \qquad (7)$$

$$d_{iu} , d_{id} \geq C_i \quad ; \quad (if\ d_i \geq 2C_i) \qquad (8)$$

We make the following definition:

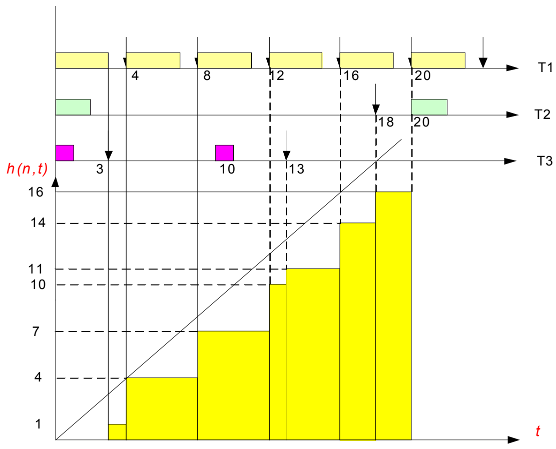


Figure 2: Example of the workload function

- ***A Deadline-Partitioning Scheme (DPS)*** *is a function that maps the deadlines d_i of all the channels in the system into two deadlines d_{iu}, d_{id} such that the condition (Equation 7) is upheld for each RT channel.*

The Domain of the function DPS is thus all the possible system states (Equation 1). The presence of a DPS gives us the freedom to create d_{iu} and d_{id} from every channel i. In fact, the availability of a DPS is not optional, but the system cannot operate without a DPS. There are different ways of looking at DPSs, but the most mathematically satisfying one is as a multi-dimensional function. The dimension of the function is then

$$dim = size(K) \qquad (9)$$

where K is the set of channels in the system state.

We can make the DPS more agreeable as a function, by turning it into a vector field, with the range of its elements fixed between 0 and 1. To start out with, the function would not generate scalars, but it would be *dim* number of pairs of deadlines, $\{ d_{iu}, d_{id} \}$. We now take steps to change this function. First, we normalize with the original deadline, d_i for each corresponding pair. Because of (7) this would mean that we would have pairs, ranging from 0 to 1. The output would look like:

$$Upart_i = d_{iu} / d_i$$
$$Dpart_i = d_{id} / d_i \qquad (10)$$

where $Upart_i$ and $Dpart_i$ are the factors of d_i to get d_{iu} and d_{id}, respectively. But because of (10) we conclude that:

$$Upart_{,i} = 1 - Dpart_i \qquad (11)$$

This means that both $Upart_i$ and $Dpart_{,i}$ contain all the information by themselves. Only one is sufficient, and thus we have:

$$Upart = DPS(SS) \qquad (12)$$

Upart is a *dim* dimensional vector, of elements ($0 < Upart < 1$).

The number of different possible DPSs is infinite, and in this paper we present Asymmetric Deadline Partitioning Scheme (ADPS).

With bottlenecks we mean links with a greater number of channels traversing them than other links. We say that bottlenecks have a higher link-load, which we define in the following manner.

- *The* ***LinkLoad (LL)*** *of a link is the number of channels traversing it which is the same as the number of tasks running on the link.*

The parameters of the channels can also be taken into account when calculating the *LL* but this is not treated here.

A logical approach in the case of a bottleneck is for the system to partition deadlines of the channels that traverse the bottleneck, so that as much of the deadlines of the channels can be found in the tasks of the bottleneck. Obviously, the SDPS does not do anything to relieve bottlenecks, as the SDPS, as stated above, is invariant of the System State.

The ADPS is a DPS devised to distribute, when possible, the deadline of channels, to where it is most needed, i.e. where the *LL* is greatest. We define *ADPS(SystemState) (ss)* as*:*

$$U_{parti} = LL(Source_i)/(LL(Source_i) + LL(Destination_i)) \quad (19)$$

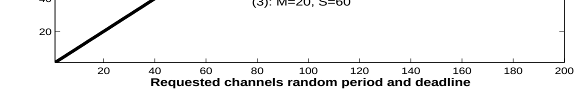
$$D_{parti} = LL(Destination_i)/(LL(Source_i) + LL(Destination_i)) \quad (20)$$

The simulation analysis presented here shows the performance of the network using the EDF scheduling with Asymmetric deadline partitioning. We simulated a network with a single 100 Mbit/s full-duplex Ethernet switch, *M* master nodes, *S* slave nodes, and sets of RT channels. Each RT channel is randomly generated with uniformly distributed source (master node) and destination (slave node). Figure 3 shows the different results when the number of master nodes is varied between 12, 15 and 20, the number of slave nodes is 60. Periods are randomly generated in a range from 80 to 120, while deadlines range from 30 to 50. We can see the different results when each master node has responsibility for 3,4 or 5 slave node. In these cases, all the deadlines d_i and period P_i are generated randomly.

Due to limited space, the result for symmetric deadline partitioning scheme is not presented here.

4 Conclusion

So far, we have only discussed DPSs in terms of their performance. There are other things to consider besides how well the DPS performs, however. The most obvious considerations are perhaps the ones of computational speed and memory requirements. In comparison with deviding deadline into two, with ADPS we gain in DPS performance, but we lose time for calculating the new elements of the algorithms and we need more memory to keep track of data structures required to implement them. The simulation shows that ADPS is especially suitable for a network with a traffic pattern that generates bottlenecks, of which the Master-Slave pattern is the most relevant.



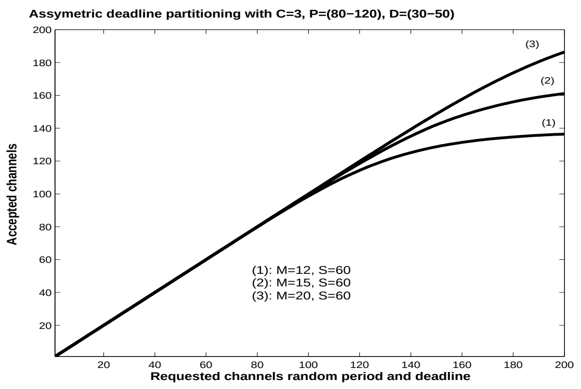


Figure 3: Number of accepted channels with ADPS. Number of master nodes is variable while number of slave nodes is constant.

References

[1] C. Venkatramani and T. S. Chiueh, "Supporting real-time traffic on Ethernet," *Proc. 15th IEEE Real-Time Systems Symposium (RTSS'94)*, pp. 282-286, 1994.

[2] D. W. Pritty, J. R. Malone, D. N. Smeed, S. K. Banerjee, and N. L. Lawrie, "A real-time upgrade for Ethernet based factory networking", *Proc. IEEE IECON'95*, vol. 2 , 1995.

[3] S.-K. Kweon, K. G. Shin, and G. Workman, "Achieving real-time communication over Ethernet with adaptive traffic smoothing," *Proc. 6th IEEE Real-Time Technology and Applications Symposium (RTAS'2000)*, Washington, D.C., USA, 31 May - 2 June 2000, pp. 90-100.

[4] S. Varadarajan and T. Chiueh, "EtheReal: A host-transparent real-time Fast Ethernet switch," *Proc. ICNP*, Oct. 1998.

[5] H. Zhang, "Service disciplines for guaranteed performance service in packet switching network," *Proc. of the IEEE*, vol. 83, no. 10, Oct. 1995.

[6] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, "Switched real-time Ethernet with earliest deadline first scheduling - protocols and traffic handling", *Proc. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2002) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS'02)*, Fort Lauderdale, FL, USA, April 15-16, 2002.

[7] C. L. Liu and J. W. Layland, "Scheduling algorithms for multipprogramming in hard real-time traffic environment", *Journal of the Association for Computing Machinery*, vol. 20, no. 1, Jan. 1973.

[8] C.M.Krishna and K. G. Shin, *Real-Time Systems*. McGraw-hill international edition.1997

[9] J. A.Stankovic, M. Spuri, K. Ramamritham, Giorgio C. Buttazzo, *Deadline Scheduling for Rreal-Time Systems - EDF and Related Algorithms*. Kluwer Academic Publishers*,* 1998.