Thesis for the degree of Licentiate of Engineering

Switched Real-Time Ethernet for Industrial Applications

Hoai Hoang

Department of Computer Engineering
School of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

School of Information Science, Computer and Electrical Engineering

HALMSTAD UNIVERSITY

Göteborg, Sweden, 2003
Switched Real-Time Ethernet for Industrial Applications

HOAI HOANG

Copyright © Hoai Hoang, 2003. All rights reserved

Technical Report No.20L
Department of Computer Engineering
School of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Hörsalsvägen 11
SE-412 96 Göteborg
Sweden
Telephone: +46 - (0)31 - 772 10 00
Fax: +46 - (0)31 - 772 36 63

Contact Information:
Hoai Hoang
School of Information Science, Computer and Electrical Engineering
HALMSTAD UNIVERSITY
SE-301 18 Halmstad
Sweden
Telephone: +46 - (0)35 - 16 71 00
Fax: +46 - (0)35 - 12 03 48
Email: Hoai.Hoang@ide.hh.se

Printed in Sweden
Chalmers Reproservice
Göteborg, Sweden, April 2003
Switched Real-Time Ethernet for Industrial Applications

HOAI HOANG

School of Information Science, Computer and Electrical Engineering, Halmstad University.
Department of Computer Engineering, Chalmers University of Technology

Thesis for the degree of Licentiate of Engineering

Abstract

The research reported in this thesis has been focused on developing and analyzing how to support real-time traffic over a switched Ethernet network without any hardware or protocol modifications. The work has resulted in a proposed systems model, supporting both real-time and non real-time traffic. Currently this model is intended for a one-switch network, with no shared media. All added traffic handling to support real-time communication is positioned in a thin layer (RT layer) added between the Ethernet layer and the TCP/IP suite. This assures adaptation to the surrounding protocol standards. The RT layer manages traffic on the basis of virtual connections, denoted as RT channels, as well as packet level scheduling. RT channels are created between end-nodes prior to any occurrence of real-time traffic. Asymmetric deadline partitioning between the links of the RT channels is also proposed, in order to increase the number of possible RT channels.

Keywords: Switched Ethernet, Real-time communication, EDF scheduling, Industrial networks.
Acknowledgements

Since I started my Ph.D. study, many people have helped and supported me along the way. Without them, the thesis would obviously not have looked the way it does now.

First I would like to thank my supervisor, Professor Magnus Jonsson, who introduced me to this field of research and assisted me in many ways. His enthusiastic engagement in my research and his never ending stream of ideas has been absolutely essential toward the result I present here. I am very grateful that he has spent so much time with me discussing different problems.

I would like to express my thanks to Professor Bertil Svensson for his encouragement, and for giving me many great opportunities to grow as researcher under his thoughtful guidance.

I would also like to thank the members of CC-Lab and the other staff at the school of IDE for being good friends as well as helpful colleagues. In particular, many thanks go to Veronica Gaspes, Eva Nestius and Ulrika Hult for administrative help and to the other PhD-students at CC-lab for valuable support and fruitful discussions.

I am thankful to my friends here and there, for their encouragement and help, especially Stefan Karlsson.

Finally and heartfelt thanks go to my parents, my sister Hienmit, Viet and my little Gau for love and support.

This work is funded by ARTES - A network for Real-Time research and graduate Education in Sweden, www.artes.uu.se.

Hoai Hoang, Halmstad University, May 2003
List of appended papers

This thesis contains a thesis summary, followed by three technical papers. The first paper has been accepted for publication in a special issue of the journal: Parallel and Distributed Computing Practices [Paper A]. One paper [Paper B] is published in a conference proceeding and the final one [Paper C] has been submitted for reviewing.

Appended papers:


Other related but not appended papers:


Contents

Abstract .................................................................................................................... iii
Acknowledgement .................................................................................................. v
List of appended papers ..................................................................................... vii
Contents ............................................................................................................... ix
Thesis Summary ................................................................................................ 1
1. Introduction ..................................................................................................... 1
2. Related work ................................................................................................... 1
3. A new architecture for Real-time communication over switched Ethernet networks ........................................................................................................... 4
   3.1 Network architecture ................................................................................ 4
   3.2 Real-time traffic handling ........................................................................ 4
4. Deadline Scheduling ....................................................................................... 7
   4.1 Scheduling Algorithm ............................................................................ 7
   4.2 Feasibility Analysis .............................................................................. 7
   4.3 Deadline partitioning schemes ...................................................... 10
5. Conclusions .................................................................................................... 12
6. Summary of the appended papers ................................................................ 13
7. References ...................................................................................................... 14
Thesis Summary

1. Introduction

An important trend in the networking community is to involve more switches in the networks (e.g., LAN, Local Area Networks) and pure switched-based networks become more and more common [6]. 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 has also improved. Now when the cost of LAN switches has reached the level where pure switched-based networks have become affordable, the collision risk 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 wide-spread protocol standard. The main research question is how to form methods to support typical industrial real-time traffic without changing the underlying protocols.

This thesis proposes a switched Ethernet based network configuration, which supports real-time communication with guaranteed bit rate and worst-case latency. No modification to the Ethernet standard is needed in the network that supports both real-time and non-real-time TCP/IP communication. The switch is responsible for admission control where feasibility analysis is made for each link between source and destination. Both the switch and the end-nodes use EDF (Earliest Deadline First) scheduling to control real-time traffic.

One important factor with impact on how well the system performs is the manner in which the deadlines of packets are partitioned, over the links that they sequentially traverse. This is explored, and the definition of a Deadline Partitioning Scheme (DPS) is stated. Two specific DPSs are introduced and compared with each other with respect to performance of the network.

2. Related work

This chapter summarizes the research area of real-time communication over Ethernet. Distributed real-time computer applications basically impose two requirements to the communication network: guaranteed throughput and bounded delay.
Implementing distributed real-time applications on top of standard Ethernet-based networks using TCP/IP and UDP is a particular approach proposed for factory automation [9][14][16][17][18][19]. Using switched Ethernet with the TCP/UDP/IP suite [20] or client-server based protocol [8] is a particular approach in industrial applications. Fieldbus [21][22], Ethernet/IP (Ethernet/ Industrial Protocol) [7], IDA [10] are the examples of industry automation networks that are based on the Internet protocol over Ethernet. Ethernet/IP uses standard Ethernet technology and the TCP/IP suite protocol together with CIP (Control and Information Protocol). Ethernet/IP exchange time-critical data based on the producer/consumer model on top of the TCP/IP suite. IDA (Interface for Distributed Automation), is an Ethernet-based real-time LAN for automation applications that has been presented in [10]. The IDA communication system provides both real-time and non real-time communication services. Real-time services use the RTPS (Real-time Publish/Subscriber) protocol, which is based on the UDP protocol. In other papers, researchers suggest to smooth the non-real-time traffic to improve the characteristics of real-time traffic [15].

In RETHER [3], the authors proposed and evaluated a software-based protocol that provides real-time performance guarantees to multimedia applications. RETHER works in two modes, CSMA-mode for non-RT traffic, while switching to RETHER-mode for RT traffic, this decision is to minimize the performance impact on non-real-time traffic when there is no RT traffic. When a node receives an RT request from a local application, it becomes an initiator by broadcasting a Switch-to-RETHER message on the Ethernet. Every node that receives this message then switches to RETHER mode. Each node then holds off sending any data and awaits completion of transmission of the packets that are already put in the buffer. After this, each node sends an acknowledgement to the initiator. As soon as the initiator has received all the acknowledgements, it creates a token. If there are more than one node that wants to send an RT request and all of the initiators content for initiation, collision is possible. The initiator may finally reach time out because of some acknowledgement may be lost or some nodes do not receive a Switch-to-RETHER message.

RETHET adopts a timed-token passing scheme to provide bandwidth guarantees. An average constant bandwidth is reserved for each session. Only nodes that have made a bandwidth reservation belong to the RT set while all other nodes belong to the non-RT (NRT) set. The token visits all nodes in the RT set in order, where one RT node can send one unit of RT data before passing the token to its neighbor in the RT set. The last node in the RT set passes the token to the first node in
the NRT set. The token could visit RT nodes more often than NRT nodes, i.e. giving the RT traffic priority over NRT traffic. There is no support for delay bounds in RETHER.

An important feature of RETHER is that the admission control decision is made locally in a node at the time it receives the token. The reason is that the token carries most of the information about the RT request and the bandwidth reservations. This may, however, lead to incorrect admission decisions when two nodes receive RT requests simultaneously and both nodes admit the request without knowledge of admission decisions made from other nodes. In our work, the admission control is made in the switch instead.

With the same authors as RETHER, the goal of EtheReal [4] is to build a scalable real-time Ethernet switch that supports bandwidth reservation/guarantee over a switched LAN environment without any hardware or OS (Operating System) modification. All nodes are connected to each other via the switches. Special software has to be installed in each node to support real-time traffic. The QoS (Quality of Services) in EtheReal is mainly defined as the predefined bit rate real-time traffic with priority scheduling. EtheReal has supported also for non real-time traffic, i.e. the switch will support ordinary Ethernet traffic to co-exist with real-time traffic. A weakness of Ethereal is that the influence of ordinary traffic on non real-time traffic is not well controlled.

Simply adding a switch to an Ethernet network as mentioned above is not enough to guarantee real-time behavior. In order to avoid the kind of collisions that appear in shared medium Ethernet networks [10], it is necessary to restrict the topology so that consists of only full duplex point-to-point links. Our purpose is to use full-duplex switched Ethernet for real-time communication. By adding a RT layer between Ethernet and TCP/IP, we can support real-time traffic while non real-time service is still provided by UDP. The use of EDF scheduling in the end-nodes and the switch guarantees the QoS of the RT traffic. Moreover, we focus on achieving real-time performance without modifying the Ethernet hardware.
3. A new architecture for Real-time communication over switched Ethernet networks

3.1 Network architecture

We assume a network with a switched Ethernet topology, in which both the switch and the end-nodes have a RT layer added to support guarantees for real-time traffic. Each node is connected to other nodes via the switch allowing the nodes to communicate with each other over logical real-time channels (RT channels), while still supporting regular non-real-time traffic. Full-duplex switched Ethernet is assumed for the network. End-nodes have the capability of controlling outgoing traffic from the nodes using the Earliest Deadline First (EDF) algorithm. The switch has the same capability. The proposed network is well suited for, e.g., master-slave communication, in which the end-nodes are divided into master nodes and slave nodes according to Figure 1.

3.2 Real-time traffic handling

The function and interaction with the RT layer etc is shown in Figure 2. When an application wants to set up an RT channel, it interacts directly with the RT layer (1). The RT layer then sends a question to the RT channel management software in the switch (2). Outgoing real-time traffic from the end-node uses UDP and is put in a deadline-sorted queue in the RT layer (3). Outgoing non-real-time traffic from the end-node typically uses TCP and is put in a FCFS-sorted (First Come First Serve) queue in the RT layer (4). In the same way, there are two different output queues for each port on the switch too (5).
The network supports dynamic addition of RT channels. Each RT channel is a virtual connection between two nodes in the system with guaranteed bit rate and a maximum latency. A RT channel with index $i$ is characterized by $\{P_i, C_i, d_i\}$, 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. Both $P_i$, $C_i$, and $d_i$ are expressed as the number of maximum sized frames.

Before real-time traffic may be delivered, an RT channel must be established. The creation of an RT channel consists of request and acknowledgment communication where the source node, the destination node, and the switch agree on the channel establishment. After that, the nodes can begin to use the channel. Both the switch and the end-nodes have software (RT layer) added (see Figure 2) which shapes the traffic on the RT channel. The network guarantees to deliver each generated message with a bounded delay over the RT channel. When a node wants to establish an RT channel, it sends a RequestFrame to the switch, which includes (see Figure 3): source and destination node MAC and IP addresses and $\{P_i, C_i, d_i\}$. The connection ID field is set to a unique ID for each source node, so that the response
can be distinguished in the case of several requests. At this stage, the RT channel ID field is not set with a valid value yet.

When receiving a RequestFrame, the switch will calculate the feasibility of the scheduling the channel between the requesting node and the switch and between the switch and the destination node (admission control). If the switch finds it is feasible to schedule the channel, the RequestFrame is then forwarded to the destination node, after adding a network unique ID in the RT channel ID field. The destination node responds with a ResponseFrame (see Figure 4) to the switch, telling whether the establishment is accepted or not. The switch will then, after taking notation of the response, forward the ResponseFrame to the source node. If the switch did not find the requested RT channel feasible to schedule, the RequestFrame is not forwarded to the destination node. Instead, a ResponseFrame is sent directly to the source node reporting the rejection.

![Data field in Ethernet-frame containing connection response details]

<table>
<thead>
<tr>
<th>Source MAC addr. = switch addr.</th>
<th>48 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>8 bits</td>
</tr>
<tr>
<td>Response:</td>
<td>Response:</td>
</tr>
<tr>
<td>packet</td>
<td>0 = Not OK</td>
</tr>
<tr>
<td>1 = OK</td>
<td>1 bit</td>
</tr>
<tr>
<td>Connect. request ID</td>
<td>8 bits</td>
</tr>
<tr>
<td>RT channel ID</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

*Figure 4: ResponseFrame*

All end-nodes are synchronized by synchronization frames that are sent periodically from the switch. The synchronization frames also give flow-control information for non-real-time traffic, i.e., reporting the buffer status of the switch. For real-time traffic, the switch always checks at RT channel establishment if there is enough buffer space.

The RT layer in an end-node prepares outgoing real-time IP datagrams by changing the IP header (in the data part of an Ethernet frame) before allowing the Ethernet layers to forward the datagram to the switch. The IP source address and the 16 most significant bits of the IP destination address, 48 bits together, are set to the absolute deadline of the frame. A 48 bit absolute deadline with a resolution of $T_{frame} = 125 \mu s$, gives a “life time” longer than one thousand years. The 16 least significant bits of the IP destination are set to the RT channel ID for the RT channel to which the frame belongs. The Type of Service (ToS) field is always set to value 255. Other values than 255 in the ToS field can be used for future services.
The switch has two MAC addresses, one for control traffic (e.g., RequestFrames) and one for real-time traffic over RT channels. A non-real-time frame carries the final destination MAC address in the Ethernet header already when leaving the source node. The end-nodes recognize control frames by reading the MAC source address that is set to the switch address. The switch exchanges the source and destination IP addresses and the MAC destination address of an incoming real-time frame with the correct ones (as stored in the switch when the RT channel was established) for delivery to the final destination. Also, the IP header checksum and the Ethernet CRC are recalculated before putting the frame in the correct deadline-sorted output queue. The checksums of non-real-time frames do not need to be recalculated.

4. Deadline Scheduling

4.1 Scheduling Algorithm

As mentioned above, both the switch and the end-nodes use the EDF algorithm to control the outgoing flow of RT traffic. An RT channel must traverse two physical links, one from the source node to the switch (uplink), and one from the switch to the destination node (downlink). The relative deadline of a RT channel is divided into two parts: $d_{iu}$ and $d_{id}$, as the guaranteed worst-case time to deliver on the uplink and downlink, respectively. The source node controls the traffic flow on the uplink and the switch controls the downlink part.

The network guarantees to deliver each generated message with a bounded delay:

$$T_{\text{max, delay},i} = d_i + T_{\text{latency}}$$

Where $T_{\text{latency}}$ is a value determined by the medium propagation delay and the medium access time. Consider a system consisting of one centralized switch and several end-nodes with identical connections to the switch. In such a system, $T_{\text{latency}}$ can be considered as a system specific constant.

4.2 Feasibility Analysis

The switch checks the feasibility of accepting a new RT channel, using an EDF theory modified to reflect the characteristics of the Ethernet network proposed.
Each part of the RT channel can be looked upon 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 organize two independent CPUs, one executing the download parts of all channels traversing the link, and the other executing the upload parts. Following are some definitions that are used throughout this chapter.

The Utilization factor: According to basic EDF theory [23] the utilization of periodic real-time traffic is defined as:

$$U = \sum \frac{C_i}{P_i}$$

(2)

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:

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

(3)

A feasible link: A feasible link is a link with a set of channels traversing it that can be feasibly scheduled using EDF.

The system state: The system state, denoted SS, is defined by the pair: $\{N, K\}$, where $N$ is the set of nodes connected to the system, and $K$ is the set of RT channels currently active.

A feasible system: A feasible system is a system state (SS) 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 equivalent to testing if the new state is still feasible, given that the new channel has been added. The feasibility test of a link is done in two steps, each step being a test of its own, as shown below.
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 

The following pseudo-code shows the general idea:

For each link \( n \) in the system:

```java
{ 
    // 1:st constraint
    If Utilization(\( n \)) > 1 Then "not feasible"

    // 2:nd constraint
    For all positive integers \( t \):
        If \( h(n, t) > t \) Then "not feasible"
}
```

Liu and Layland [24] showed that the first constraint is enough for the RT channels that have relative deadlines equal to their period. In [paper A], it is assumed that all RT channels have their deadlines equal to their periods (see [paper A] for more exact assumptions). In such a scenario, it is enough for the switch to check utilization only, in order to determine if a RT channel can be added or not.

The second constraint, in the form given above, does not lend itself out particularly well to computation. It is shown in [2] how to reduce the time and memory complexity of the second constraint check. If \( h(n, t) \leq t \) in the first busy period of the hyperperiod in the supposed schedule to come, then \( h(n, t) \leq t \) for all \( t \). The following upperbound would therefore be an improvement of the algorithm above:

\[
t, \text{ such that } 1 \leq t \leq \text{BusyPeriod}(n)\] (4)

where \( \text{BusyPeriod}(n) \) is the first \( \text{BusyPeriod} \) in the schedule at the start of the hyper-period, on link \( n \).

Further more, one does not need to check every integer from the first timeslot, but only the integers \( t \) where
assuming that $Q$ denotes the number of RT channels traversing the considered link in the considered direction.

### 4.3 Deadline partitioning schemes

The method of looking at links as processing units is used, each link 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. For this approach to work, it is required 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:

\[
T_{iu} = \{\text{Source}_i, P_i, C_i, d_{iu}\} \quad (6)
\]

\[
T_{id} = \{\text{Destination}_i, P_i, C_i, d_{id}\} \quad (7)
\]

Where $\text{Source}_i$ and $\text{Destination}_i$ are the source and destination nodes for the channel and $d_{iu}$ and $d_{id}$ are deadlines for the tasks on the uplink part and downlink part respectively. Obtaining such tasks is accomplished by partitioning the deadline of the channel into two parts: $d_{iu}$ and $d_{id}$ where

\[
D_i = d_{iu} + d_{id} \quad (8)
\]

\[
d_{iu}, d_{id} \geq C_i \quad (if \ D_i \geq 2C_i) \quad (9)
\]

Condition (8) must be satisfied, because otherwise the channel as a whole will be different. If one divides a task into separate smaller tasks, it should follow that, if the original task had a deadline, then the sum of the subtasks’ deadlines must equal this larger deadline. Condition (9) should be upheld because otherwise the partitioning will automatically yield a non-EDF-feasible situation. The deadline cannot be allowed to be shorter than the capacity as the capacity is the WCET of the assumed tasks. We can also assure ourselves that if $D_i < 2C_i$ then the channel cannot, by definition, be EDF feasible.

A deadline-partitioning scheme is defined as:

- **DPS** *(Deadline-Partitioning Scheme)* 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 Equation (8) is satisfied for all RT channels.
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

$$\text{dim} = \text{size}(K)$$  \hspace{1cm} (10)

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 $\text{dim}$ 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 (8) this would mean that we would have pairs, ranging from 0 to 1. The output would look like:

$$U_{\text{part},i} = d_{iu} / d_i$$
$$D_{\text{part},i} = d_{id} / d_i$$ \hspace{1cm} (11)

where $U_{\text{part}}$ and $D_{\text{part}}$ are the factors of $d_i$ to get $d_{iu}$ and $d_{id}$, respectively. But because of (8) we conclude that:

$$U_{\text{part},i} = 1 - D_{\text{part},i}$$ \hspace{1cm} (12)

This means that both $U_{\text{part},i}$ and $D_{\text{part},i}$ contain all the information by themselves. We can now write a DPS in the general form:

$$U_{\text{part}} = \text{DPS(system state)}$$ \hspace{1cm} (13)

$U_{\text{part}}$ is a $\text{dim}$ sized vector of elements ($0 < U_{\text{part},i} < 1$).

The number of different possible DPSs is infinite. In [paper B] and especially in [paper C] two different DPSs are examined: Symmetric (SDPS) and Asymmetric (ADPS).

- **SDPS (Symmetric Deadline Partitioning)**

In [paper A], it was proposed to partition the deadline of each channel into two equal parts, i.e. to split it in half. Following the notation introduced above, this would imply that

$$d_{iu} = d_{id} = d_i / 2$$ \hspace{1cm} (14)

$$U_{\text{part},i} = D_{\text{part},i} = 1 / 2$$ \hspace{1cm} (15)
We define this approach as a Symmetric DPS (SDPS). It is easily seen that condition (8) is satisfied under this function. We can also note that the SDPS only depends on the size(K) of the system state. In the view of DPSs as vector fields this means that the SDPS is represented by a vector of size(K) number of elements, with each element being constant, equal to 0.5. Obviously, as the SDPS doesn’t take into consideration what the system looks like, we should be able to propose a better DPS.

- **ADPS (Asymmetric Deadline Partitioning Scheme)**

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 propose to 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 be taken into account also when calculating the LL but this is not treated here.

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 as:

\[
U_{\text{part},i} = \frac{LL(\text{Source}_i)}{LL(\text{Source}_i) + LL(\text{Destination}_i)}
\]

\[
D_{\text{part},i} = \frac{LL(\text{Destination}_i)}{LL(\text{Source}_i) + LL(\text{Destination}_i)}
\]

5. Conclusions

In this thesis, we present a switched Ethernet based network concept supporting real-time communication with guaranteed bit rate and worst-case delay for periodic traffic. Two deadline-partitioning schemes are presented. While SDPS is straightforward to implement, ADPS is devised in order to have a more flexible feasibility test. ADPS in show promises to remove bottleneck from links. In [paper C] master-slave communication is considered as the traffic pattern, and ADPS proves to be a better choice than SDPS.

Future work includes investigating the use of more complex network topologies, i.e, networks consisting of many interconnected switches and links having shared medium. Alternative communication models and scheduling algorithms could be explored as well.
6. Summary of the appended papers

**Paper A: Switched Real-Time Ethernet with Earliest Deadline First Scheduling – Protocols, Traffic Handling and Simulation Analysis**

This paper presents enhancements to full-duplex switched Ethernet for the ability of giving throughput and delay guarantees. The switch and the end-nodes control the real-time traffic with Earliest Deadline First (EDF) scheduling on the frame level. No modification to the Ethernet standard is needed in the network that supports both real-time and non-real-time TCP/IP communication. The switch is responsible for admission control where a feasibility analysis is made for each physical link between source and destination. The switch broadcasts Ethernet frames regularly to clock synchronize the end nodes and to implement flow control for non-real-time traffic. We have characterized the performance of the network in terms of channel acceptance ratio by simulations with different numbers of nodes connected to the switch.

**Paper B: Deadline First Scheduling in Switched Real-Time Ethernet – Deadline Partitioning Issues and Software Implementation Experiments**

A network configuration with a master node and several slave nodes are considered. One master node is responsible for a number of slave nodes. The deadline of an RT channel is partitioned differently depending on the number of slave nodes that are connected to the master node. We show by example that the performance can benefit a lot from asymmetric deadline partitioning. An increase in utilization from 50 % to 75 % (with the network of one master node and 5 slave nodes) on the outgoing link from the master node (the bottleneck) is observed, still not violating the real-time guarantees.

From a software-implemented switch, we show experimental results. The measurements show that the switch bottlenecks are 96 Mbit/s (measured for maximum-sized frames) and 14 400 frames/s (measured for minimum-sized frames). From these measurements one can get a good feel for the amount of real-time traffic and number of ports that can be supported by an Ethernet switch that is fully or partly implemented in software.

**Paper C: Switched Real-Time Ethernet in Industrial Applications - Deadline Partitioning Schemes**

This is a continuation of the ideas on asymmetric deadline partitioning proposed in [paper B]. The network configuration is expanded to
comprise many master nodes and many slave nodes. One master node can communicate with more than one slave node via the switch at the same time. Two Deadline Partitioning Schemes (DPS) are proposed, and an effort to mathematically model the workings of DPSs is described. Two DPSs, SDPS (Symmetric DPS) and ADPS (Asymmetric DPS) are compared. As we go from SDPS to ADPS, we gain in DPS performance. 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 a relevant example.

7. References


SWITCHED REAL-TIME ETHERNET WITH EARLIEST DEADLINE FIRST SCHEDULING - PROTOCOLS, TRAFFIC HANDLING AND SIMULATION ANALYSIS*

HOAI HOANG†, MAGNUS JONSSON†, ULRIK HAGSTROM†, AND ANDERS KALLERDAHLþ

Abstract. There is a strong interest of using the cheap and simple Ethernet technology for industrial and embedded systems. This far, however, the lack of real-time services has prevented this change of used network technology. This paper presents enhancements to full-duplex switched Ethernet for the ability of giving throughput and delay guarantees. The switch and the end-nodes controls the real-time traffic with Earliest Deadline First (EDF) scheduling on the frame level. No modification to the Ethernet standard is needed in the network that supports both real-time and non-real-time TCP/IP communication. The switch is responsible for admission control where feasibility analysis is made for each link between source and destination. The switch broadcasts Ethernet frames regularly to clock synchronize the end nodes and to implement flow control for non-real-time traffic. We have characterized the performance of the network in terms of channel acceptance ratio by simulations with different number of nodes connected to the switch.

Key words. Switched Ethernet, Real-time communication, Industrial networks.

1. Introduction. This paper focus on how to form methods to be able to support typical industrial real-time traffic without changing the underlying protocols and while still supporting existing higher-level protocols for non-real-time traffic (e.g., web based maintenance which is highly desirable to coexist with the real-time traffic).

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 with full-duplex support 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 and it has no explicit support for periodic traffic. It is therefore not suitable for typical industrial applications. A review of research on real-time guarantees in packet-switched networks is found in [5].

†School of Information Science, Computer and Electrical Engineering, Halmstad University, Box 823, SE-30118, Sweden. Email: Hoai.Hoang@ide.hh.se and Magnus.Jonsson@ide.hh.se
‡Kvaser AB, Aminogatan 25, SE-43153, Mölndal, Sweden
§Volcano Communications Technologies AB, Gruvgatan 35B, SE-42130 Västra Frölunda, Sweden

*This work was supported by the Swedish Foundation for Strategic Research through the ARTES (A network for Real-Time research and graduate Education in Sweden) Program.
This paper presents a switched Ethernet network with support for both bit rate and timing guarantees for periodic traffic. 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 [6]. Internet communication is supported at the same time as nodes connected to the switch can be guaranteed to meet their real-time demands when they communicate with each other. This is highly appreciated by the industry since it makes remote maintenance possible, e.g., software upgrades or error diagnostics (see Figure 1.1). Some implementation experiments have been done [7], but are not covered in this paper.

The rest of the paper is organized as follows. The network architecture is presented in section 2. Section 3 describes how real-time channels are setup and how real-time traffic is treated. Deadline scheduling, including details on the admission control, is then presented in section 4. Section 5 presents a simulation analysis of the network while the paper is concluded in section 6.

2. Network architecture. We consider a network with the topology of a switched Ethernet (a single switch is assumed in this paper) and end-nodes. Both the switch and the end-nodes have software (RT layer) added to support guarantees for real-time traffic. Every node is connected to other nodes via the switch and nodes can communicate with each other over logical real-time channels (RT channels), each being a virtual connection between two nodes in the system (see Figure 2.1). In our network configuration, end-nodes have the capability of controlling traffic from the nodes using the Earliest Deadline First (EDF) algorithm. The switch has the same capability.

Full-duplex switched Ethernet is assumed for the network, which supports both real-time and non-real-time traffic. MAC function, frame buffering and centralized transmission arbitration are included in the switch. Non-real time frames are redirected based on the MAC destination address. An Ethernet switch must contain address table, address learning and other functions needed to support the standard switching. How real-time frames are treated is discussed in the next section.

The switch periodically sends synchronization frames to the end-nodes, at an interval, $T_{cycle}$, of ten maximum sized frames, $T_{frame}$, i.e.,
In this way, every node has a uniform comprehension about global time. The resolution of the global time is $T_{frame}$. In this paper, we assume Fast Ethernet (100 Mbit/s) with a maximum frame size of 12 144 bits which, with some extra time for timing uncertainties and for simplicity, gives $T_{frame} = 125 \mu s$, which just happens to match the time resolution of many telecommunication systems. The synchronization frames also give flow-control information for non-real-time traffic, i.e., telling the buffer status of the switch. For real-time traffic, the switch always checks, at RT channel establishment, if there is enough buffer space.

The function of and interaction with the RT layer etc shown in Figure 2.2 is explained below. When an application wants to setup an RT channel, it interacts directly with the RT layer (1). The RT layer then sends a question to the RT channel management software in the switch (2). Outgoing real-time traffic from the end-node uses UDP and is put in a deadline-sorted queue in the RT layer (3). Outgoing non-real-time traffic from the end-node typically uses TCP and is put in a FCFS-sorted queue.
(First Come First Serve) queue in the RT layer (4). In the same way, there are two different output queues for each port on the switch too (5).

3. Real-time communication. Below, real-time channel establishment and real-time traffic handling, are discussed respectively. One aim of the section is to explain the function of the switch (see Figure 3.1 for a flow diagram for the switch) and the end-nodes, in terms of real-time communication support.

3.1. Real-time channel establishment. Before real-time traffic may be delivered, an RT channel must be established. The creation of an RT channel consists of request and acknowledgment communication where the source node, the destination node, and the switch agree on the channel establishment. After that, the nodes can begin to use the channel. Both the switch and the end-nodes have software (RT layer) added which shapes the traffic on the RT channel.

An RT channel with index \( i \) is characterized by:

\[
\{ T_{\text{period},i}, C_i, T_{\text{deadline},i} \}
\]

where \( T_{\text{period},i} \) is the period of data, \( C_i \) is the amount of data per period, and \( T_{\text{deadline},i} \) is the relative deadline used for the end-to-end EDF scheduling. Both \( T_{\text{period},i} \), \( C_i \), and \( T_{\text{deadline},i} \) are expressed as the number of maximal sized frames, i.e., the number of \( T_{\text{frame}} \). In this paper,

\[
T_{\text{deadline},i} = T_{\text{period},i}
\]

is assumed. When an RT channel has been established, the network guarantees to deliver each generated message with a bounded delay, \( T_{\text{max\_delay},i} = T_{\text{deadline},i} + T_{\text{latency}} \) (see section 4), expressed in number of \( T_{\text{frame}} \). When a node wants to establish an RT channel, it sends a RequestFrame to the switch (see Figure 3.2), which includes (see Figure 3.3): source and destination node MAC and IP addresses and \( \{ T_{\text{period},i}, C_i, T_{\text{deadline},i} \} \). The connection ID field is set to a source-node unique ID for the ability to distinguish the response in the case of several requests. The RT channel ID field is not set with a valid value yet.

When receiving a RequestFrame, the switch will calculate the feasibility of the traffic schedule between the requesting node and the switch and between the switch and the destination node (admission control). If the switch finds the schedule feasible (see subsection 3.2), the RequestFrame is then forwarded to the destination node, after adding a network unique ID in the RT channel ID field. The destination node responds with a ResponseFrame (see Figure 3.4) to the switch telling whether the establishment is accepted or not. The switch will then, after taking notation of the response, forward the ResponseFrame to the source node. If the switch did not find the requested RT channel feasible to schedule, the RequestFrame is not forwarded to the destination node. Instead, a ResponseFrame is sent directly to the source node telling about the rejection.

The switch has two own MAC addresses, one for control traffic (e.g., RequestFrames) and one for real-time traffic over RT channels. The switch will in this way be able to easy (e.g., in hardware) filter out the different kinds of frames: (i) control frames, (ii) frames belonging to established real-time channels, and (iii) non-real-time frames. A non-real-time frame carries the final destination MAC address in the Ethernet header already when leaving the source node. The end-nodes recognize control frames by reading the MAC source address that is set to the switch address.
3.2. Real-time traffic handling. The RT layer in an end-node prepares outgoing real-time IP datagrams by changing the IP header before letting the Ethernet layers sending it, in the data part of an Ethernet frame, to the switch (see Figure 3.5). The IP source address and the 16 most significant bits of the IP destination address, 48 bits together, are set to the absolute deadline of the frame. A 48 bit absolute
Fig. 3.2. Establishment of an RT channel.

Fig. 3.3. RequestFrame sent by source node trying to establish an RT channel.

deadline with a resolution of $T_{frame} = 125 \mu s$, gives a “life time” longer than one thousand years. The 16 least significant bits of the IP destination are set to the RT channel ID for the RT channel to which the frame belongs. The Type of Service (ToS) field is always set to value 255. Other values than 255 in the ToS field can be used for future services.

The switch exchanges the source and destination IP addresses and the MAC destination address of an incoming real-time frame with the correct ones (as stored in the switch when the RT channel was established) for delivery to the final destination. Also, the IP header checksum and the Ethernet CRC are recalculated before putting the frame in the correct deadline-sorted output queue. The checksums of non-real-time frames do not need to be recalculated. A variant is to just change the MAC destination address in the switch and let the end nodes recover the IP fields.

4. Deadline scheduling. We assume that Node 1 wants to send real-time traffic to Node 2. The traffic is carried over RT channel $i$, where $T_{D1,i}$ and $T_{D2,i}$ are the deadlines for real-time frames from Node 1 to the switch and from the switch to Node 2, respectively. The relation between the period duration, $T_{period,i}$, of the RT channel and the deadlines is:

$$T_{period,i} = T_{D1,i} + T_{D2,i} \quad (4.1)$$

$$T_{D1,i} = T_{D2,i} = \frac{T_{period,i}}{2} \quad (4.2)$$

The scheduling of real-time frames in the switch (and for outgoing real-time frames in the end-nodes) is made according to earliest deadline first, i.e., all incoming real-time traffic is served in deadline order to guarantee the worst-case delay.

When the switch checks the feasibility of accepting a new connection (admission control), it uses an EDF theory modified to reflect the characteristics of the Ethernet network proposed in this paper. According to the basic EDF theory [6], the utilization of real-time traffic is defined as
Data field in Ethernet-frame containing connection response details

| Source MAC addr. = switch addr. 48 bits | Type: Response packet 8 bits | Response: 0 = Not OK 1 = OK 1 bit | Connect. request ID 8 bits | RT channel ID 16 bits |

Fig. 3.4. ResponseFrame.

IP header

| Dest. MAC addr. = switch addr. 48 bits | Type of Service (ToS) 8 bits | IP source address 32 bits | IP dest. address 32 bits | IP data field containing an UDP datagram |

Value = 255

Fig. 3.5. Data frame sent over an RT channel.

\[ U = \sum \frac{C_i}{T_{period,i}} \]  

when the period is equal to the deadline (maximum delay) over the communication link (not true for this system as explained below). To be sure that all deadlines are met, the utilization of real-time traffic must not exceed a certain level, \( U_{max} \):

\[ U = \sum \frac{C_i}{T_{period,i}} < U_{max} \]  

where \( U_{max} \) in the theoretical case is 100 %, i.e., \( U_{max} = 1 \). This guarantee holds for deadline scheduling of traffic when the deadline for a specific link is equal to the period multiplied by a constant \( k \leq 1 \), for all RT channels traversing the link in the same direction. When scheduling a channel with 100 % theoretical utilization, \( U_{max} = k \). For deadline scheduling of traffic with arbitrary deadlines instead, see [8] or subsequent work (e.g. [9]).

The worst-case maximum utilization for a link between the switch and the destination node, \( U_{max2} \), i.e., utilization that can be gained at full load, is reduced from 100 % to 90 % due to having every tenth possible frame being a synchronization frame. Also, because \( T_{D1,i} = T_{D2,i} = T_{period,i} \), the maximum utilization accountable for real-time traffic according to the EDF analysis is \( \frac{90\%}{2} = 45\% \), i.e., we only have half of the period duration to get from the switch to the destination node. In summary we have:

\[ \sum \frac{C_i}{T_{period,i}} < U_{max2} \]
where

\[ U_{\text{max}2} = \frac{T_{\text{cycle}} - T_{\text{frame}}}{2T_{\text{cycle}}} = 0.45 \] (4.6)

In the same way we can calculate the worst-case maximum utilization for a link between the source node and the switch, \( U_{\text{max}1} \). However, we get a higher utilization compared to \( U_{\text{max}2} \) because there is no synchronization frames on the links in this direction. Thus we get:

\[ U_{\text{max}1} = \frac{1}{2} = 0.5 \] (4.7)

In practice, a lower value of \( U_{\text{max}} \) can be used to always allow part of the bandwidth for non-real-time traffic.

In the worst-case situation, when all RT-channel start at the same time or all messages using its RT channel’s full capacity allowance, the RT-channel with the longest deadline will be scheduled at last so it has the worst delay. However, the worst-case delay is, for all RT channels, characterized by:

\[ T_{\text{max delay},i} = T_{D1,i} + T_{D2,i} + T_{\text{latency}} \] (4.8)

or

\[ T_{\text{max delay},i} = T_{\text{period},i} + T_{\text{latency}} \] (4.9)

where \( T_{\text{latency}} \) is the worst-case latency experienced by a frame before it is transmitted even though it has the earliest deadline. The Worst-case latency to be added to the deadline is:

\[ T_{\text{latency}} = 2T_{\text{link prop delay}} + T_{\text{node access}} + T_{\text{switch access}} \] (4.10)

where \( T_{\text{link prop delay}} \) is the maximum propagation delay over a link between an end-node and the switch, \( T_{\text{node access}} \) is the worst-case latency for a frame with the earliest deadline to leave the source node, and \( T_{\text{switch access}} \) is the worst-case latency for a frame with the earliest deadline to leave the switch. The source node latency is:

\[ T_{\text{node access}} = QT_{\text{frame}} \] (4.11)

where \( Q \) is number of frames that can be stored on the NIC (Network Interface Card). In other words, we assume that we cannot interrupt the transmission of frames that have been stored on the NIC, even though they might have later deadlines than other frames.

The switch latency, before being able to forward a frame with the earliest deadline to the destination node, is

\[ T_{\text{switch access}} = \text{MAX}(2T_{\text{frame}}, QT_{\text{frame}}) \] (4.12)

The first term in the MAX expression is the maximum wait time due to the case when a frame is generated just after the transmission of a data frame has been initiated and the synchronization frame should be sent immediately after that (see Figure 4.1). The
Fig. 4.1. Worst-case latency when waiting for the completion of one data frame followed by a synchronization frame.

The second term in the MAX expression tells the same thing as for the source node latency. The MAX operator is used because the first term is included in the second term. If \( Q \) is equal to 1 (which can be expected for a switch), we get \( T_{\text{switch, access}} = 2T_{\text{frame}} \).

Below we calculate an example of a system configuration. We use a 100 Mbit/s Ethernet switch with Ethernet frames that has the data field maximized (1518 bytes in IEEE 802.3 Ethernet), while the number of frames that can stored in the NIC is 2:

\[
Q = 2 \tag{4.13}
\]

The time for the maximum sized frame to be sent is:

\[
8 \times 1518 \text{Bytes} \over 100 \text{Mbit/s} = 121 \mu s \tag{4.14}
\]

if not assuming that \( T_{\text{frame}} = 125 \mu s \). If the distance between two nodes is 100 m and \( Q = 1 \) for the switch, we have:

\[
T_{\text{link, prop, delay}} = 500 \text{ns}
\]
\[
T_{\text{switch, access}} = 2T_{\text{frame}} = 242 \mu s
\]
\[
T_{\text{node, access}} = 2T_{\text{frame}} = 242 \mu s
\]
\[
T_{\text{latency}} = 2 \times 500 \text{ns} + 242 \mu s + 242 \mu s = 485 \mu s
\]

In typical industrial applications, the \( T_{\text{latency}} \) that is added to the period in the calculation of the worst-case delay does not restrict the usability of the network.

5. Simulation analysis. The simulation analysis presented here shows the performance of using the EDF scheduling in the network. Each RT channel is randomly generated with uniformly distributed source and destination nodes. We simulated a network with a single 100 Mbit/s full-duplex Ethernet switch. Each RT channel is characterized by three parameters: the period of the data, the capacity and the deadline. In this paper, we assigned \( T_{\text{deadline}} = T_{\text{period}} \) and the deadline is split into two parts, i.e., from the source to the switch and from the switch to the destination node. For performance metric, we used the acceptance ratio, which is defined as the number of the RT channels that are feasible to use according to the EDF scheduling divided by the number of requested RT channels. In each simulation, RT channels are added one by one and checked whether accepted or not. A number of such simulations are run to get the acceptance ratio at different traffic loads, i.e., total number of requested RT channels. We have compared the acceptance ratio for different traffic and network characteristics.
In the first case (see Figure 5.1) we consider the relation between the average value of the acceptance ratio and the number of channels per node. The simulation parameters include number of nodes $N = 20$, capacity $C = 4$ and the period $T_{\text{period}} = 80$ (all the channels have the same parameters). In this case, the utilization per channel is 5% and the acceptance ratio is 100% until the number of channel per node is approximately 4. The acceptance ratio then decreases slowly when the traffic intensity per node increases. If we replace the number of request channels per node by the total number of requested channels in the network, we get the result showed.
in Figure 5.3. This result is the average values after 5000 simulation runs. The standard deviation of the acceptance ratio for the network with 20 nodes is presented in Figure 5.2.

In order to investigate the effectiveness of the network when the number of nodes is changed, we investigated network sizes from 20 to 50 nodes (see Figure 5.3). In this case, each channel has a utilization of 5%. For the link from the source to the switch we can have a maximum of 100 channels, but over the link from the switch to the destination we can only have 90 channels. The simulation has showed that the acceptance ratio is 100% when the number of requested channels is approximately 80 and the network size is 20 nodes. The acceptance ratio is increased fast when the network size is increased. This is due to the increased number of links that share the load. In a similar way we get the result in Figure 5.4 when the period and the number of nodes are constant but the value of the capacity is varied between 2 and 16. Figure 5.5 visualizes the dependency of the acceptance ratio of the network to the number of nodes and the capacity per channel when $T_{\text{period}}$ of the network is unchanged.

6. Conclusions. In this paper, we have presented a switched Ethernet based network concept supporting real-time communication with guaranteed bit rate and worst-case delay for periodic traffic. The Ethernet switch operates at 100 Mbit/s over full-duplex links, and handles non-real-time traffic as well as real-time traffic. In the proposed solution there are no modifications in the Ethernet hardware on the network interface cards, which is important to allow the network to be connected to existing Ethernet networks. Real-time communication is handled in the nodes and the switch, by software added between the network layer and the link layer. Support
for real-time communication is made by dynamically setting up real-time channels. Using Ethernet and the TCP/IP suite allows the network to be connected to the office network and to the Internet at the same time as it carries important real-time traffic in, e.g., a manufacturing industry. The simulation analysis has showed rather high average acceptance ratios up to the theoretical limit of 45% average utilization. For example, we get an acceptance ratio above 80% at 45% average utilization when each RT channel has an utilization of 5% (C = 4 and $T_{period} = 80$).

**REFERENCES**


Fig. 5.5. The acceptance ratio when the network has $T_{\text{period}}$ constant, but both the number of nodes and the capacity per channel are varied between 2 and 16.


Deadline First Scheduling in Switched Real-Time Ethernet –
Deadline Partitioning Issues and Software Implementation Experiments

Hoai Hoang, Magnus Jonsson, Anders Larsson, Rikard Olsson, and Carl Bergenhem

Abstract

This paper presents work on a switched Ethernet network extended to allow for earliest deadline first (EDF) scheduling. We show by example that asymmetric deadline partitioning between the links of a real-time channel can increase the utilization substantially, still not violating the real-time guarantees. We also report measurements on a software implementation of the switch on an ordinary PC.

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 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]. Internet communication is supported at the same time as nodes connected to the switch can be guaranteed to meet their real-time demands when they communicate with each other. This is highly appreciated by the industry since it makes remote maintenance possible, e.g., software upgrades or error diagnostics.

The rest of the paper is organized as follows. The network architecture and real-time traffic handling are presented in Section 2. In Section 3, asymmetric deadline partitioning is described and exemplified. Experiments with a software implementation of the switch are then presented in Section 4. The paper is concluded in Section 5.

2 Network architecture and traffic handling

We consider an example of a network with a full-duplex switched Ethernet and end-nodes. Both the switch and the nodes have software (RT layer) added to support guarantees for real-time traffic. All nodes are connected to the switch and nodes can communicate with each other over logical real-time channels (RT channels), each being a virtual connection between two nodes in the system.

In our network configuration, both the switch and the end-nodes use the Earliest Deadline First (EDF) algorithm for traffic control. The switch is responsible for admission control, MAC functions, frame buffering and traffic scheduling. The switch periodically sends synchronization frames to the end-nodes, at an interval, $T_{cycle}$, of ten maximum sized frames, $T_{frame}$, i.e.,

$$T_{cycle} = 10T_{frame}.$$  \hspace{1cm} (1)

In this way, every node has a uniform comprehension about global time, with the resolution of $T_{frame}$. In this paper, we assume Fast Ethernet (100 Mbit/s) with a maximum frame size of 1500 bytes.
The function of and interaction with the RT layer etc shown in Figure 1 is explained below. When an application wants to setup an RT channel, it interacts directly with the RT layer (1). The RT layer then sends a question to the RT channel management software in the switch (2). Outgoing real-time traffic from the end-node uses UDP and is put in a deadline-sorted queue in the RT layer (3). Outgoing non-real-time traffic from the end-node typically uses TCP and is put in an FCFS-sorted (First Come First Serve) queue in the RT layer (4). In the same way, there are two different output queues for each port on the switch too (5).

An RT channel with index \(i\) is characterized by:

\[
\{ T_{\text{period},i}, C_i, T_{\text{deadline},i} \} \]  

(2)

where \(T_{\text{period},i}\) is the period of data, \(C_i\) is the amount of data per period, and \(T_{\text{deadline},i}\) is the relative deadline used for the end-to-end EDF scheduling. Both \(T_{\text{period},i}, C_i\), and \(T_{\text{deadline},i}\) are expressed as the number of maximal sized frames, i.e., the number of \(T_{\text{frame}}\).

When a node wants to establish an RT channel, it sends a request frame (ReqF) with source and destination node MAC and IP addresses and \(\{ T_{\text{period},i}, C_i, T_{\text{deadline},i} \}\) to the switch. A connection ID to distinguish between several possible connection requests is also added. When receiving a ReqF, the switch will calculate the feasibility of the traffic schedule between the requesting node and the switch and between the switch and the destination node. The ReqF is then forwarded to the destination node, after adding a network unique ID in the RT channel ID field. The destination node responds with a response frame (ResF) to the switch telling whether the establishment is accepted or not. The switch will then, after taking notation of the response, forward the ResF to the source node.

The RT layer in an end-node prepares outgoing real-time IP datagrams by changing the IP header before letting the Ethernet layers sending it (see Figure 2). The IP source address and the 16 most significant bits of the IP destination address, 48 bits together, are set to the absolute deadline of the frame. A 48 bit absolute deadline with a resolution of \(T_{\text{frame}} = 125 \mu s\), gives a “life time” longer than one thousand years. The 16 least significant bits of the IP destination are set to the RT channel ID for the RT channel to which the frame belongs. The MAC destination address is set to a special address that all nodes use for real-time traffic, while the Type of Service (ToS) field is always set to value 255.

The switch exchanges the source and destination IP addresses and the MAC destination address of an incoming real-time frame with the correct ones (as stored in the switch when the RT channel was established) for delivery to the final destination.

## 3 Asymmetric Deadline Partitioning

Below, we will show by example that the possible amount of guaranteed real-time traffic can be increased by partitioning the deadline asymmetrically between the different links crossed by an RT channel. We compare the asymmetric partitioning with a symmetric partitioning for a master-slave situation.

We assume that master node \(M_1\) is responsible for five slave nodes \(S_1 \cdot S_5\). The master node has one RT channel per slave node via the switch. The real-time guarantee from \(M_1\) to \(S_i\) is upheld by RT channel \(i\), \(1 \leq i \leq 5\). For the deadline scheduling we assume:

\[
T_{\text{deadline},i} = T_{\text{period},i} = T_{D1,i} + T_{D2,i} \]  

(3)

for each RT channel \(i\), where \(T_{D1,i}\) is the relative deadline for real-time traffic from the master node to the switch and \(T_{D2,i}\) is the relative deadline from the switch to the destination node. In the same way, let us assume that \(N_1\) and \(N_2\) represent the total number of RT channels on the links a specific RT channel crosses, i.e. the load of the links to and from the switch. For simplicity, all channels are assumed having the same characteristics and being unidirectional with the master node as the source node.

With an asymmetric deadline partitioning, \(T_{\text{deadline}}\) is partitioned so the local deadline for a link of the end-to-end path is weighted according to the load of the link divided by the sum of the loads across the whole path. For our example, this gives:

\[
T_{D1,i} = \frac{N_1}{N_1 + N_2} T_{\text{period},i} \quad T_{D2,i} = \frac{5}{6} T_{\text{period},i} \]  

(4)
\[ T_{D2,j} = \frac{N_2 \cdot T_{period,j}}{N_1 + N_2} = \frac{1}{6} T_{period,j} \] (5)

This is a simple partitioning method to show the opportunities with deadline partitioning. Our future work includes looking at partitioning method that can handle more complex traffic patterns and dynamic channel setup as the network is designed for.

According to the basic EDF theory [7], the utilization of real-time traffic is defined as

\[ U = \sum \frac{C_i}{T_{period,j}}. \] (6)

One is assured that all deadlines are met if the utilization of real-time traffic does not exceed a certain level, \( U_{\text{max}} \):

\[ U = \sum \frac{C_i}{T_{period,j}} \leq U_{\text{max}} \] (7)

This guarantee holds for deadline scheduling of traffic when the deadline for a specific link is equal to the period multiplied by a constant \( k \leq 1 \), for all RT channels traversing the link in the same direction. When scheduling a channel with 100 % theoretical utilization, \( U_{\text{max}} = k \). For deadline scheduling of traffic with arbitrary deadlines, see [8] or subsequent work (e.g. [9]). We define \( U_{\text{max}} \) as the maximum utilization for the link from the master node to the switch and \( U_{\text{max}}^2 \) as the maximum utilization from the switch to the slave node. In the theoretical case \( U_{\text{max}} \) is 100 %, but when using the network proposed in this paper the worst-case maximum utilization for the link from the switch and to the slave node is reduced from 100 % to 90 % due to having 10 % bandwidth for the network control. In symmetric deadline partitioning [6], we have \( U_{\text{max}}^2 = 45 \% \) and \( U_{\text{max}}^1 = 50 \% \). When using asymmetric deadline partitioning with the weights from Equation 4 and 5 \((k_1 = 5/6 \text{ and } k_2 = 1/6, \text{respectively})\), we get the following maximum utilizations instead:

\[ U_{\text{max}}^1 = \frac{5}{6} \cdot 100\% = 83\% \] (8)

\[ U_{\text{max}}^2 = \frac{1}{6} \cdot 90\% = 15\% \] (9)

We denote \( C_{\text{max}}^1 \) and \( C_{\text{max}}^2 \) as the maximum capacity (transmission time per period) per channel for the first and the second link traversed by an RT channel, respectively. When assuming the same period, \( T_{\text{period}} \), and deadline, \( T_{\text{deadline}} \), for all RT channels, we have

\[ \frac{5 C_{\text{max}}^1}{T_{\text{period}}} = U_{\text{max}}^1 \Rightarrow C_{\text{max}}^1 = \frac{U_{\text{max}}^1 T_{\text{period}}}{5} \] (10)

for the master link and

\[ \frac{C_{\text{max}}^2}{T_{\text{period}}} = U_{\text{max}}^2 \Rightarrow C_{\text{max}}^2 = \frac{U_{\text{max}}^2 T_{\text{period}}}{5} \] (11)

for a slave link. For example, let us assume that \( T_{\text{period}} = T_{\text{deadline}} = 24 \text{ ms} \) (12)

According to Equations 4, 5, 8, and 9, we then have:

\[ T_{D1} = 20 \text{ ms} \]

\[ T_{D2} = 4 \text{ ms} \]

\[ C_{\text{max}}^1 = \frac{U_{\text{max}}^1 T_{\text{period}}}{5} = 4 \text{ ms} \] (13)

\[ C_{\text{max}}^2 = \frac{U_{\text{max}}^2 T_{\text{period}}}{5} = 3.6 \text{ ms} \]

The second link is the bottleneck because \( C_{\text{max}}^2 < C_{\text{max}}^1 \). With \( C = C_{\text{max}}^2 = 3.6 \text{ ms} \) for all RT channels we get a utilization of \( U = C / T_{\text{period}} = 0.15 \) on each slave link and \( U = 5C / T_{\text{period}} = 0.75 \) on the master link.

In the asymmetric case, we have 75 % utilization on the master link compared with 50 % in the symmetric case. We still guarantee worst-case delay for real-time traffic. The analysis given above holds for the opposite direction (from each slave and to the master) and for other, not overlapping, master slave groups in the network too.

### 4 Software Implementation

We have implemented the switch using a PC with a 200 MHz Pentium processor, some network interface cards and LINUX 2.4.2. The two most important parts in the switch regarding the real-time communication are the RT-layer and the RT channel management (see Figure 1). The RT channel management is responsible for approving RT channels requested by nodes in the system. Information about the approved RT channels is made available to the RT layer, which handles the actual scheduling and forwarding of data frames. If the load on the switch becomes too high it simply discards non-RT frames.

The tests were performed in a network with the switch and three nodes, two sending and one receiving. All nodes were equipped with 100 Mbit/s full-duplex Ethernet cards. The PC acting as switch operates at 200MHz. In the first test, the frame size, \( N_{\text{byte}} \), is set to 1000 bytes, including headers. In the receiving node, a timer, \( t_{\text{measure}} \), starts at the arrival of the first RT frame and is stopped when the last frame is detected. During this time all received frames are registered either as RT frames or non-RT frames.

The corresponding data rates for received data, \( R_{RT} \) and \( R_{NRT} \), are calculated as:

\[ R_{RT} = \frac{8 N_{\text{byte}}}{t_{\text{measure}}} \] (14)
The measured data rates in the receiving node are plotted against the frequency of the injected periodic traffic (see Figure 3). In this test, both RT traffic and non-RT traffic were injected with the frequency indicated in the figure. The result shows that the switch prioritizes the RT frames. It also shows that the breakpoint, where the switch begins to discard non-RT frames, is when the total traffic load reaches 96 Mbit/s, i.e., when the total traffic-load approaches the maximum capacity of the switch, the switch starts to discard non-RT traffic. As the traffic load increases, the RT-Switch is finally forced to discard RT traffic as well. This happens when the period of the traffic from the sending application is about 0.1 ms. This situation can only arise when the system runs without any channel management.

Additional tests were made including the two extremes: (i) Only maximum sized frames (1518 bytes) are transmitted. This gives a maximum throughput for the switch, measured to 96 Mbit/s, which is near wire speed. (ii) Only minimum sized frames (64 bytes) are transmitted. This gives the lowest throughput, measured to 18 Mbit/s. This case also gives us a maximum switching capacity of 14 400 frames/s.

The tests that have been performed shows that it is possible to build an Ethernet switch with deadline scheduling by using only standard components. This gives an indication on the possibility of implementing real-time capabilities in an Ethernet network. One can get a good feeling from the measurements about the amount of real-time traffic and number of ports that can be supported if, for example, a processor should assist a switch chip by handling deadline scheduling in software.

5 Conclusions

In this paper, we have presented an Ethernet network with support for real-time traffic by deadline scheduling. We have showed by example that the performance can benefit a lot from asymmetric deadline scheduling. An increase in utilization from 50 % to 75 % on the outgoing link from the master node (the bottleneck) is observed, still not violating the real-time guarantees.

From a software-implemented switch, we have showed experimental results. The measurements showed that the switch bottlenecks are 96 Mbit/s (measured for maximum-sized frames) and 14 400 frames/s (measured for minimum-sized frames). From these measurements one can get a good feeling of the amount of real-time traffic and number of ports that can be supported by an Ethernet switch that is fully or partly implemented in software.

References

conjunction with International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale, FL, USA, April 15-16, 2002.


Switched Real-Time Ethernet in Industrial Applications - Deadline Partitioning

Hoai Hoang and Magnus Jonsson.


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 from/to the switch is proposed (ADPS). The scheme is compared with the simpler approach of dividing the deadlines in two (SDPS). The results of several software simulations of setting up RT channels over a full-duplex switched Ethernet network are presented. The simulations show that the ADPS performs better than the SDPS 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) [444]. 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 and real-time traffic handling are presented in Section 2. In Section 3, a feasibility analysis is introduced. In section 4, two deadline partitioning schemes (DPS) are presented: the Symmetric Deadline Partitioning Scheme (SDPS) and the Asymmetric Deadline Partitioning Scheme (ADPS). In section 5, practical considerations are discussed. In section 6, a simulation is presented. The paper is concluded in Section 7.

2 Network architecture and traffic handling

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

\[ SS = \{ N, K \} \]  \hspace{1cm} (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:

\[ \{ \text{Source}_i, \text{Destination}_i, P_i, C_i, d_i \} \]  \hspace{1cm} (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 \( T_{\text{frame}} \). \( \text{Source}_i \) and \( \text{Destination}_i \) are Ethernet MAC addresses for channel \( i \)'s source and destination nodes respectively. It is evident that both \( \text{Source}_i \) and \( \text{Destination}_i \) are parts of the system, i.e. \( \text{Source}_i, \in N; \text{Destination}_i, \in N \).

In our network configuration, both the switch and the end-nodes use the Earliest Deadline First (EDF) algorithm for traffic control. The switch is responsible for admission control, MAC functions, frame buffering and traffic scheduling. The switch periodically sends synchronization frames to the end-nodes at an interval, \( T_{\text{cycle}} \), of ten maximum sized frames, \( T_{\text{frame}} \), i.e.,

\[ T_{\text{cycle}} = 10T_{\text{frame}}. \]  \hspace{1cm} (3)

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

**Figure 2: Layers and output queues.**

**Figure 3: Data frame sent over an RT channel.**

In this way, every node has a uniform comprehension about global time, with the resolution of \( T_{\text{frame}} \). In this paper, we assume Fast Ethernet (100 Mbit/s) with a maximum frame size of 12 144 bits which, with some extra time for timing uncertainties and for simplicity, gives \( T_{\text{frame}} = 125 \mu s \), which just happens to match the time resolution of many telecommunication systems.

The function of and interaction with the RT layer etc shown in Figure 2 is explained below. When an application wants to setup an RT channel, it interacts directly with the RT layer (1). The RT layer then sends a request to the RT channel management software in the switch (2). Outgoing real-time traffic from the end-node uses UDP and is put in a deadline-sorted queue in the RT layer (3). Outgoing non-real-time traffic from the end-node typically uses TCP and is put in an FCFS-sorted queue in the RT layer (4). In the same way, there are two different output queues for each port on the switch too (5).

When a node wants to establish an RT channel, it sends a request frame (ReqF) with source and destination node MAC and IP addresses and \( \{ P_i, C_i, d_i \} \) to the switch. A connection ID to distinguish between several possible connection requests is also added. When receiving a ReqF, the switch will determine if the channel can be added to the system, i.e. whether the real-time guarantees can be upheld. If the switch determines the channel to be acceptable, the ReqF is forwarded to the destination node, after adding a network unique ID in the RT channel ID field. The destination node responds with a response frame (ResF) to the switch telling whether the establishment is accepted or not. The switch will then, after taking notation of the response, take necessary steps towards updating the system.

The RT layer in an end-node prepares outgoing real-time IP datagrams by changing the IP header before letting the Ethernet layers sending it (see Figure 3). The IP source address and the 16 most significant bits of the IP destination address, 48 bits together, are set to the absolute deadline of the frame. A 48 bit absolute deadline with a resolution of \( T_{\text{frame}} = 125 \mu s \), gives a “life time” longer than one thousand years. The 16 least significant bits of the IP destination are set to the RT channel ID for the RT channel to which the frame belongs. The MAC destination address is set to a special address that all nodes use for real-time traffic, while the Type of Service (ToS) field is always set to value 255. One (or both) of these fields can be used to filter out real-time frames for correct service treatment.

The switch exchanges the source and destination IP addresses and the MAC destination address of an incoming real-time frame with the correct ones (as stored in the
switch when the RT channel was established) for delivery to the final destination.

### 3 Feasibility Analysis

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 EtherDaemon software handles the scheduling, but otherwise the same is true.

In our system we use EDF as the scheduling algorithm, for both the switch and the end-nodes. Liu and Layland [7] showed the advantage of using EDF, as it is the optimal uniprocessor scheduling algorithm. The one big drawback of EDF is the complexity of the feasibility test, compared with other scheduling algorithms such as the rate-monotonic[10]. Before we describe the feasibility test, we make the following definitions:

- **"The Utilization factor"**
  According to basic EDF theory [7] the utilization of periodic real-time traffic is defined as
  \[
  U = \frac{\sum C_i}{P_i}. \tag{4}
  \]

- **"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
  \[
  h(n, t) = \sum_{i \in K_n} \left(1 + \frac{t - d_i}{P_i}\right) C_i \tag{5}
  \]
  where $K_n$ is the set of tasks running on link $n$

- 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 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 or equal to $t$**

The second constraint, in the form given above, does not lend itself out particularly well to computation. It is shown in [11] how to reduce the time and memory complexity of the second constraint check. If $h(n, t) \leq t$ in the first busy period of the hyperperiod in the supposed schedule to come, then $h(n, t) \leq t$ for all $t$. The following upperbound would therefore be an improvement of the algorithm above:

\[
 t, \text{ such that } 1 \leq t \leq \text{BusyPeriod}(n) \tag{6}
\]

where BusyPeriod($n$) is the first BusyPeriod in the schedule at the start of the hyper-period. Further more, one needs not check every integer from the first timeslot, but only the integers $t$ where

\[
 t \in \bigcup_{i=1}^{\infty} \{m T_i + d_i : m = 0, 1, \ldots\} \tag{7}
\]

**Deadline Partitioning**

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.

For this approach to work, 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:

\[
 T_{iu} = \{\text{Source}_i, C_i, d_{i u}, P_i\} \tag{8}
\]

\[
 T_{id} = \{\text{Destination}_i, C_i, d_{i d}, P_i\} \tag{9}
\]

where Source$_i$, Destination$_i$, $C_i$ and $P_i$ are the parameters of the channel $i$. $d_{i u}$ and $d_{i d}$ 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, $d_{i u}$ and $d_{i d}$ 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 2 parts: $d_{iu}$ and $d_{id}$ where

\[
 d_i = d_{iu} + d_{id} \tag{10}
\]
\[ d_{in}, d_{id} \geq C_i \quad ; \quad (if \; d_i \geq 2C_i) \quad (11) \]

The condition (10) must be upheld, because otherwise the channel as a whole will be difficult. If one divides a task into separate smaller tasks, it should follow that if the original task had a deadline, then the sum of the subtasks' deadlines must equal this larger deadline. The condition (11) should be upheld because otherwise the partitioning will automatically yield a non-EDF-feasible situation. The deadline cannot be allowed to be shorter than the capacity because the capacity is the WCET of the supposed tasks. We can also assure ourselves that if \( d_i < 2C_i \) then the channel cannot, by definition, be EDF-feasible.

4 Deadline Partitioning Schemes (DPS)

We make the following definition:

- A Deadline-Partitioning Scheme (DPS) is a function that maps the deadlines \( d_i \) of all the channels in the system into two deadlines \( d_{in}, d_{id} \) such that the condition (Equation 10) 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 \( d_{in} \) 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

\[ \text{dim} = \text{size}(K) \quad (12) \]

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, \{ \( d_{in}, d_{id} \) \}. We now take steps to change this function. First, we normalize with the original deadline, \( d_i \) for each corresponding pair. Because of (10) this would mean that we would have pairs, ranging from 0 to 1. The output would look like:

\[ U_{part,i} = d_{in} / d_i \quad (13) \]

\[ D_{part,i} = d_{id} / d_i \]

where \( U_{part,i} \) and \( D_{part,i} \) are the factors of \( d_i \) to get \( d_{in} \) and \( d_{id} \) respectively. But because of (10) we conclude that:

\[ U_{part,i} = 1 - D_{part,i} \quad (14) \]

This means that both \( U_{part,i} \) and \( D_{part,i} \) contain all the information by themselves. Only one is sufficient, and thus we have:

\[ U_{part} = \text{DPS}(SS) \quad (15) \]

\[ U_{part} \] is a dim dimensional vector, of elements \( (0 < U_{part} < 1) \). The number of different possible DPSs is infinite, and in this paper two different ones will be examined: Symmetric (SDPS) and Asymmetric (ADPS).

4.1 DPS Performance Measure

A channel that is requested by the system can fail any of the two constraints posed by the feasibility test. Firstly, if the first constraint test fails, we note that whatever generous deadline the task-set has, it will not affect the outcome. The first constraint is a test of utilization only, and the utilization is invariant of the deadline, which we note from (Equation 4). On the other hand, if it is the second constraint that fails, the choice of another DPS could have made the channel feasible. It is clear that we can note a difference of performance, in the different DPSs we can choose to divide the deadlines. We define an optimal DPS as follows:

- A DPS is optimal if it allows for a maximum number of channels to be accepted.

When a task set fails the second constraint, it does so because one or more values of the workload function \( h(n,t) \) exceeds the value of \( t \) \( (h(n,t) > t) \). The test \( h(n,t) \leq t \) tells us if the task set is EDF schedulable but it lacks any levels of magnitude other than the Boolean true or false. In an effort to get a significance measure, that is, a measure of how well the task set passed the second constraint, we do the following definition:

- The "workload margin" (wm) for a link is defined as the lowest value of the expression:

\[ t - h(n,t) \quad t \geq \text{min}(d_i) \quad (16) \]

where \( \text{min}(d_i) \) is the minimum of the deadlines of the tasks running on the link. The "workload margin" is negative for the links that are not feasible. For links that are feasible it is positive, and gives us the minimum differences between \( h(n,t) \) and \( t \).

4.2 Symmetric Deadline Partitioning

In the original paper [6], it was proposed to partition the deadline of the channels into two equal parts, i.e. to split it in half. Following the notation introduced above, this would imply that

\[ d_{in} = d_{id} = d_i / 2 \quad (17) \]

\[ U_{part,i} = D_{part,i} = 1 / 2 \quad (18) \]

We define this approach as a Symmetric DPS (SDPS). It is easily seen that condition (10) is upheld under this function. We can also note that the SDPS only depends on the size(K) of the system state SS. In the view of DPSs as vector fields this means that the SDPS is a vector of size(K) dimension, with each element constant, equal to 0.5. Obviously, as the SDPS doesn’t take into consideration what the system looks like, we should be able to propose a better DPS.

4.3 Asymmetric Deadline Partitioning Scheme

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:

$$U_{pari} = \frac{LL(Source_i)}{LL(Source_i) + LL(Destination_i)} \quad (19)$$

$$D_{pari} = \frac{LL(Destination_i)}{LL(Source_i) + LL(Destination_i)} \quad (20)$$

5 Practical Considerations

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. We will address three issues: (a) memory and time complexity, (b) the need for node-feedback, (c) the need for synchronous deadline updating.

The most obvious consideration is the first item (a), computational speed and memory requirements. As we go from SDPS to 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 implementing them (The calculations are, however, only made when a new channel is requested).

The second and third considerations deal with whether the DPS is actually implementable or not. A DPS, as we recall, has full freedom to divide the deadlines of the entire system - in reality the switch may not have this freedom. After the DPS has operated on the deadlines, the feasibility test is performed on the tasks we create from the partitioned deadlines. It is the switch that does the actual scheduling of packets for the download links, but for the upload parts, it is the various nodes in the system that performs scheduling. For the SDPS we can note a definite advantage, and that is the following. For the actual scheduling of packets, the nodes need only to look at the channels relative deadline. They do not need an updated version of $d_{up}$ for each channel, because under SDPS, the system always generates the same $U_{pari}$, namely 0.5. A node can get relative deadlines of all its tasks by multiplying the $d_i$ of the corresponding channel with 0.5. For ADPS, we need to be certain that all nodes actually have the partitioned deadline for each channel that we assume when doing the feasibility test (node-feedback). We also need to have guarantees that the nodes update their relative deadlines at exactly the same timeslot as the switch (synchronous deadline updating). We cannot have a system where condition (10) does not hold at all times. The following steps describe the operation of adding a channel to the system:

1. Create a system state, with the new channel added.
2. Apply DPS to the system state to get deadlines.
3. Test feasibility of all task-sets of the links.
4. Make sure that all nodes are updated with the new deadlines that the DPS provided.
5. Make sure that all nodes update their relative deadlines synchronous with the switch.
6. Permit traffic over the new channel.

No other channel can be tested while the switch software is occupied with any of these steps. The items we want to put emphasis on are step 4 and 5. How do we let the nodes know of the update that they need to do, and how do we synchronize it with the Switch? Because the nodes are assumed to be independent of each other, save for the connection to the switch, the only way of providing the nodes of the update information would be to somehow send it over the links.

Unfortunately we cannot simply send a frame over the links as soon as we want to update the nodes, as this would destroy the guarantees of all the other channels. However, we do have the sync frames that need to be sent every $T_{cycle}$ (3) anyway. Depending on how much free space we have in the synchronization frame, a finite number of tasks in the nodes can be updated each $T_{cycle}$. To assure that the nodes actually update their upload tasks at the same time as the switch updates its download tasks, a pre-determined time lag of a few time slots should be implemented for both the node and the switch, counting from when the node receives the synchronization frame with the updating information.

6 Simulation Analysis

The simulation analysis presented here shows the performance difference between using the EDF scheduling with Symmetric and Asymmetric deadline partitioning in the network. 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). As described in Section 3, the feasibility test of a link is done in two steps, the first constrain and the second constrain.

In Figure 4, the number of accepted channel is showed for a network configuration of 10 master node and 50 slave nodes. To compare the result between symmetric and asymmetric deadline partitioning, in this simulation, every requested channel have the same parameters: $C=3$, $P=100$, $d=40$. The result proved that we get much better result with asymmetric deadline partitioning. With many requested channels the number of accepted channels is about 110, while there are only 60 accepted channel with symmetric deadline partitioning. It means that with master-slave
traffic communication, EDF scheduling with asymmetric deadline partitioning is much better than symmetric deadline partitioning.

Figure 5 (for symmetric deadline partitioning) and Figure 6 (for asymmetric deadline partitioning show 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 generate in a range from 80 to 120, while deadlines range from 30 to 50. We can see the different result when each master node has responsibility for 3, 4 or 5 slave node. In these cases, all the deadline $d_i$ and period $P_i$ are generate randomly. The acceptance ratio is still better when we use ADPS than DPS.

7 Conclusions

In this paper, we have presented an Ethernet network with support for real-time traffic by EDF scheduling. Two Deadline Partitioning Schemes have been proposed, and an effort to mathematically model the workings of DPSs has been described. Two DPSs, SDPS and ADPS have been compared. Clearly, As we go from SDPS to ADPS, we gain in DPS performance. 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.

References
