On Hierarchical Server-based Communication with Switched Ethernet

Rui Santos, Paulo Pedreiras
IEETA / University of Aveiro
Aveiro, Portugal
{rsantos, pbrp}@ua.pt

Farahnaz Yekeh, Thomas Nolte
MRTC / Mälardalen University
Västerås, Sweden
thomas.nolte@mdh.se

Luis Almeida
University of Porto
Porto, Portugal
lda@fe.up.pt

Abstract

Ethernet is becoming a common network technology for industrial and factory automation systems and, in recent years, a big effort has been made in enabling real-time communications using Ethernet technology. Many of these systems are complex, extend over relatively large places and/or integrate a significant number of nodes, thus requiring the use of multiple switches (hop). In this paper we look into the usage of Flexible Time-Triggered (FTT) enabled Ethernet switches in this class of systems, more specifically using the recently proposed server-based scheduling mechanism supported by this protocol. The paper proposes and validates a resource reservation protocol, presents a method for computing the end-to-end deadlines and discusses possible strategies for the deadline partitioning. [1]

1 Introduction

Over the past years, Ethernet has moved in the direction of becoming the de-facto standard networking technology for industrial and factory automation systems. However, when comparing to the original applications of the Ethernet technology, where throughput was the main driving factor, industrial and factory automation systems often have real-time requirements. These requirements are not straightforwardly guaranteed by standard Ethernet, which led to the development of several protocols, commonly known as Real-Time Ethernet (RTE), able to provide adequate temporal behaviour for that class of applications.

At the same time the complexity, amount and heterogeneity of the data exchanged between node in these applications is also increasing. Namely, many systems nowadays integrate periodic and sporadic message flows (flows, for short) with sizes ranging from a few bits (simple I/Os) to several KB (e.g. multimedia sensors such as cameras) and diverse timeliness constraints. Furthermore, the configuration of these flows is often dynamic in nature, thus flows can be added, changed or removed during run-time. However, the support of these features in a network is hampered, since the network components usually do not have 1) efficient mechanisms to differentiate flows, 2) support of temporal isolation among flows, and 3) efficient sharing of the network resource.

In order to solve the limitations enumerated above, the integration of *flow management* with hierarchical server-based scheduling (originally inherent in CPU scheduling) has been tested in network domain, specifically in the context of single switch architectures [5] [6]. In this case, hierarchically arranged servers can manage a flow or a group of flows, preforming contention, i.e., ensuring that the flows do not use more resources (bandwidth) than what has been negotiated while at the same time providing a guaranteed level of negotiated QoS. However, a single switch architecture is not naturally scalable and it does not respond to traditional RTE network requirements, such as great coverage and composition of different sub-systems (corresponding sub-networks). Therefore, a number of issues must be looked into, stretching from timing analysis of message transmissions across a hierarchical architecture of multiple switches (hop) to simple and efficient bandwidth reservation mechanisms.

Therefore, in this work-in-progress paper we present and discuss a preliminary protocol for bandwidth reservation, allowing for the transmission of traffic flows with real-time guarantees using server-based scheduling techniques implemented in the switches. The presented solution aims to improve the process to handle concurrent requests, always assigning available resources to requests intended for higher priority flows. Moreover, timing analysis is introduced and discussed. As a test platform we have been using FTT-enabled Ethernet switches, since new features are easily implementable on that platform.

The remainder of the paper is organized as follows. Section 2 presents overview of related work, and Section 3 outlines the system model. Section 4 discusses a preliminary protocol for bandwidth reservation, followed by Section 5 that addresses the timing analysis and Section 6 that concludes the paper.

[1] This work was partially supported by the Swedish Foundation for Strategic Research (SSF), the Swedish Research Council, the iLAND project, call 2008-1 of the EU ARTEMIS JU Programme, by the European Community through the ICT NoE 214373 ArtistDesign and by the Portuguese Government through the FCT project HaRTES - PTDC/EEA-ACR/73307/2006 and Ph.D. grant - SFRH/BD/32814/2006.

978-1-4244-6850-8/10/$26.00 ©2010 IEEE

2 Related work

EtheReal [7] is a protocol which uses a switch that provides guaranteed-bandwidth network services without the need of cooperative end nodes. This way, all real-time capabilities of the network are implemented through a combination of user-level libraries and switch hadware/sofware. To guarantee differentiated QoS of real-time connections, explicit resources need to be reserved and associated to these connections. During the connection setup a reservation request message is sent to the network and an admission control in each switch checks if the new real-time connection interferes in the guarantees of the real-time connections already admitted. If the resources requested/required are available, the connection is admitted, resources are reserved, and the request is forwarded to the next hop – otherwise it is rejected and the reservation failure message is sent back to the requesting node, releasing the reserved resources along the way.

Hoang et al. [1] developed a technique that supports a mix of Real-Time (RT) and non-real-time traffic coexisting in a switch-based Ethernet network. The RT traffic is scheduled according to the Earliest-Deadline-First (EDF) policy, and its timeliness is guaranteed by means of adequate online admission control. The creation of the RT channel consists of sending requests and getting responses between the source, the switch and the destination. If all of these devices agree on establishing an RT channel, after checking their feasibility tests, the RT channel will be established.

Zhang et al. [8] address hard real-time communication over multi-hop switched Ethernet without any modification to existing Ethernet hardware. A dual-level traffic smoothing mechanism realizes hard real-time communication over multi-hop commodity switched Ethernet. The message transmission delays at each networking component are firstly analyzed using the network calculus theories over multi-hop switched Ethernet.

Summing up, current real-time communication approaches in switched Ethernet, and respective resource reservation protocols, are topically based on a single switch architecture, assuming the existence of cooperative nodes hence not allowing for legacy nodes. Moreover, other approaches do not provide an efficient mechanism to share the bandwidth and they do not allow for composability in the time domain.

3 System Model

In this paper we look into the usage of Flexible Time-Triggered (FTT) enabled Ethernet switches [4] in multi-hop architectures. The FTT-enabled Ethernet switch is a novel real-time Ethernet switch created in the scope of the FTT paradigm. The communication is organized in an infinite succession of Elementary Cycles (ECs). These ECs are divided in two windows – the synchronous window and the asynchronous window. The former window is used for synchronous communications that are coordinated by a *master* implemented inside the switch. Therefore, to control which messages that are to be transmitted in the synchronous window the switch (master) broadcasts a Trigger Message (TM), in the beginning of each EC, to all nodes (slaves) containing the schedule for the corresponding (upcoming) synchronous window. The latter window, the asynchronous window, is used for asynchronous communications (flows) automatically triggered by the nodes and managed during runtime by the *servers* resident inside the switch. Each flow, or a group of flows, requires a suitable admission control by the switch that allocates/configures a server to handle it.

Summing up, the FTT-enabled Ethernet Switch provides the following features: 1) online admission control, dynamic QoS management and arbitrary traffic scheduling policies; 2) high system integrity with unauthorized real-time messages being eliminated at the switch input ports; 3) asynchronous traffic autonomously triggered by the nodes, with arbitrary arrival patterns; 4) high configurability: fully synchronous mode, adjustable mixed synchronous/asynchronous mode and fully asynchronous mode; 5) a standard node can take advantage of the real-time services simply negotiating with the switch the creation of a server, i.e., a virtual channel (the negotiation can even be done by a third party node); 6) a standard node can readily transmit non-real-time traffic using a background server thus not interfering with the real-time traffic.

Therefore, we rely on an architecture of hierarchically composed FTT-enabled Ethernet switches, as depicted in Figure 1. Inside each switch, each flow of messages is managed by one corresponding server. Hence, a switch hosts a number of servers equal to the number of flows going through the switch. All switch internal servers are then scheduled by the switch during runtime. Hence, whenever a flow has to go through several switches in the network, corresponding servers must be present in each switch. As a result, when flows are dynamically managed with flows added and removed during runtime, servers must be added or removed in all switches that the flow pass through. From an application point of view, this framework is easily applicable. The RT channel can be allocated through a simple protocol interface application or by third party nodes. Moreover, the legacy nodes can communicate without guarantees, using background servers available in the FTT-enabled Ethernet switches and without interfere with allocated RT channels.

In the following section we present our work on the development of a protocol for management of this dynamic management of servers in a hierarchical architecture of FTT-enabled Ethernet switches.

4 Protocol

This section presents the protocol that reserves the resources (bandwidth), i.e., creates/reserves the servers, in all network components (switches) that are in the path between two distinct nodes of the network. A suitable reservation mechanism cre-

ates a real-time channel that allows to transmit a flow or a group of flows with a minimum QoS level required and defined in the reservation process. Basically, the protocol is divided in two steps, the first step pre-reserves the resources in all switches in the path between the source and the destination node, running an admission control test in each of them. The second step performs the reverse path and allocates or releases the pre-reserved resources depending on if the first step was successful or not.

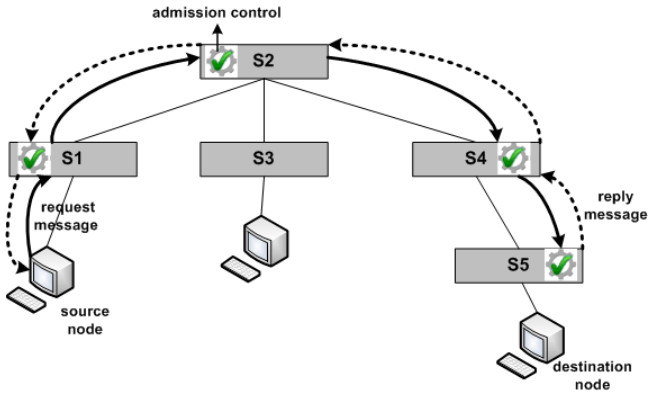

Figure 1: Resource reservation protocol.

Particularly, the application source node that wants to initiate a flow transmission with real-time guarantees has to create a RT channel between itself and the destination node. For that, it should send a connection setup request message to the network (step 1) containing 1) the RT channel identifier, 2) the flows identifier and 3) the required QoS parameters. Then each switch in the path executes the feasibility analysis in its admission control, by checking the 1) requested memory needed for transmitting the desired data, 2) available bandwidth for the requested channel and 3) if the required channel verifies the end-to-end transmission delay at this point. If the result of a corresponding feasibility analysis becomes true, the RT channel is accepted, resources (servers) are pre-reserved and the connection setup *request message* is forwarded to the next hop and so on, until it reaches the final switch along the path to the destination. If the connection request passes the admission control criteria of all the switches along the path, the last switch sends a reply *success message* to the source node. This message is sent in the reverse way, allocating the pre-reserved resources in each switch along the path. This way, after the reception of the *success message* the application source node is allowed to start the transmission of its flows. On the other hand, if the admission control checking fails at a switch, the pre-reservation step stops and the switch sends to the source node a reply *reject message* that will release all the pre-reserved resources. This way, the RT channel can not be created.

4.1 Protocol Model

In order to validate the correctness of the protocol described above, the Uppaal [2] modeling tool was used. Three state machines have been considered which model three existing elements in the network, namely senders, switches and receivers.

The trigger occurs when the source node transmits an RT channel *request message* to the first switch, then goes to sending request state and waits to receive a *success* or a *reject message*. When a switch gets a request, after checking the feasibility/schedulability and according to its result (true or false), it goes to *Accept_Forward_Request* or *Reject_Request*. In case of accepting the request and being the last switch, it goes to *Back_Success_Allocate* and sends success to its previous node. If it is not the last switch, it forwards the request to the next switch by sending a *sw_request*. In case of rejection, the backwarding reject message continues until it reaches the sender. The model of the reservation protocol is shown in Figure 2. Using the Uppaal model checker, several features of the protocol were proven, such as its correctness, absence of deadlocks and its guaranteed end in one of two possible states (accept/foward the request or reject the request).

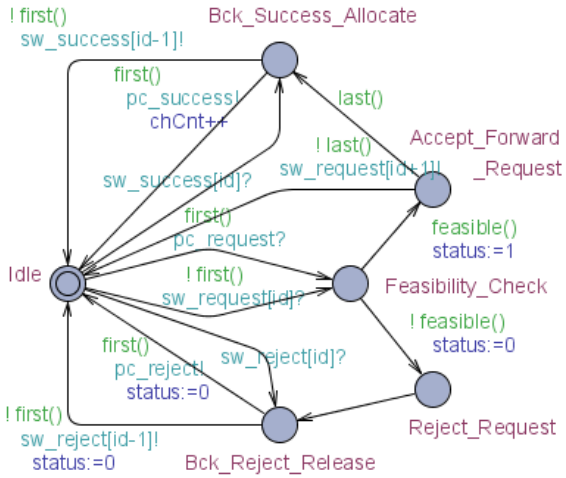

Figure 2: Switch Model in Uppaal.

5 Timing Analysis

In order to properly configure the servers in the switches such that they can guarantee timing requirements to be met, we have looked into timing analysis of messages sent in the hierarchical architecture of FTT-enabled Ethernet switches. Hence, in this section we address issues related with the timing analysis considering the hierarchical structure of switches presented above.

Consider a request of an RT channel (RTC_i) characterized as in (1), where C_i is the maximum transmission time, $Tmit_i$ represents the minimum interarrival time, D_i is the associated deadline or end-to-end timing requirement, Src_i and Dst_i is the

respective source and destination node and $SP_i = \left\{ s_i^1, .., s_i^{k_i} \right\}$ is the set of switches in the path between the source and the destination node. Hence, an RT channel RTC_i is characterized as follows

$$RTC_i = (C_i, Tmit_i, D_i, Src_i, Dst_i, SP_i), i = 1..N_{RTC} \quad (1)$$

When establishing a flow of messages stretching over a set of switches, the summation of the worst case transmission time $wctt_{s_i^{j_i}}$ associated to RTC_i in each switch $s_i^{j_i}$ should be less than or equal to the corresponding end-to-end deadline D_i. Hence, the following must hold

$$\sum_{j=1}^{k} wctt_{s_i^{j_i}} \leq D_i \quad (2)$$

and the following question should be considered: how does each switch $s_i^{j_i}$ that will handle a part of the RT channel (RTC_i) define its corresponding server parameters (capacity and period) such that (2) is respected? There are several answers to this question, resulting in a more or less complex RT channel establishment behavior. One solution is that each switch tries to find proper server parameters that minimize its corresponding server's required network bandwidth while maintaining the correctness of (2). However, this solution can result in a fairly complex problem to solve, as the overall end-to-end timing requirement has to be fulfilled by the sequence of servers involved in the transmission of messages belonging to one flow. By allowing servers to locally minimize impact in terms of required network bandwidth may impose a much higher (than optimal) network bandwidth requirement at another switch. A less complex solution to the problem is to divide the deadline D_i by the number of switches (k_i) in the path that the flow has to traverse as in [3]. However, in this solution each switch need the complete view of the network architecture in order to know the exact number of switches in the path. This aspect is achieved with a static configuration of the network, which is acceptable in some industrial communications or with protocol to discover the network architecture. On the other hand, this approach can reject a stream although the end-to-end delay meets the deadline (D_i), since if only one congested node misses the partitioned deadline, then the flow will be rejected. To overcome this limitation, we propose to study another approach. If one node misses the partitioned deadline, then it will compute its best effort delay and the difference to the partitioned deadline will be subtracted by the flowing nodes in the path and transmitted to them through the request message. In our work we are considering these approaches, as we intend to investigate them from the perspective of runtime complexity as well as success ratio when it comes to establishment of flows.

6 Conclusions

The integration of *flow management* with hierarchical server-based scheduling (originally meant for CPU scheduling) has been tested in the network domain, however just in a small network with a single switch. Therefore, in this work-in-progress paper we address the extension of this approach to networks with multiple switches. Particulary, this paper presents a preliminary work on a protocol for bandwidth reservation, allowing for the transmission of traffic flows with real-time guarantees using server-based scheduling techniques implemented in the switches. This protocol can be improved with an efficient process to handle concurrent requests, always assigning available resources to requests intended for higher priority flows. Moreover, a timing analysis is introduced and discussed. Particulary, the paper discusses some solutions to find servers parameters for all servers that will handle a particular RT channel. This paper contains mostly preliminary results that will be developed, analyzed and tested, in order to show its correctness and performance. Moreover, this protocol can be generalized and applied to other switches, since they provide the enumerated features above.

References

[1] H. Hoang, M. Jonsson, U. Hagstrom, and A. Kallerdahl. Switched Real-Time Ethernet with Earliest Deadline First Scheduling - Protocols and Traffic Handling. In *10th International Workshop on Parallel and Distributed Real-Time Systems*, 2002.

[2] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, Oct. 1997.

[3] P. Pedreiras and L. Almeida. Message Routing in FTT multi-segment networks: the Isochronous Approach. In *12th IEEE Workshop on Parallel and Distributed Real-Time Systems*, April 2004.

[4] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida. Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication. In *7th IEEE Workshop on Factory Communication Systems*, May 2008.

[5] R. Santos, A. Vieira, R. Marau, P. Pedreiras, A. Oliveira, L. Almeida, and T. Nolte. Flexible, Efficient and Robust Real-Time Communication with Server-based Ethernet Switching. In *8th IEEE Workshop on Factory Communication Systems*, May 2010.

[6] R. Santos, A. Vieira, R. Marau, P. Pedreiras, A. Oliveira, L. Almeida, and T. Nolte. Improving the efficiency of Ethernet switches for real-time communication. In *1st International Workshop on Adaptive Resource Management*, April 2010.

[7] S. Varadarajan and T. Chiueh. EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch. In *International Conference on Network Protocols*, October 1998.

[8] M. Zhang, J. Shi, T. Zhang, and Y. Hu. Hard Real-Time Communication over Multi-hop Switched Ethernet. In *International Conference on Networking, Architecture, and Storage*, 2008.