University of Telecommunications, Leipzig

Ethernet as a Real-Time Technology

Sebastian Lammermann

University of Telecommunications, Leipzig

Ethernet as a Real-Time Technology

Seminar Paper

Author:	Dipl.-Ing. Sebastian Lammermann
	Matriculation number 074111
Subject:	Operating Systems II
Date of completion:	17 June 2008
Contact:	sebastian@lammermann.eu

Contents

1 Preface

This seminar paper relates to the subject “Ethernet as a real-time technology” which was part of the Operating Systems course offered by the University of Telecommunications, Leipzig in summer term 2008.

This document divides into three main parts as well as an introduction and a conclusion. The first main part, chapter 3, delivers a definition of the term “real-time”. Chapter 4 focuses on Ethernet today and identifies the technology's general issues concerning real-time communication. Chapter 5 offers several approaches to solve these problems as well as further considerations.

Due to the relatively low number of sources it was refrained from a division of the bibliography into linear and hypertext sources.

2 Introduction

The goal of this seminar paper is to point out and to analyse different methods towards a real-time Ethernet-based networking technology. The focus is on two relatively popular approaches, RTnet and RETHER, which will be described in detail. In both the original algorithm responsible for access on the medium is replaced with a deterministic one. In addition other suggestions are mentioned and all is supplemented by further considerations. The functionality of each technology is illustrated and advantages and drawbacks are pointed out. The listed sources deliver more detailed information.

A definition of the term “real-time” is given as well as information about Ethernet in general and its CSMA/CD algorithm. The reason why Ethernet does not fulfil real-time constraints is also discussed.

3 Defining “real-time”

In computer science a “real-time” system is any hard or software system which is a subject to a “real-time” constraint [LiuJ00], meaning that the system has to respond within a defined period of time. Even if this definition does not mean that a real-time system answers immediately it guarantees that an action occurs within a given interval following an event (relative time) or at

fixed dates (absolute time) [HoHa91].

By contrast, a non-real-time system operates without these *deadlines* and responds as soon as the job is terminated. In soft real-time systems a missing deadline means that the result might be worthless if processing takes too much time. If for example a ticket machine needs too long to print a ticket, one might miss the train. Highly critical are hard real-time systems. If the result is not available within the defined time limit – e. g. the activation of an air bag in a car accident – the process ends in a catastrophe.

If we understand computer networks as hard and software systems, aspects of real-time communication might be of importance. In this case it must be guaranteed that information arrives at its destination within a preassigned period of time. In today's packet switching world this leads to general difficulties in transmitting time-critical data – like speech or video streams – using the popular TCP/IP protocol family, simply because neither route nor delay of the packets is known in advance [AKRS94] [Detk02]. But even if real-time is introduced to OSI layer 3 (network layer) and above, the lower ones might be the bottleneck in the end.

4 Ethernet today

4.1 General information

During the 1990s Ethernet became the most popular networking technology in *Local Area Network* (*LAN*) environments and since then developed quickly and largely replaced other standards such as Token Ring or FDDI. Nowadays 100 and 1000 Mbit/s twisted pair based LANs are the rule and even Internet service providers begin to replace ATM by Ethernet over fibre in their backbone and access networks [TIRC07].

Ethernet components are not just relatively cheap compared to other standards, but the technology is also a flexible foundation for diverse network protocols like TCP/IP, DECnet or AppleTalk as well. Depending on the standard, Ethernet can be operated on several media like coaxial and twisted-pair copper cable or optical fibre. Using a converter one might interconnect two different physical to one logical network. In addition Ethernet is able to operate in a switched network environment, which helps to cut down costs and network load.

Ethernet shows some similarities to human communication and radio systems. Every component may use the network – which is organised as a logical bus (the *ether*) – to transmit messages to any other as long as there is not already a different communication in progress.

Messages might be directed to a single component on the net (*unicast*) as well as to multiple (*multicast*) or all devices (*broadcast*). If a network component does not send data it listens for messages.

Ethernet is a packet switching technology which means that messages are being subdivided into discrete blocks of data. Every single one of these "packets" contains information such as its origin and destination, its length, a checksum etc. in addition to a payload. In the past Ethernet used to be a single-segment technology, which means that all network nodes were physically and logically interconnected and able to listen to all messages on the bus. Nowadays switched LANs are the rule so that nodes will only receive messages that were addressed to them.

4.2 CSMA/CD

In order to control access to the medium Ethernet uses a relatively simple algorithm known as *Carrier Sense Multiple Access with Collision Detection* (*CSMA/CD*), which is a modification of CSMA [IEEE05].

CSMA originally consisted of two procedures: the first (*Carrier Sense*) describes a network component that listens for a carrier wave before sending data. If this carrier exists the transmitter tries to detect whether there is already a signal from another device. If the bus has been free for a certain period of time (*Inter-Frame Gap*)[1] data might be sent. The second procedure (*Multiple Access*) simply describes the fact that the network forms a logical bus. This means that multiple nodes may access the same medium and that every message is received by all devices attached to the bus. Network components only process data that is addressed to them.

If two or more components try to send data at the same time a collision occurs. In order to improve the CSMA algorithm and to enhance the efficiency of Ethernet *collision detection* (*CD*) was introduced. While sending data every device now monitors its own transmission. If a node detects a collision – e. g. by receiving a signal with too high voltage – it stops its transmission and then sends out a *jam signal* consisting of a 32 bit sequence. This jam signal has two effects: the first is that all nodes listening to the signal will discard the current frame due to an incorrect checksum. Furthermore all other devices currently transmitting will stop doing so. In the end the bus becomes idle.

In order to avoid another collision all involved network nodes calculate a random time

1 Inter-Frame Gap in 100 Mbit/s Fast Ethernet: 960 ns.

interval before trying to retransmit its message. The following formula is used:[2]

$$t_{wait} = (0, \ldots, 2^M - 1) \cdot t_{slot} \quad (4.2.1)$$

where M is the number of total retransmission attempts[3] but not higher than 10. Figure [4.2.1] shows an example of a collision detection:

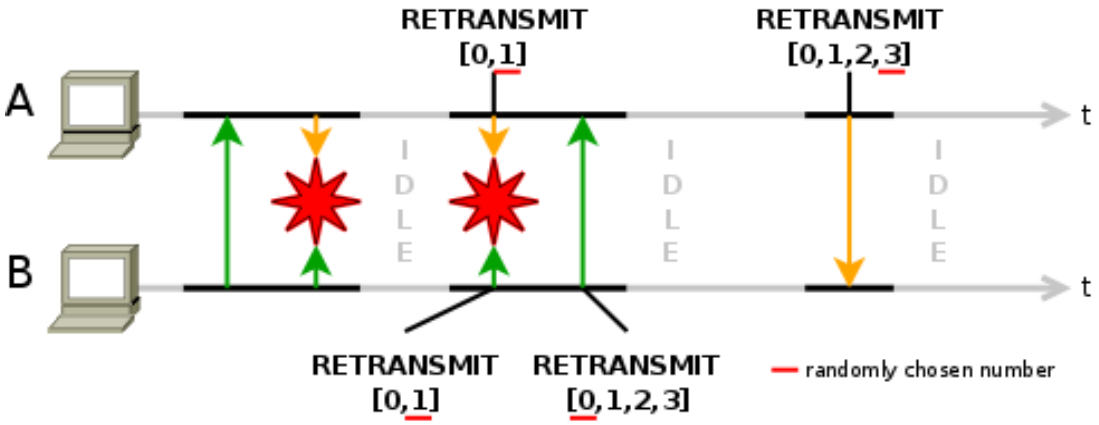

Figure [4.2.1]: CSMA/CD collision detection

4.3 Issues

Its low cost, flexibility and worldwide application made Ethernet the most promising candidate for "the one" networking technology to replace all others. Nevertheless Ethernet faces one major issue which prevents its entry into all fields of inter-system communication: it does not feature real-time transmission. Even if regular computer networks work fine without this attribute there are fields where real-time does matter. Media streaming, industrial networks, telephony and vehicle systems are just a few examples.

Compared to other technologies (e. g. ATM) the packet length in Ethernet may vary from 84 up to 1542 octets.[4] In combination with CSMA/CD this is the main reason why Ethernet is non-deterministic [YCZS05]. This means that neither transmission duration nor arrival time of messages can be predicted which is the major requirement for real-time communication.

2 Slot time in 100 Mbit/s Fast Ethernet: 5120 ns.
3 After 15 failed attempts a waiting packet is discarded.
4 Based on Ethernet-II-Frames incl. *VLAN tag*, *preamble* and *Start Frame Delimiter*.

5 Ethernet as a real-time technology

5.1 General information

As described earlier Ethernet is the most popular networking technology today but currently is not capable of being used for real-time applications. However there are plenty of approaches towards making Ethernet fulfil real-time constraints.

Each of the following two chapters presents a popular approach in detail and part three supplements these with general information about additional proposals. Finally the last two sections give a summary of these already existing technologies as well as further considerations and a discussion about other suggestions.

Due to the fact that switched real-time networks are considerably complex, all approaches in this paper are single-segment variants only. In switched environments every device has its own cable so that collisions can no longer occur. The real-time scheduling algorithm then has to be implemented in the switches. More about switched real-time Ethernet networks can be found in the papers of C. Venkatramani [Venk96] or J. Jasperneite and P. Neumann [JaNe01].

5.2 Approach 1: RTnet

5.2.1 Introduction

The following chapter describes an *RTnet* approach published at the University of Twente by F. Hanssen, P. Jansen et al. [HJSH04]. The network protocol's state machine and the different states will not be explained in more detail.

The aim of the authors is to develop a real-time Ethernet-based networking technology without making any changes to the Ethernet protocol itself. This would guarantee that the popular and low cost Ethernet hardware could be used further on. The documentation points out the following requirements for RTnet:

- *Scheduling*: A scheduling technique is required so that packets arrive before their deadline expires.
- *Non-real-time traffic*: A certain percentage of the total bandwidth has to be reserved for non-real-time communication.
- *Network recovery*: Because of physical failures etc. network faults cannot be completely ruled out. On this account the network must be able to recover from failure.

- *Plug & Play*: Adding and removing network components dynamically should be possible.

The authors act on the assumption that real-time traffic can always be separated into streams between two nodes. So real-time communication is always taking place between a primarily sending and a primarily receiving network component. Because of the bus architecture of Ethernet only one stream can be active at once while all others have to wait. This necessitates the implementation of a scheduling algorithm.

5.2.2 Algorithm

In order to make Ethernet real-time capable the authors decided to replace CSMA/CD with a completely different and deterministic algorithm. In this version of RTnet a preemptive *Earliest Deadline First* (*EDF*) algorithm has been implemented which is relatively simple but may use the network's full capacity. All nodes use the same algorithm and all scheduling information has to be transmitted to all nodes attached to the network.

To prevent collisions on the bus and to transfer scheduling information a token is used. The component holding the token is the only one in the network that is allowed to send data while all others are listening. The time a node may hold a token (*Token Holding Time*, *THT*) is determined in advance by the EDF network schedule. Once the THT has expired the token will be forwarded to the next node assigned by the schedule.

To improve the quality of RTnet and to avoid token loss every node will become the *monitor* once it has passed the token. The monitor keeps a copy of the token in case the node currently holding the token is unable to forward it, e. g. due to a device failure. If so the monitor is still able to pass the copy of the token to the next node. Usually monitor and token holder are two different devices. One node may only be monitor and token holder at the same time in two cases: when there is only one node in the network and during network initialisation. If an error occurs it may happen that there are two tokens in the network. In this case the receiving node will inform one of the monitors to delete its token and to stop monitoring.

RTnet allows adding and removing nodes – and therefore real-time streams – dynamically. While removing nodes does not create any difficulties, adding streams might cause some trouble. The reason is that the network might not be able to provide the required resources. Thus a feasibility analysis is carried out when a new component joins the network to see if all

components can still meet their deadlines. A stream has three scheduling parameters:

- The bandwidth B it needs (replacing time in CPU scheduling)
- The time difference (period) T between two consecutive messages
- A maximum message size S of a message, which is equal to the product of bandwidth and period (5.2.2.1)

$$S = B \cdot T \quad (5.2.2.1)$$

It is necessary to calculate the utilisation of all streams and to compare it with the available bandwidth. If the total utilisation in general is smaller than the available bandwidth the stream can be added:

$$\sum_{stream=1}^{n} \frac{B_{stream}}{0.8 \cdot B_{available}} \leqslant 1 \quad (5.2.2.2)$$

The available bandwidth is not equal to the maximum transfer rate of the network. First of all it is important to reserve some bandwidth for non-real-time communication so that real-time traffic does not clog the entire network.[5] Attention should also be paid to tokens and Ethernet overhead which costs bandwidth as well. Furthermore the authors expect real-time streams to use maximum size Ethernet packages.[6] Before a transmission may start a node requests a token and after finishing a transmission the token will be forwarded to the next node. In the worst case a stream preempts another so that a second transmission is necessary and additional bandwidth is needed for sending a confirmation message to the monitor. Altogether the stream's bandwidth can be calculated as follows:

$$B_{stream} = B_{data} \frac{1542\,Byte}{1500\,Byte} + 2 \cdot (B_{token} + B_{monitor}) \quad (5.2.2.3)$$

Although RTnet is a real-time networking technology, hardware failures may cause deadline misses. They occur whenever a node receives a token too late. As a result of this the affected node informs the sending application and tries to keep all other streams within their deadlines by forwarding the token to the next device immediately. Thus the next stream is likely to meet its deadline.

5 The authors recommend reserving 20 % of the total bandwidth for non-real-time traffic.

6 The authors did not consider VLAN tags so that the following formula originally expected the maximum Ethernet packet size to be 1538 Bytes.

5.2.3 Non-real-time traffic

One of RTnet's requirements is the possibility to transmit non-real-time traffic as well. Therefore a certain percentage is reserved for regular Ethernet communication. Non-real-time traffic may only be transmitted when no real-time stream needs to send data. Due to the fact that regular packets know neither a deadline nor a priority no scheduling is needed. On this account the token passing algorithm will change from EDF to *round-robin*. As soon as a real-time stream becomes ready the token has to be passed to the related node immediately. However the last non-real-time node holding the token should be remembered so that non-real-time communication may start again at this point as soon as there are no more real-time streams transmitting. By this timeouts can be avoided.

5.2.4 Criticism

Tests showed that the algorithm's results were very good for streams with periods larger than 100 ms on 10 Mbit/s Ethernet while improvement is needed for shorter intervals. Further development is intended.

But even if the algorithm works fine this RTnet variant implies some major issues. First of all RTnet does not account for priority and therefore does not differentiates between hard and soft real-time systems. Furthermore the authors expect large amounts of unidirectional data to be sent which indicates that the technology has been designed for soft real-time systems. However it seems to not be optimised for streaming media like IPTV where multicasts play an important part. Hard real-time systems on the other hand often send unicasts which are short but highly time-critical. And lastly RTnet is not compatible with Ethernet. The hardware can be reused but the software needs to be replaced on all network components which will incur enormous costs.

5.3 Approach 2: RETHER

5.3.1 Introduction

RETHER is a real-time networking technology that is based on Ethernet and was developed at the State University of New York at Stony Brook. It exists in two versions: single-segment RETHER for basic networks and – not further discussed in this document – an extended

version for multi-segment networks. C. Venkatramani specifies it in her dissertation in more detail [Venk96] and F. Hanssen and P. Jansen present it in their paper [HaJa03].

The idea of RETHER is to leave Ethernet mainly untouched but to modify OSI layer 2 (data link layer) so that only one node may transmit packets at once. This ensures that access to the medium is collision-free and deterministic. Real-time channel access is regulated by a token so that a scheduling algorithm can be applied.

RETHER is a hybrid technology that allows using real-time and non-real-time communication in the same physical and logical network. In general real-time communication is prioritised and uses a token-based scheduling algorithm, while non-real-time traffic makes use of Ethernet's classical CSMA/CD.

RETHER operates in CSMA/CD mode as long as no request for real-time traffic is generated. If a request is detected all nodes switch into RETHER mode. To ensure this works correctly all devices attached to the network must be able to generate and to understand these real-time requests.

In RETHER mode the token circulates around the network nodes in configurable periods (*round-robin* scheduling algorithm). In each cycle all nodes that previously reserved bandwidth may send real-time data. If there is no more real-time data waiting in the cycle-specific queue of a node the network component may also send non-real-time data using up the remaining cycle time. Once the token has passed all sending nodes a new cycle is started in case another real-time session exists. If not, the system switches from RETHER back to CSMA/CD mode.

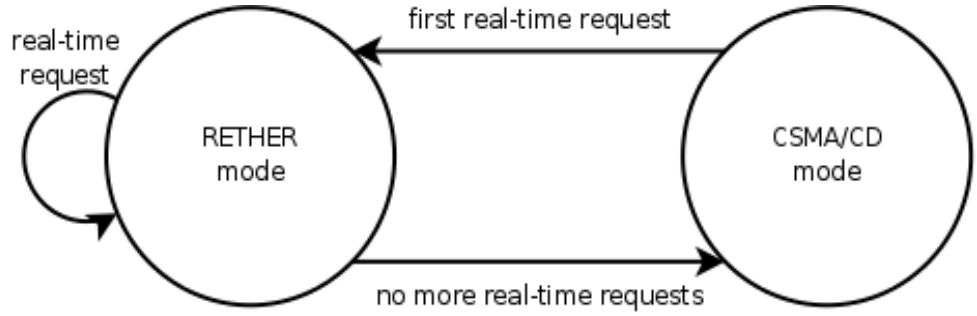

Figure [5.3.1.1]: RETHER and CSMA/CD mode

In RETHER every node receives a unique *identification number* (*ID*) in the range from *0* to *n-1*, where *n* is the number of nodes in the network. Every node also knows exactly how many devices are attached to the network.

5.3.2 Algorithm

By broadcasting a real-time request a node becomes *initiator* and is responsible for monitoring the real-time session. Before the transmission of real-time data may start, all nodes have to send previously collided messages, switch into RETHER mode and thereafter answer by sending an acknowledgement packet back to the initiator. After receiving all acknowledgement messages the initiator creates the token and starts the circulation process.

It may happen that two or even more nodes generate a request at the same time. This is caused by delays in message transmission. This problem is resolved by prioritising the node with the lower ID. A node receiving two or more requests will only answer the initiator with the lowest ID or will not answer if it has already sent out an acknowledgement to a prioritised node.

But even in the case that only one request is generated a node failure may occur. In this case the initiator does not receive acknowledgements from all other nodes (*n-1*). This could have any of three reasons:

- Even if *n* is the maximum number of nodes, some of them could be powered off
- The initiator's network card could have dropped acknowledgement messages
- Some nodes did not receive the request and therefore did not switch into RETHER mode

In all cases this issue is solved by the introduction of a timer. If the initiator does not receive *n-1* acknowledgements within a defined interval it will time out. Then it tries to contact the missing nodes by broadcasting its request a second time. If there is another initiator with a lower ID in the network it informs the first about its priority. If none of the nodes responds they are not operational. In this case the initiator updates the token's device list and starts the first cycle.

Once in RETHER mode, a token circulates between two different types of nodes: real-time and non-real-time components. All nodes that have successfully reserved bandwidth for real-time traffic become part of the real-time set. Accordingly all others belong to the non-real-time set.

In RETHER real-time data is expected to be periodic. The period between two visits of a token at one node is called *Token Rotation Time* (*TRT*). A node may only send its message during the *Token Holding Time* (*THT*). The THT is calculated using the following formulas,

depending on whether is is a real-time or a non-real-time node:

$$THT_{real-time} = \frac{Data\ per\ TRT}{B_{Ethernet}} + t_{processing} + t_{token} \quad (5.3.2.1)$$

$$THT_{non-real-time} = \frac{Message\ size}{B_{Ethernet}} + t_{processing} + t_{token} \quad (5.3.2.2)$$

where $t_{processing}$ is the necessary software processing time and t_{token} refers to the token processing overhead. THT for real-time messages is predetermined and based on the requested time. By contrast THT for regular traffic is calculated individually every time a node holds a token.

Whenever a node requests real-time traffic an *admission control module* checks whether there is enough bandwidth available so that all packets may meet their deadline. This module is implemented in every device so that each node is able to determine if its own request might be admitted or has to be denied. A new real-time session can only be accepted if the following condition is fulfilled:

$$TRT \geq \left(\sum THT_{real-time\ set} \right) + THT_{non-real-time} + THT_{new\ request} \quad (5.3.2.3)$$

where $\sum THT_{real\text{-}time\ set}$ is the total THT of the real-time set and $THT_{non\text{-}real\text{-}time}$ the time reserved for non-real-time traffic in each cycle.

If a request is accepted it is added to the token and may start sending data in the next cycle.

5.3.3 Criticism

RETHER is a real-time technology that works fine in theory and had been enhanced over the last few years. But like RTnet is does not fulfil all necessary requirements for all fields of real-time application.

RETHER does not differ between hard and soft real-time systems which makes it inappropriate for industrial networks. Furthermore it is questionable whether the cyclic round-robin scheduling algorithm used in RETHER generates the best results and guarantees that all deadlines can be met.

RETHER is also not compatible with existing Ethernet networks. In a RETHER network at least all drivers of the network interfaces need to be replaced which is time-consuming and costly. In addition all stand-alone network components, like switches and routers, are incompatible with RETHER and need to be replaced.

5.4 Other approaches

5.4.1 Circumventing CSMA/CD

A deterministic Ethernet technology that is relatively simple to implement has been developed by J. Kerkes. He published a paper [Kerk08] in which he tries to circumvent the CSMA/CD algorithm by implementing a bus arbitration scheme at a higher level of the system. Therefore software has to be implemented on all network nodes. This software is responsible for controlling the network device's access to the medium. In order to make this system work the author points out four rules that have to be followed:

- No regular nodes without this software may be attached to the network
- One device functions as a controller while all others are remote nodes
- No remote node may access the bus before the controller sends a permission message
- Every remote node has a predetermined span of time to respond to a permission message

Even if this technology is simple compared to others it contains some deficiencies. First of all J. Kerkes de facto replaces the CSMA/CD algorithm by his software controller which is a great idea but surely is not the most efficient way of implementing a real-time technology. Furthermore neither a scheduling algorithm nor a priority scheme seems to be included which raises the question how “real-time” this technology really is. It also lacks recovery functionality that enables the network to maintain operation when a network failure occurs. And finally it is not compatible with classical Ethernet – even if the hardware could be reused – so that it cannot be employed in existing networks.

5.4.1 Constraining generated traffic

F. Hanssen, P. Jansen et al. present in their papers [HaJa03] and [HJSH04] two methods to limit generated traffic by the network nodes so that a certain bandwidth can be guaranteed.

The first is called *virtual time CSMA* and it implements a packet release delay mechanism. Each network node is equipped with a *real time* and a *virtual time clock*. Whenever the bus is idle the virtual clock starts running and it stops as soon as the bus is busy. If behind the virtual clock runs faster than real time. Data may only be sent when the message's arrival time is equal to the time of the virtual clock (Figure [5.4.1.1]).

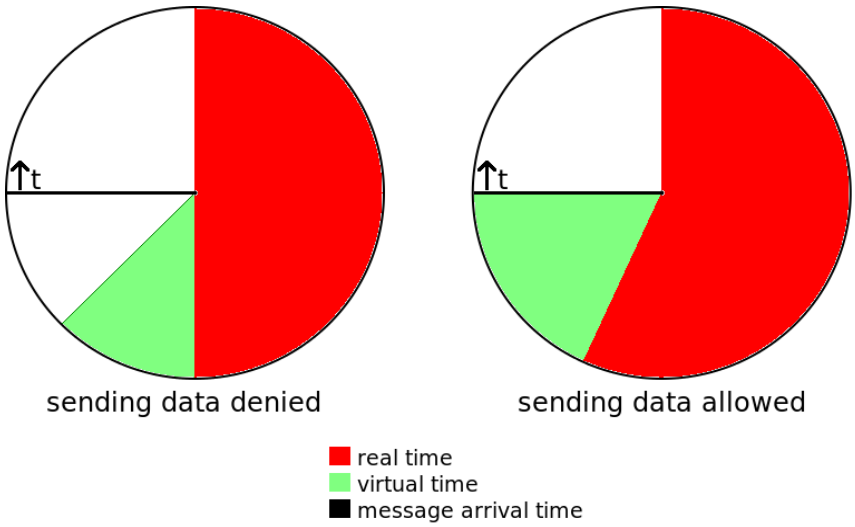

Figure [5.4.1.1]: Packet release delay

When a collision occurs the sending node – depending on the protocol variant - either retransmits the message with a certain probability or modifies the virtual time insofar that the message is put back into the queue.

Several variants of the protocol exist. Each one shows strengths and weaknesses in message loss, network utilisation ratio and collision strategy but none of it is outstanding.

The second method mentioned to constrain generated traffic is *window protocols*. First of all every node has a network-wide knowledge of all messages waiting to be sent. Usually all nodes have messages waiting in a queue that were generated during a certain time window. The most simple case is if only one node has a message in the current window – the message will be sent. If more than one node has a message in the interval the window size will be reduced as long as only one node with a message in the window remains. If there are no more nodes with a message in the interval, the window is shifted further in time. The protocol itself is not suitable for real-time applications so it needs some modifications. According to the authors a policy like *Minimum Laxity First* (*MLF*) makes it fulfil this requirement.

In MLF a collision may occur when at least two messages have the same laxity. This problem is solved by assigning a *contention parameter* to every node that is based on its logical position and the latest time to send. Each contention parameter is unique so that two different nodes cannot select the same value.

Both approaches lack in some area necessary for a real-time network and they are definitely not acceptable for hard real-time systems.

5.5 Summary of existing technologies

Every technology described in this paper lacks in one or another point but none of them analyses real-time traffic in detail. But before designing a real-time networking technology it is highly significant to be aware of some conditions and requirements of real-time traffic.

First of all it is important to bear in mind that the application of real-time communication differs from short distances in separated LAN environments (e. g. industrial inter-machine communication) to long distances through switched networks (e. g. multimedia streaming). Then a priority schedule that accounts for hard real-time messages is needed, simply because hard real-time systems, in contrast to soft ones, have to meet their deadlines by all means. And lastly the aim of a real-time technology has to be the compatibility with current Ethernet so that it can be implemented in already existing networks. Otherwise it might be better to use a completely different but approved networking technology that might even be cheaper in the end.

5.6 Possible approach

The main reason why Ethernet is not real-time capable is that its CSMA/CD algorithm is not deterministic. So what is needed is an alternative that, like in the examples above, could make use of a scheduling algorithm as well as a token that indicates which node may send data. What none of the presented technologies includes is the possibility of implementing a real-time structure parallel to Ethernet. This could be achieved by using the assumed disadvantages of CDMA/CD.

The basic idea is to “misuse” jam signals to interrupt non-real-time Ethernet communication. After a collision and a defined period of time the bus becomes idle and normally one or another node would restart its transmission. Now the trick is to occupy the bus right after the forced collision and to not clear it until all real-time messages have been exchanged. This means that a node, after sending its packet and passing the token, has to occupy the bus as long as the next node may start its transmission. This *handover* could be realised by sending an *epilogue* packet to inform the following node when it has to begin with its transmission. It is also conceivable that the second node synchronises on the epilogue packet of the first so that it starts its preamble immediately when the first node is finished. However a mechanism is needed that avoids the case that two or more real-time nodes try to occupy the bus at the same

time.

The scheduling algorithm behind the technology needs to be priority-based but has to keep in mind that hard real-time traffic always has to be favoured over soft, so that it is guaranteed as far as possible that all schedules meet their deadline.

This “rude” approach towards a real-time system surely will not fulfil the Ethernet standard's condition but some variation might work and is definitely worth being discussed and tested.

6 Conclusion

Nowadays Ethernet is not just the most popular, it is also a very cheap networking technology. But a drawback is that it does not support real-time communication because of its non-deterministic CSMA/CD algorithm. If one would be successful in modifying Ethernet so that it fulfils real-time constraints it could also be introduced to inter-machine communication. This would be tantamount to a revolution and may even spell the end for other technologies like PROFIBUS[7] or CAN.[8]

Admittedly the development of a real-time Ethernet variant is not that far yet. Every single one of the discussed approaches lacks in at least one or another detail so that they need further improvement. And all of them have in common that the differentiation between hard and soft real-time systems is not taken into account and that the scheduling algorithms do not seem to be the best possible ones. But the major disadvantage of all presented networking technologies is its incompatibility to existing Ethernet. Even if only the software in every device needs to be replaced in order to get the system to work, the high costs will make the technology unattractive whereby other already existing real-time capable technologies will have the advantage.

To sum up, one can say that a real-time capable Ethernet variant surely will need some more years of development. But once it is available and its quality convinces, real-time Ethernet will hit the market with an impact that certainly will displace current leading technologies.

7 *Process Field Bus.* A standard for field bus communication in automation technology.

8 *Controller Area Network Bus.* A computer network protocol and bus standard designed to allow microcontrollers and devices to communicate with each other and without a host computer.

Bibliography

[AKRS94] Aras, Ç. / Kurose, J. / Reeves, D. / Schulzrinne, H.: "Real-time Communication in Packet-Switched Networks", University of Massachusetts Amherst, 1994, ftp://ftp.cs.umass.edu/pub/net/pub/incoming/rt.ps.Z.

[Detk02] Detken, K.: "Echtzeitplattformen für das Internet", Addison-Wesley, München, 2002, ISBN: 978-3827319142.

[HaJa03] Hanssen, F. / Jansen, P.: "Real-time communication protocols: an overview", University of Twente, Enschede, October 2003, http://www.ub.utwente.nl/webdocs/ctit/1/000000df.pdf.

[HJSH04] Hanssen, F. / Jansen, P. / Scholten, H. / Hattink, T.: "RTnet: a real-time protocol for broadcast-capable networks", University of Twente, Enschede, 2004, http://www.ub.utwente.nl/webdocs/ctit/1/000000e5.pdf.

[HoHa91] Hoogeboom, B. / Halang, W.: "The Concept of Time in Software Engineering for Real Time Systems", University of Groningen, IEEE, 1991.

[IEEE05] IEEE Computer Society: "Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications", IEEE, New York, 9 December 2005, http://standards.ieee.org/getieee802/802.3.html.

[JaNe01] Jasperneite, J. / Neumann, P.: "Switched Ethernet for Factory Communication", Jasperneite.net, 2001, http://www.jasperneite.net/paper/efta2001.pdf.

[Kerk08] Kerkes, J.: "Real-Time Ethernet", Embedded.com, 15 April 2008, http://www.embedded.com/story/OEG20010221S0086.

[LiuJ00] Liu, J.: "Real-Time Systems", Prentice Hall, Upper Saddle River, 2000, ISBN: 978-0130996510.

[TIRC07] The Insight Research Corporation: "Carrier Ethernet Services 2007-2012 - a market research report", The Insight Research Corporation, 17 May 2008, http://www.insight-corp.com/reports/PES07.asp.

[Venk96] Venkatramani, C.: "The Design, Implementation and Evaluation of RETHER: A Real-Time Ethernet Protocol", State University of New York, Stony Brook, November 1996, http://www.ecsl.cs.sunysb.edu/tr/TR34.ps.Z.

[YCZS05] Yin, R. / Cai, Y. / Zhang, W. / Shen, G.: "Medium Access Control with Packet Length Priority Towards a Real Time Ethernet", IEEE, 2005.

Licence