Flexibility, Timeliness and Efficiency over Ethernet

Paulo Pedreiras[1], Luís Almeida
DET / IEETA – Universidade de Aveiro
3810-193 Aveiro, Portugal
pedreiras@alunos.det.ua.pt, lda@det.ua.pt

Abstract

This paper summarises the materials presented in [11] concerning the quest for real-time behaviour over Ethernet and the new protocol FTT-Ethernet. This paper then includes a discussion on the use of this new protocol on networks based on switches, as this is becoming the main architectural choice in LANs.

1 Introduction

In the Internet Age, many new services are being created everyday, some of which will require real-time support, e.g. voice over IP, videoconference, remote monitoring of control processes, streaming services. To fulfil the requirements of such services, proper Quality-of-Service control mechanisms must be used either by transit networks as well as by access networks. In the first case, there are a few technologies that already support such mechanisms, e.g. ATM and Frame Relay. In what concerns the latter type, the main technology used is Ethernet for which the development of efficient QoS control mechanisms is particularly relevant. This paper summarises the materials presented in [11] concerning the QoS aspects of timeliness (real-time behaviour), flexibility and efficiency, as well as a new protocol, FTT-Ethernet. Finally the paper discusses the advantages that may arise from using switches to support the new protocol.

2 Real-time and Ethernet

The quest for real-time behaviour on Ethernet led to several approaches and techniques. This section presents and characterizes some of these paradigmatic efforts that, nevertheless, either require specialised hardware, are suited to soft-real-time operation only, or are bandwidth or response-time inefficient.

2.1 Modification of the Medium Access Control

This approach consists on modifying the Ethernet MAC layer to achieve a bounded access time to the bus (e.g. [2], [7] and [8]). For instance, the solution presented in [2] (CSMA/DCR) consists in a binary tree search of colliding nodes, that is, there is a hierarchy of priorities. Whenever a collision happens the lower priority nodes voluntarily cease contending for the bus, and higher priority nodes try again. This process is repeated until a successful transmission occurs. Aspects against: requires modification of the firmware (no benefit from economy of scale of standard Ethernet), long worst-case transmission time with respect to average (pessimistic analysis thus bandwidth under-utilization).

2.2 Adding transmission control over Ethernet

This method consists on adding a layer above Ethernet, intended to control the instants of message transmissions, ending up with a bounded number of collisions or even a complete avoidance of them. In favour: standard Ethernet hardware can be used. Several different transmission control implementations are referred below.

Master/Slave. Any node transmits messages only upon receiving an explicit command message issued by one particular node called Master. In favour: relatively precise timeliness (depending on the master). Against: introduces a considerable protocol overhead (master messages); inneficient handling of event-triggered traffic (unknown transmission instants).

Token-Passing. A token is circulated among the nodes. Only that one currently holding the token is allowed to transmit and the token holding time is upper bounded (e.g. IEEE 802.4). Against: protocol overhead (bandwidth used by token); poor support for periodic traffic; bus inaccessibility caused by token losses.

Virtual Timed-Token. Basis of the RETHER protocol [3]. In real-time mode, nodes are divided in two groups: the RT group for nodes with bandwidth reservations; the NRT group for all the others. The real-time messages are assumed to be periodic, and time is divided in cycles with the duration of one time unit. Access to the channel for both kinds of traffic is regulated by a token. First, the token visits all the RT senders having messages to be produced in that cycle, and after the NRT nodes, if enough time is left until the end of the cycle. In

[1] This work was partially supported by the Portuguese Government through grant PRAXIS XXI/BD/21679/99 and project CIDER-POSI/1999/CHS/33139.

[1] This work was partially supported by the Portuguese Government through grant PRAXIS XXI/BD/21679/99 and project CIDER-POSI/1999/CHS/33139.

favour: online admission control of RT messages. Against: lack of support for real-time sporadic traffic; high overhead (as in master/slave).

TDMA. In this case, nodes transmit messages at pre-determined disjoint instants in time in a cyclic fashion. In favour: high bandwidth efficiency (no control messages). Against: requires precise clock synchronization; hardly supports dynamic changes in the message set (communication requirements are distributed and changes must be done globally).

Virtual Time Protocol. This protocol [9] [10] tries to reduce the number of collisions on the bus while offering the flexibility to implement different scheduling policies, e.g. minimum-laxity first. When a node wishes to transmit a message it waits for a given amount of time counting from the moment the bus became idle. This amount of time is calculated according to the desired scheduling policy. When that time expires, and if the bus is still idle, the node tries to transmit the message. If a collision occurs, then there is another node with a message with the same laxity. In this case the protocol uses a probabilistic approach: the nodes involved in the collision either retransmit the message with probability *p* or wait for another similar amount of time. Against: high sensitiveness to the proportional constant value used to relate the waiting time with the scheduling policy; with collisions, worst-case transmission time is much higher than average.

Window protocols. These types of protocols have been proposed for both CSMA/CD and token ring networks [9]. In the former ones, the nodes on a network agree on common time interval named window. The bus state is used to assess the number of nodes with messages to be transmitted within the time window: if the bus remains idle, there are no messages to be transmitted in the window; if only one message is in the window, it will be transmitted; if two or more messages are within the window, a collision occurs. Depending on the bus state, several actions can be performed: if the bus remains idle, the time window is increased; in the case of a collision, the time window is shortened. Against: collisions lead to bus under-utilization if timeliness must be guaranteed.

2.3 Traffic shaping

This technique follows an approach based on the statistical relationship between bus utilization and collision probability. Keeping the bus utilization below a given threshold allows obtaining a deemed collision probability. One implementation of this technique is presented in [6]. An interface layer, called traffic smoother, is placed between the transport layer (TCP/UDP) and Ethernet. The traffic smoother gives real-time traffic priority over non-real-time one (NRT) within each node. Moreover, this layer keeps track of the traffic generated by the node, and controls the transmission of NRT traffic, in order to keep the network load originated by the node below the specified value. This approach provides statistical guarantees, thus it is suited to support soft real-time traffic, only.

2.4 Switched Ethernet

The use of switches became very popular recently, as a way to improve the performance of shared Ethernet. Switches provide a private collision domain for each of its ports. When a node transmits a message, this one is received by the switch and then buffered in to the output ports where the receiver(s) of the message are connected. If several messages addressed to a given port arrive in a short interval, they are queued and then sequentially transmitted. Concerning the scheduling of messages waiting in an output port, 8 priority queues are available (IEEE 802.1D). The scheduling policy used at this level is a topic currently addressed in the scientific community (e.g. [5]).

Simply adding a switch to an Ethernet network is not enough to enforce real-time behaviour. For example, in distributed control systems the producer/ consumers model is typically used, in which one producer of a given datum (e.g. a sensor reading) sends it to several consumers of that datum. In shared Ethernet this feature is supported by means of special addresses (multicasts). However, switches handle this kind of traffic as broadcasts, and thus one of the major benefits of switched Ethernet, multiple simultaneous transmission paths, can be seriously compromised. Other problems concerning the use of switched Ethernet are [1]:

- In the absence of collisions the switch introduces an additional latency;
- The number of available priority levels is too small to support efficient priority based scheduling;
- The switch only makes Ethernet deterministic under controlled loads, therefore, to support hard real-time traffic an appropriate admission control policy must be added.

3 FTT-Ethernet protocol

The rationale behind the FTT-Ethernet protocol is to support hard real-time communication in a flexible and bandwidth efficient way, using COTS hardware. It aims at distributed real-time systems used either in industrial automation, multimedia or embedded control applications. The protocol is based on the Flexible Time-Triggered (FTT) paradigm, which can be implemented on different networks, e.g. FTT-CAN [4] that uses the Controller Area Network. Independently of the underlying network, FTT-based protocols present the following features:

1. Time-triggered communication with operational flexibility;
2. Support for on-the-fly changes both on the message set and the scheduling policy used;
3. Online admission control to guarantee timeliness to the real-time traffic;
4. Indication of temporal accuracy of real-time messages;
5. Support of different types of traffic: event-triggered, time-triggered, hard real-time, soft real-time and non-real-time;
6. Temporal isolation: the distinct types of traffic must not disturb each other;
7. Efficient use of network bandwidth;

Based on these features, table 1 shows a summarized comparison among several approaches to real-time communication over Ethernet, including FTT-Ethernet.

3.1 Protocol description

To support the features referred to above, the FTT-Ethernet protocol relies on centralized scheduling and master/ multi-slave transmission control.

Protocol	Traffic classes			Dynamic comm. req.	Timeliness guarant.	Temporal isolation	Efficiency	COTS hardware
	Real-time		Non-Real-time					
	Time Trig.	Event Trig.						
CSMD/DCR	No	Yes	Yes	Yes	Hard (1)	No	Low (2)	No (5)
TDMA	Yes	No	No	No	Hard	N.A.	High	Yes
Virtual Time Protocol	No	Yes	Yes	Yes	Hard (1)	No	Low (2)	Yes
Windows Protocols	No	Yes	Yes	Yes	Hard (1)	No	Low (2)	Yes
Virtual Timed-Token	Yes	No	Yes	Yes	Hard	Yes	Low (3)	Yes
Switched Ethernet	No	Yes	Yes	Yes	No (4)	No	High	Yes
Traffic Smoothing	No	Yes	Yes	Yes	Soft	No	Low (2)	Yes
FTT-Ethernet	Yes	Yes	Yes	Yes	Hard	Yes	High	Yes

(1) Worst-case resposnse time much higher than the average value
(2) Collisions are part of the protocol
(3) Each Real-time message is preceded by a control message
(4) Can be achieved by the use of a system admition control policy
(5) Requires modifications on the NIC's firmware.
N.A. Not appliable

Table 1: Comparing different approaches to real-time communication over Ethernet.

Centralized traffic scheduling allows having both the communication requirements and the message scheduling policy localized in one node, the Master, facilitating on-line changes to both. On the other hand, such centralization also facilitates the implementation of on-line admission control in the Master node.

Master/multi-slave transmission control allows enforcing the traffic timeliness in the bus without incurring in a high penalty concerning the efficiency in bandwidth utilization. The first aspect is typical of master-slave transmission control since the master explicitly tells each slave when to transmit, thus enforcing the traffic timeliness. The second aspect results from a single Master message triggering the transmission of several messages by slave nodes.

In FTT-Ethernet traffic is allocated in fixed duration time slots called Elementary Cycles (EC). The bus time is organized in an infinite succession of ECs. Each EC starts with a trigger message (TM), sent by the Master, and is composed by two sequential phases for transmission of time and event-triggered traffic (**Figure 1**). The traffic on the synchronous window is controlled by the TM, which contains the identification and length (Tx_i in **Figure 1**) of the messages that must be transmitted within the respective EC (EC-schedule).

Each node holds a table identifying which synchronous messages it produces. Upon reception of the EC trigger message, slave nodes decode the EC-schedule information and compare it with the table of the locally produced messages, in order to identify which synchronous messages the node should produce in the current EC. These messages are then queued for transmission in the synchronous window. The information carried by the TM is enough to allow producer nodes to transmit the messages at disjoint time instants, therefore collisions are avoided. The transmission of synchronous messages is handled by the synchronous messaging system (SMS).

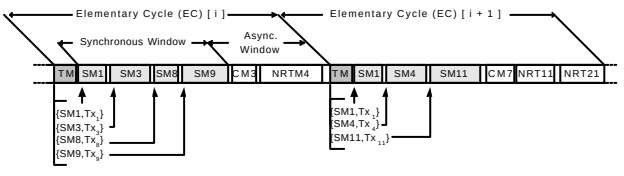

Figure 1. Elementary Cycle structure

The FTT-Ethernet protocol supports asynchronous traffic for event-triggered communication. The FTT-Ethernet protocol receives and queues the transmission requests originated in the application layer. Nodes are periodically pooled for asynchronous messages, and, when allowed to, the respective messages are de-queued and transmitted.

The FTT-Ethernet also supports non-real-time traffic, which is transmitted within the asynchronous window. This type of traffic is handled under a best effort policy and is typically associated to common applications using higher-level communication protocols such as TCP/IP (e.g. http, ftp). Real-time asynchronous traffic is transmitted first and then, if time is available within the EC, nodes producing non-real-time messages are polled. The polling order can be scheduled in order to support distinct Quality of Service (QOS) according to nodes requirements. Asynchronous message transmission is handled by the Asynchronous Messaging System (AMS).

3.2 The Master node

The Master node plays the role of system coordinator and it is responsible for keeping a database holding the system configuration and communication requirements (SRDB); building EC-schedules according to the particular scheduling policy implemented; and finally broadcasting the EC trigger message, containing these schedules, at the start of each EC.

A set of tasks residing in the Master node implement the functionalities required for proper system operation, namely system configuration and management, message scheduling and dispatching and admission control.

3.3 Slave nodes

Slave nodes execute the application software required by the user, eventually requesting the services delivered by the communication system. This system is organized as two parallel stacks, one for non-real-time and the other for real-time communication. The former uses a standard IP protocol suite, where the only specific component of the FTT-Ethernet protocol is a modified Data Link Layer (DLL). The latter follows the collapsed 3 layers OSI reference model typically found in fieldbus systems, providing a specific application interface (Real-Time Application Programming Interface), which enables the applications to configure the messages locally produced and consumed, respectively update or read the value of such real-time entities and set-up call-backs associated to communication events.

In what concerns the DLL, a transmission control layer (FTT-Ethernet Interface Layer) is added on top of the Ethernet layer, both for real-time and non-real-time communication. The FTT-Ethernet Interface Layer receives and decodes the TM and transmits the locally produced messages according to the respective EC-schedule. Moreover,

this layer also handles received data messages, passing the data to the respective protocol stack when the data is locally consumed.

4 Shared or Switched Ethernet

As stated in section 3.1, the FTT-Ethernet protocol performs collision-free message exchange over shared Ethernet, since all messages are scheduled at disjoint time instants. In this scenario, each node must decode the temporal information conveyed in the trigger message. This information is then used to set up timers that will trigger the message transmissions at appropriate time instants.

Despite having been originally designed for shared Ethernet, the new protocol can also work without any modification over switched Ethernet or over mixed shared/switched networks. However, in the absence of shared segments, i.e. relying on switches only, the FTT-Ethernet protocol can take advantage of switched Ethernet features. For example, the queueing at the switch ports can be used to alleviate the processing overhead required in each node because there is no need for tight control on the message transmission instants. In this case, upon reception of the trigger message, the nodes only need to identify which messages they should produce within the EC, and transmit them immediately. The switch prevents destructive collisions and serializes the message transmissions in the output port queues. The fact that this option requires networks built exclusively with switches is not particularly restrictive since that is fast becoming the preferential architectural choice in current LANs.

On the other hand, the use of FTT-Ethernet may even contribute to reduce some of the problems of using switches in real-time applications as referred to in section 2.4. This comes from the fact that FTT-Ethernet performs the traffic control required to support adequate management in the output port queues and enforcing priority scheduling beyond the priority levels available in 802.1D.

In the former case, the Master can schedule transmissions taking into account the destination address of messages, either broadcast, multicast or unicast. This knowledge can be used in suitable scheduling policies for two different purposes. On one hand, it can be used to avoid scheduling more messages per EC to each output port than those that fit in the respective queue, thus preventing overflow and the consequent loss of messages. On the other hand, this knowledge can also be used to promote the simultaneous scheduling of messages that follow disjoint paths, taking advantage of the possibility of simultaneous data transmissions in the switches and thus improving the overall throughput.

In the latter case, the scheduling carried out by the Master node may take into account individual priorities of each message, possibly dynamic priorities, e.g. for EDF scheduling, which are neither restricted nor correlated to the 8 priority levels defined in 802.1D. This way, FTT-Ethernet supports strict priority scheduling within each of the priority levels defined in the standard. The term *strict*, though, can only be applied at a coarse time scale, in EC units, since priority invertions within the EC can occur.

Finally, using FTT-Ethernet on hierarchical multi-switch networks requires the addition of extra idle-time in each EC to cope with the forwarding delays at each level. Notice that all the traffic related to a given EC must arrive to the farthest point in the network within the same EC as perceived by the Master.

5 Conclusion

The FTT-Ethernet protocol presented in [11] is more flexible and efficient than other existing techniques to enforce real-time behaviour on Ethernet (table 1). Furthermore, its application to switched networks has been discussed and seems particularly advantageous. Therefore, the new protocol seems well adapted to enforce the QoS control mechanisms required by emergent applications, even Internet-based.

6 References

[1] Decotignie, J-D. A perspective on Ethernet as a Fieldbus. *FeT'01, 4th Int. Conf. on Fieldbus Systems and their Applications*. Nancy,France. Nov. 2001.

[2] LeLann, G, N. Rivierre. Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD-CSMA-CD Protocols. *INRIA Report RR1863*. 1993.

[3] Venkatramani, C., T. Chiueh. Supporting Real-Time Traffic on Ethernet. *IEEE Real-Time Systems Symposium*. San Juan, Puerto Rico. Dec 94.

[4] Almeida, L., P. Pedreiras and J.A. Fonseca. FTT-CAN: Why and How. *IEEE Trans. on Industrial Electronics* (to appear). December 2002.

[5] Jasperneit, J., P. Neumann. Switched Ethernet for Factory Communication. *ETFA'01, IEEE Conf. Emerging Techn. on Factory Automation*. Antibes, France. Oct. 2001.

[6] Kweon, S-K. and K. G. Shin. Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. *IEEE Real-Time Technol. and Applications Symp*., 90-100. Washington DC, USA. June 2000.

[7] Shimokawa, Y. and Y. Shiobara. Real-Time Ethernet for Industrial Applications. *IECON'85*, pp829-834. 1985.

[8] Court, R.. Real-Time Ethernet. *Computer Communications*, **15** pp. 198-201. April 1992.

[9] Malcolm, N., W. Zhao. Hard Real-Time Communications in Multiple-Access Networks. *Real Time Systems* **9**, 75-107. Kluwer Academic Publishers. 1995.

[10] Molle, M., L. Kleinrock. Virtual Time CSMA: Why two clocks are better than one. *IEEE Transactions on Communications*. **33(9)**:919-933. 1985.

[11] Pedreiras, P. L. Almeida and P. Gai. The FTT-Ethernet Protocol: Merging Flexibility, Timeliness and Efficiency. *ECRTS'02, EUROMICRO Conf. on Real-Time Systems*, Vienna, Austria. June 2002.