ISSN 0280-5316
ISRN LUTFD2/TFRT--5683--SE

Scheduling of Real-Time Traffic in a Switched Ethernet Network

Anders Martinsson

Department of Automatic Control
Lund Institute of Technology
March 2002

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* March 2002 *Document Number* ISRN LUTFD2/TFRT--5683--SE
Author(s) Anders Martinsson	*Supervisor* Karl-Erik Årzén and Anders Blomdell *Sponsoring organization*

Title and subtitle
Scheduling of real-time traffic in a switched Ethernet network.
(Schemaläggning av realtidstrafik i ett switchat Ethernet).

Abstract
Traditional Ethernet networks are not suitable for real-time communication due to the nondeterministic handling of the network communication. The reason for the nondeterministic behavior is the CSMA/CD access control protocol that is used when the media is shared.
The protocol can cause collision in the transmission. If this happens the transmission ceases and after a random amount of time a retransmission is tried. Over the years Ethernet transmission rate and communication reliability have increased, which make it a more interesting alternative for periodic real-time communication with a high update rate. This master thesis investigates if it is possible to avoid the nondeterministic behavior of Ethernet, by scheduling the periodic real-time traffic in a switched network.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 50	*Recipient's notes*
Security classification		

Scheduling of real-time traffic in a switched Ethernet network

Anders Martinsson

4 March 2002

Acknowledgment

I would like to thank my supervisors Professor Karl-Erik Årzén and Research Engineer Anders Blomdell for their support.

A special thank to my great-hearted wife, Marisete, for her encouragement and faith during my studies in Lund. I would also like to thank my nearly new born daughter, Felicia, for letting me sleep almost every night.

Contents

1. Introduction

1.1 Goal

Networks for industrial communication are usually some kind of fieldbus. The common characteristics for these networks are high reliability, low data throughput, and a high price tag. Over the years Ethernet transmission rate and communication reliability have increased. This together with serious attempts to adapt Ethernet hardware to industrial environments, make it an interesting alternative for real-time communication.

Real-time traffic in a distributed control system is usually periodic, consisting of reading of sensors values and setting of actuators. If the connecting network has a fixed time for transmitting values, the distributed control system has to be designed for this period. If it is possible to schedule real-time traffic on a Ethernet network, the periodic update frequency can be chosen more freely. This will allow designers to test different types of control systems.

This master thesis investigates if there is a possibility to run periodic real-time traffic on a switched Ethernet network. The hardware used for the network should be standard products.

1.2 Problem

The nondeterministic behavior of traditional Ethernet, caused by the CSMA/CD access control, has prohibited this type of network to run periodic real-time traffic. The CSMA/CD access control protocol is used when the media is shared. The protocol can cause collisions in the transmission. If this happens the transmission ceases and after a random amount of time a retransmission is tried.

The development of new equipment for network interconnection, i.e. Ethernet switches, has made a different approach possible. A data terminal equipment connected to a switch communicating in full-duplex, does not have to use the CSMA/CD access control. However new problems arise with the buffer memory in the switch. To prevent the switch to run out of memory space it utilizes a low level flow control. This flow control makes the switched Ethernet network to behave almost as nondeterministic as before. There is however a solution to this problem, that is to ensure that the switch never runs out of memory. This can be done if all the traffic that goes through the switch is scheduled.

2. Networking

This chapter will give some background to what network communication protocols are, and how they are used. Finally there are some examples of how networks can be interconnected to each other.

Among the large amount of books on network communication, I found two books who answered most of my questions. The first one, William Stallings book "Data & Computer Communicatio" [1], covers almost every type of communication network available today. The second book is "TCP/IP Illustrated Volume 1" by Richard Stevens [2], which describes the TCP/IP protocol suite in an excellent way.

2.1 OSI reference model

The open system interconnection (OSI) reference model was developed by the International Organization for Standardization (ISO). The final standard, ISO 7498, was published 1984. The model consists of seven layers. The following list describes the layers briefly:

Application. Provides access to the OSI environment for users and also provides distributed information services.

Presentation. Provides independence to the application processes from differences in data representation.

Session. Provides the control structure for communication between application; establishes, manages, and terminates connections between cooperating applications.

Transport. Provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control.

Network. Provides the upper layers with independence from the data transmission and switching technologies used to connect the systems; responsible for establishing, maintaining, and terminating connections.

Data link. Provides the reliable transfer of information across the physical link, sends frames with the necessary synchronization, error control, and flow control.

Physical. Concerned with the transmission of unstructed bit streams over physical media; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium.

The layers are usually referred to with numbers, starting with the physical layer as layer one.

Every layer encapsulates the data in a protocol. The encapsulation is usually done by adding a header to the data, but one layer, the data link layer, can also add a tail to the data. The physical layer is a little bit different, the protocol instead specifies a set of rules and the physical interface. The physical protocol can be divided into four specifications:

Mechanical: Specifies the pluggable connectors, signal conductors, and wiring scheme.

Electrical: Specifies the representation of bit values and transmission rates.

Functional: Specifies the functions performed between the physical interface and the transmission media.

Procedural: Specifies the sequence of events by which bit streams are exchanged across the physical medium.

Figure 2.1 shows how each layer adds and removes their protocol when application A sends data to application B.

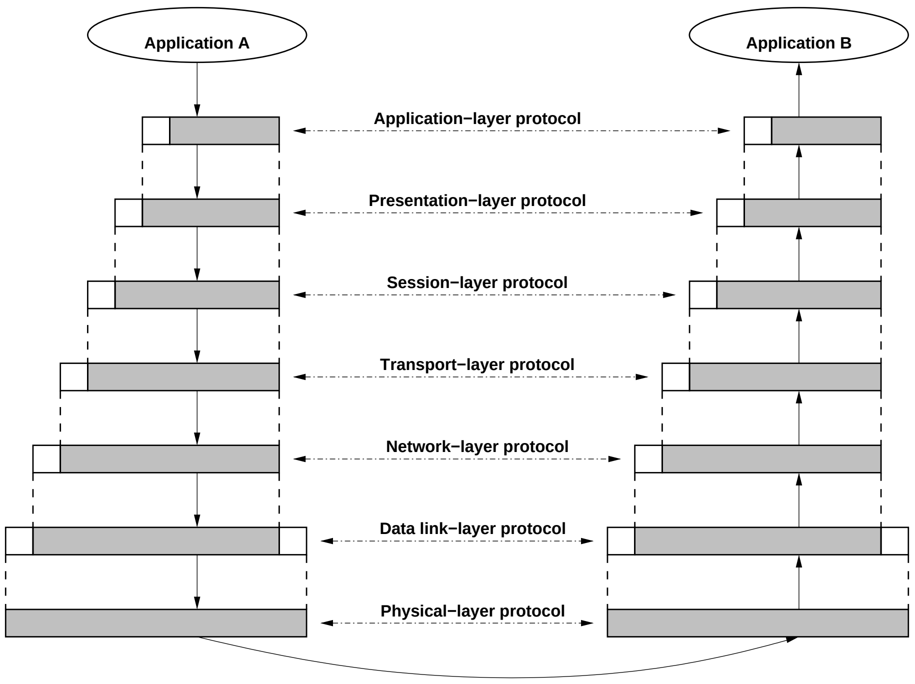

Figure 2.1 OSI reference model protocol

2.2 TCP/IP protocol suite

The TCP/IP protocol suite is a result of protocol research and development conducted on the experimental network, ARPANET, funded by the Defense Advanced Research Project Agency (DARPA). The work started in the late 1960s and has become the most used protocols for network communication. The is no official TCP/IP protocol model, as in the case of OSI. However, based on the protocol standards that have been developed, it is possible to organize the communications task into five relatively independent layers.

Application. Provides communication between processes or application on separate hosts.

Transport. Provides end-to-end data transfer service. This layer may include reliability mechanisms. It hides the details of the underlying network or networks from the application layer.

Internet. Concerned with the routing of data from source to destination host on one or more networks connected by routers.

Network access. Concerned with the logical interface between an end system and a network.

Physical. Defines the characteristics of the transmission medium, signaling rate, and signal encoding scheme.

Figure 2.2 shows a comparison between the OSI reference model and the TCP/IP model. The following list shows where in the TCP/IP protocol stack some well-known protocols are located.

Application layer: File transfer protocol (FTP), Hypertext transfer protocol (HTTP), and telnet.

Transport layer: Transmission control protocol (TCP) and User datagram protocol (UDP).

Internet layer: Internet protocol (IP).

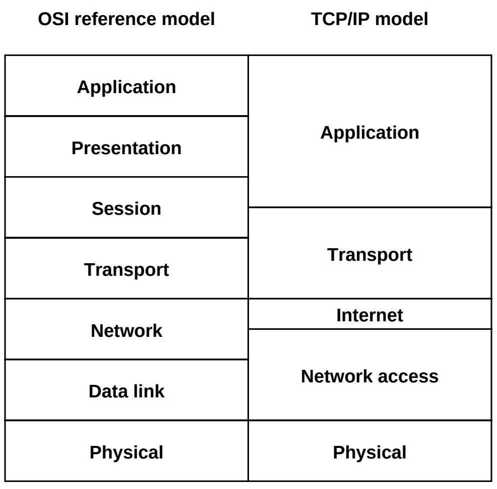

Figure 2.2 OSI reference model vs TCP/IP model

2.3 Network interconnection

This section briefly describes how local area networks (LAN), wide area networks (WAN), and data terminal equipment (DTE) can be interconnected.

Network topologies

Figure 2.3, 2.4, and 2.5 show how Data Terminal Equipment (DTE) can be connected to each other in a network. The star topology is usually interconnected with a hub or a switch. Networks of different topologies can be connected to each other using a bridge, a hub, a switch, or a router.

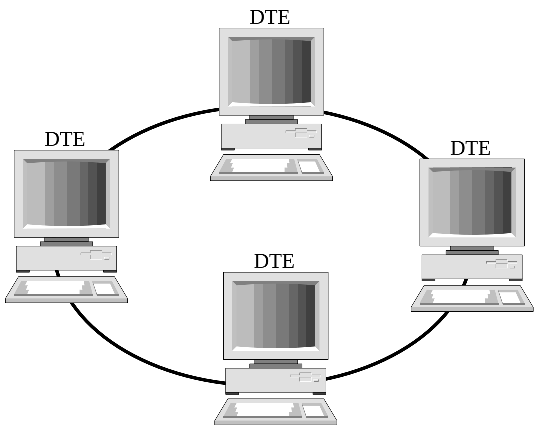

Figure 2.3 Ring topology

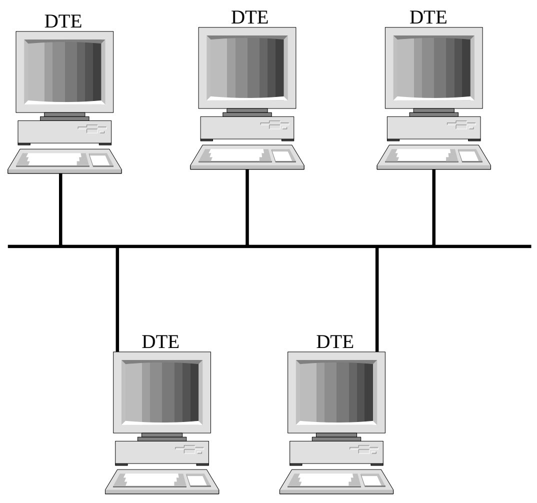

Figure 2.4 Bus topology

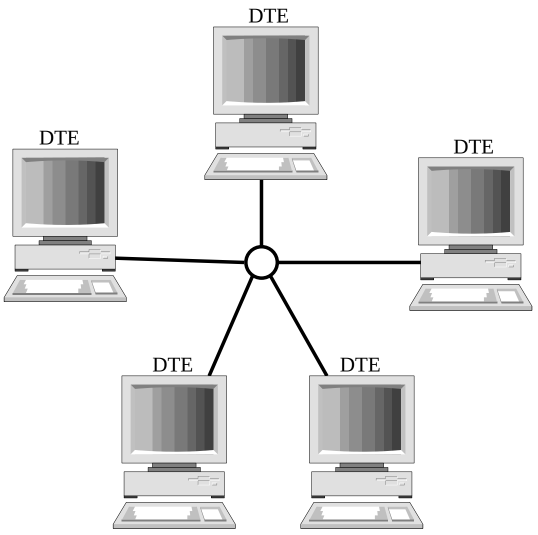

Figure 2.5 Star topology

Bridge

A bridge is primarily used for interconnecting two LANs, with the same physical layer and data link layer. Figure 2.6 shows two LANs, A and B, connected with a bridge. The bridge makes forwarding decisions on the OSI layer two. The function of the bridge can be described as:

- Read all frames transmitted on LAN A and accept those addressed to any station on LAN B. Retransmit accepted frames to LAN B.
- Read all frames transmitted on LAN B and accept those addressed to any station on LAN A. Retransmit accepted frames to LAN A.

The only problem for the bridge is to know where the stations are located. This can be done by a fixed routing table or using automatic address learning.

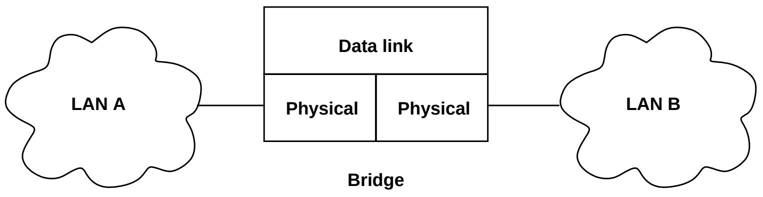

Figure 2.6 Connection of two LANs with a bridge

Router

The router is a more general purpose device, capable of interconnecting a variety of LANs and WANs. Figure 2.7 shows how two LANs are connected with each other using two routers. The router makes the routing decisions on the OSI layer three, which means the Internet layer for the TCP/IP model.

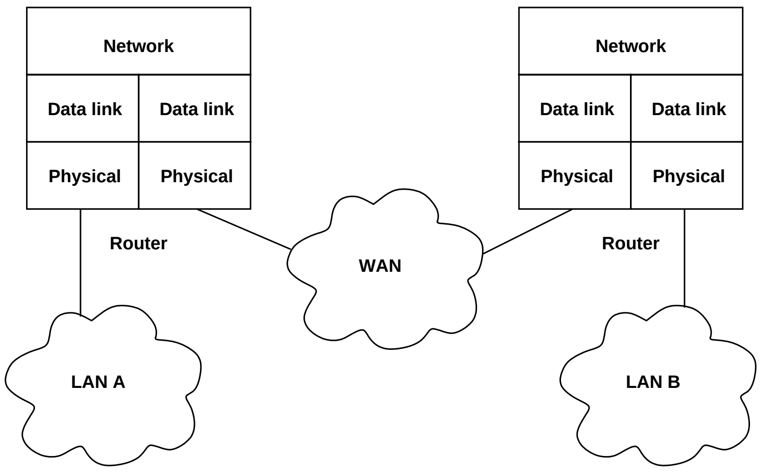

Figure 2.7 Connection of two LANs using two routers

Hub

A hub, also called multi-point repeater, is usually used to interconnect DTEs together. When the hub senses a transmission on one port, it simply takes the incoming signal and repeats or amplifies it on all the other connected ports. All the connected DTEs share the same capacity of the media, and will also share the same collision domain.

Switch

A switch can be referred to as a multi-point bridge. Forwarding decision are also made on the OSI layer two. The address learning function is usually automatic update. Unlike the hub the switch only forwards the incoming frame to all ports, if the frame is a broadcast or the switch does not know on what port the destination address is located. There are two basic transmission methods:

Cut-through switching starts sending packets as soon as they enter a switch and their destination address is read (within the first 20-30 bytes of the frame). The entire frame is not received before a switch begins forwarding it to the destination port. This reduces transmission latency between ports, but it can propagate bad packets.

Store-and-forward switching, a function traditionally performed by bridges and routers, buffers incoming packets in memory until they are fully received and a cyclic redundancy check (CRC) is run. Buffered memory adds latency to the processing time and increases in proportion to the frame size. The store-and-forward switching reduces bad packets and collisions that can adversely effect the overall performance of the segment.

The switch can use one of the two transmission methods or possibly a mixture of both. The advantages of a switch over a hub are:

- Every port on the switch has it own collision domain.
- If a switch port operates in full-duplex it can receive and transmit simultaneously.

Store-and-forward switches must buffer the frame, until the frame is retransmitted, as described above. The common approaches are:

1. Input buffering. One buffer per port.
2. Output buffering. One buffer per port.
3. Internal buffering. One memory pool used by all ports.

The third approach is probably the most popular today, since it utilizes the memory better. Low price switches may still use output buffering.

3. Ethernet

This chapter will emphasize some properties of Ethernet that are important for this work.

3.1 History

The term Ethernet used to refer to a specification published in 1982 by Digital Equipment Corp., Intel Corp., and Xerox Corp. The original Ethernet network operates at 10 Mbps and uses an access method called CSMA/CD, which stands for Carrier Sense, Multiple Access with Collision Detection.

A few years later the IEEE 802 Committee published a slightly different set of standards. The standard 802.3 covers the CSMA/CD networks, 802.4 covers token bus networks, and 802.5 covers token ring networks. Common to all these three standards is the 802.2 standard that defines the logical link control (LLC).

Ethernet is the predominant form of local area network technology used with TCP/IP today. The IEEE 802.3 standard specifies both the physical layer and the data link layer of the OSI-model. Most of the Ethernet networks today follow the IEEE 802.3 standard, but the original Ethernet frame format is usually used instead of the IEEE 802.3 frame format.

3.2 Ethernet frame

As mentioned before there are the earlier Ethernet specification and the IEEE 802.3 standard. Figure 3.1 shows the two frame formats. The frames consist of the following fields:

Preamble 7-byte pattern of alternating 1s and 0s used by the receiver to establish bit synchronization.

Start frame delimiter (SFD) The bit sequence 10101011, which indicates the actual start of the frame and enables the receiver to locate the first bit of the rest of the frame.

Destination address (DA) Specifies the station(s) for which the frame is intended. It may be a unique physical address, a group address, or a global address.

Source address (SA) Specifies the station that sent the frame.

Length Length of LLC header and data field in bytes. (Only for IEEE 802.3 frame.)

Type Ethernet type field for identifying the contents of the data field. (This field is included in the LLC header for IEEE 802.3.)

LLC header Logical link control header, i.e. IEEE 802.2 protocol.

Data The data to send (usually a IP datagram). This field has a minimum size and has to be padded if it is shorter.

Frame check sequence (FCS) A 32-bit cyclic redundancy check, based on all fields except preamble, SFD, and FCS.

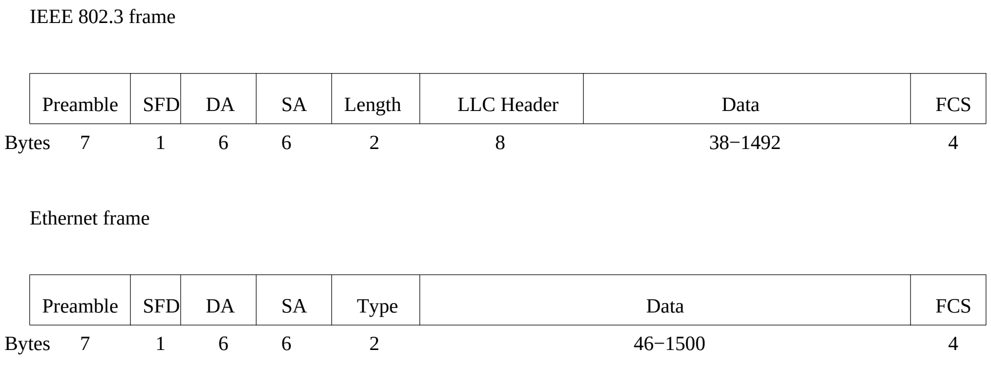

Figure 3.1 IEEE 802.3 frame and Ethernet frame

The length of both frames, excluding preamble and SFD, is between 64 and 1518 bytes. The minimum length of 64-bytes is to ensure a proper collision detect. This is discussed in the next section.

Inter frame gap (IFG) is the minimum time between two frames. This time depends on the transmission speed because it is defined as 96-bits. So for a 10 Mbps LAN it is 9.6 μs and for a 100 Mbps LAN it is 0.96 μs.

3.3 CSMA/CD access control

When using Carrier Sense, Multiple Access with Collision Detection the DTEs communicate with half-duplex, see Section 3.4. The CSMA/CD is an improvement of the CSMA access control technique. The difference is that in CSMA/CD the station continues to listen to the medium while transmitting. This leads to the following rules for CSMA/CD.

1. If the medium is idle, start transmit; otherwise go to Step 2.
2. If the medium is busy, continue listen until the channel is idle, then start transmit immediately.
3. If a collision is detected during transmission, transmit a brief jamming signal to assure that all stations know that there has been a collision and then cease to transmit.
4. After transmitting the jamming signal, wait a random amount of time, then attempt to transmit again.

To ensure that all DTEs detect a collision the segment length of the network has a maximum value. The IEEE 802.3 specifies this value for different physical layer media. The two most common used physical layer media are further described in Section 3.6 and 3.7

3.4 Half-duplex

For a Ethernet network with shared medium, the DTEs must use the CSMA/CD access control. This means that only one DTE is allowed to transmit at a time. Examples of network topologies that have shared medium are the bus topology, and the star topology interconnected with a hub.

Backpressure flow-control is very commonly used, but is is not standardized. Backpressure simply mean that the receiver sends a jamming signal when it detects an upcoming buffer overflow. The transmitting side makes attempts for retransmission after a random period of time. During this time the receiver gets some additional time for processing the frames in the buffer.

3.5 Full-duplex

Full-duplex Ethernet can be used between two DTEs. This also includes a network with star topology interconnected with a switch. Full-duplex means that transmission can be made simultaneously in both directions. This also means that the sending DTE does not have to sense that the medium is idle before transmitting.

IEEE 802.3 defines a flow-control for full-duplex Ethernet, namely the MAC Control Pause. When a DTE detects an upcoming buffer overflow, it will transmit a PAUSE control frame to the sender, requesting it to stop transmission for a certain period of time. The time is expressed as an multiple of 512 bit-times, which for a 100 Mbps LAN is equal to 5.12 μs. If sufficient buffers will become free in the meantime, the DTE can re-admit transmission by sending a PAUSE control frame with a pause duration parameter of zero to the sender. Usually the PAUSE control frames are used to turn transmission on and off, because it is difficult to calculate an appropriate pause timeout. The Pause control frame is not forwarded through switches, but can of cause propagate through switches.

The switch propagation of the pause control frame is easiest explained with an example. Figure 3.2 shows a LAN with two DTEs and two switches, with the communication speed on each segment specified. If the DTE #1 starts to send with 100% of the capacity, the buffer in switch B will become full, since the DTE #2 only can receive 10% of the capacity. Switch B will send a pause control frame to switch A, causing switch A to stop the forwarding of frames. Eventually switch As buffer will also become full, and switch A will then send a pause control frame to DTE #1.

3.6 IEEE 802.3 10BASE-T Medium specification

The IEEE 802.3 10BASE-T Medium specification is the most common 10 Mbps LAN medium specification used in office buildings today. The 10BASE-T specification defines a star topology, where the DTEs are connected to a multi point repeater, i.e. a hub or a switch. The wiring method is unshielded twisted pair cable, where two pairs are used for communication. Due to the poor transmission quality of the unshielded twisted pair, the maximum segment length is limited to 100 meters.

The encoding technique is differential Manchester. The encoding scheme for differential Manchester is the following:

- Always a transition in the middle of a interval.

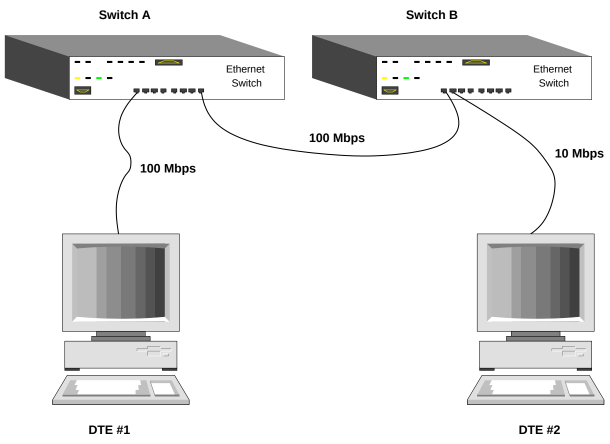

Figure 3.2 Pause control propagation

- Bit value zero leads to a transition at the beginning of a interval
- Bit value one leads to no transition at the beginning of a interval.

Figure 3.3 shows an example of differential Manchester encoding. Since there is an extra transition in the middle of a interval, the actual data rate is only 50% of the physical. To reach 10 Mbps the clock rate for the physical interface has to be 20 MHz.

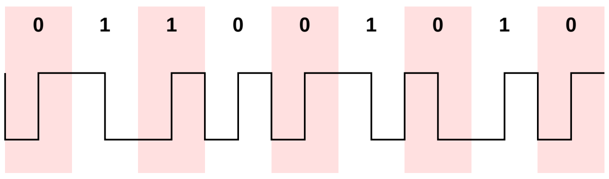

Figure 3.3 Example of differential Manchester encoding

3.7 IEEE 802.3 100BASE-TX Medium specification

The 100 Mbps specification is usually called Fast Ethernet, where the 100BASE-TX medium specification corresponds to the 10BASE-T specification for the 10 Mbps LAN. The network topology is star shaped and the wiring method is two pairs in an unshielded twisted pair cable.

Two encoding technique are used in 100BASE-TX. First the bit values are encoded with 4B5B. The 4B5B encoding takes 4 bit data and converts it into a 5 bit code. Then the resulting code is transmitted with the MLT-3 encoding. The MLT-3 encoding uses three voltage levels; a positive voltage (+V), a negative voltage (-V), and no voltage (0). The encoding scheme can be described as:

1. If the next bit value is zero, the preceding output voltage level is used.
2. If the next bit value is one, the output voltage level changes to:

- If the preceding output voltage level is +V or -V, the next output voltage changes to 0.
- If the preceding output voltage level is 0, the next output voltage changes to the opposite sign of the last output level that was not 0.

Figure 3.4 shows an example of MLT-3 encoding. The MLT-3 encoding is added to concentrate the energy in the transmitted signal below 30 MHz. This reduces the radiated emissions in the transmitted signal. To reach 100 Mbps the clock rate for the physical interface has to be 125 MHz, due to the 4B5B encoding.

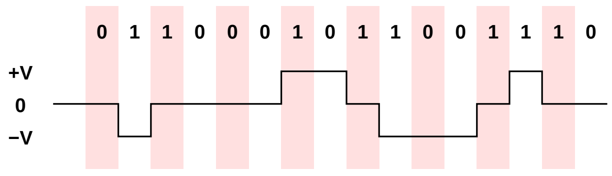

Figure 3.4 Example of MLT-3 encoding

3.8 Summary

The new Ethernet networks that are built today are usually designed to run at 100 Mbps and they usually consist of interconnected Ethernet switches, where every switch is connected to a small number of DTEs.

If it is possible to control the traffic in a simple 100 Mbps LAN, consisting only of DTEs connected to one switch, and with all the DTEs using full duplex communication, it should be possible to send periodic real-time traffic between the DTEs with a high frequency and with a predictable maximum latency. By controlling the traffic, it should be possible to avoid the low level flow control in the switch from being activated.

4. Test program

4.1 Introduction

The technical specification that the switch manufacturers deliver with the equipment only gives some values of performance. This is usually for 64-bytes Ethernet frames. So what about other frame sizes?

In order to find the throughput and latency for the switch you have to test it. Unable to find a test program free of charge, the only solution was to write my own program. The test program was written in C, sending UDP-packets with sockets. The concept for how the test was supposed to be done, was inspired by the RFC2889 [3].

The switch that was tested was a DES 1016D from D-Link. The transmission method for the switch is "store and forward". Eight identical nodes were connected to the switch, see Figure 4.1. Each node was equipped with a 100 Mbps Ethernet card, which communicates in full-duplex with the switch. The nodes were running Linux.

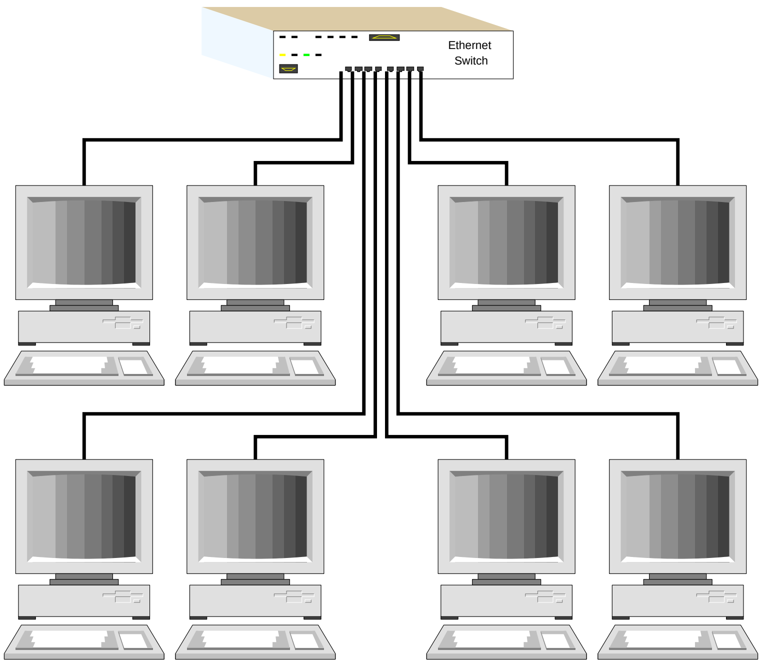

Figure 4.1 Test network

4.2 Throughput

Goal

With the throughput test it should be possible to check that the switch is capable to forward frames without performance losses.

Description

All of the nodes in Figure 4.1 are used in this test. Each node transmits and receives frames simultaneously.

Given a frame size and a number of packets to send, the nodes start to send frames to each other according to Table 4.1, without any extra delay between the frames. The total time for sending and receiving all packets is measured by reading the real-time clock in each node at transmission start and transmission end.

Table 4.1 Node transmit order

Source Node	Destination Nodes (in order of transmission)							
Node #1	2	3	4	5	6	7	8	2...
Node #2	3	4	5	6	7	8	1	3...
Node #3	4	5	6	7	8	1	2	4...
Node #4	5	6	7	8	1	2	3	5...
Node #5	6	7	8	1	2	3	4	6...
Node #6	7	8	1	2	3	4	5	7...
Node #7	8	1	2	3	4	5	6	8...
Node #8	1	2	3	4	5	6	7	1...

Result

The measured time from the throughput test is used with Equation 4.1 and 4.2 and as an reference the theoretical performance is calculated with Equation 4.3 and 4.4. The result is presented in Figure 4.2 and 4.3. The 160 extra bits in the equation origin from the sum of bits for inter frame gap (IFG), preamble, and start frame delimiter (SFD).

$$\begin{aligned} &Measured\ performance\ \ [Mbps] = \\ &= \frac{nbr\ of\ frames\ *\ frame\ size\ *\ 8}{measured\ time\ [s]\ *\ 10^6} \end{aligned} \tag{4.1}$$

$$\begin{aligned} &Measured\ performance\ \ [frames/s] = \\ &= \frac{nbr\ of\ frames}{measured\ time\ [s]} \end{aligned} \tag{4.2}$$

$$\begin{aligned} &Theoretical\ performance\ \ [Mbps] = \\ &= \frac{LAN\ speed\ [Mbps]\ *\ frame\ size\ [byte]\ *\ 8}{frame\ size\ [byte]\ *\ 8\ +\ 160\ [bits]} \end{aligned} \tag{4.3}$$

$$\begin{aligned} &Theoretical\ performance\ \ [frames/s] = \\ &= \frac{LAN\ speed\ [Mbps]\ *\ 10^6}{frame\ size\ [byte]\ *\ 8\ +\ 160\ [bits]} \end{aligned} \tag{4.4}$$

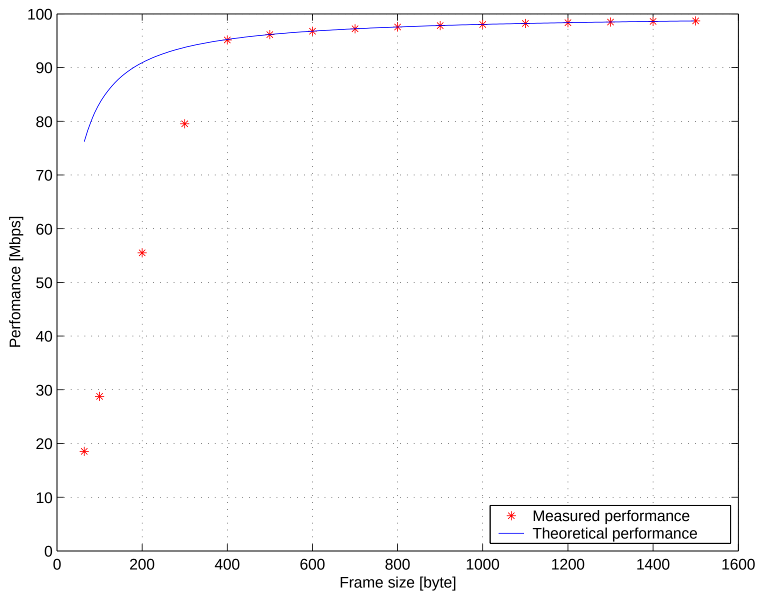

Figure 4.2 Switch performance [Mbps]

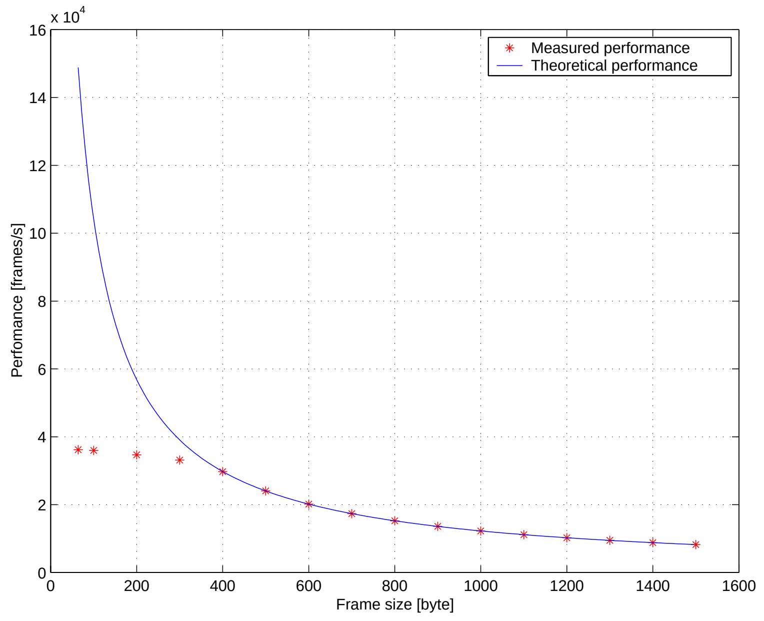

Figure 4.3 Switch performance [frames/s]

Figure 4.2 and 4.3 both show that for frame sizes less than 400 byte the measured performance is less than the theoretical performance. This does however not depend on the switch. The cpu usage when running the test program with small frames is close to 100%. The test node performance with small frames becomes a bottle neck, and the measurement is not only showing the switch performance. One reason for why the cpu usage increases is the context switches. Since small frames come more frequent there will also be more frequent context switches.

With frame size over 400 bytes the switch performs as it should. So the conclusion is that the switch throughput is as good as it can be.

4.3 Latency

Goal

The purpose of this test is to verify that the average latency sending frames through the switch is not deviating too much from what could be expected.

Description

Only two of the nodes are used in this, one node is selected as a sender and one node is selected as a receiver. Figure 4.4 shows the reduced network. The real-time clocks in the two nodes are synchronized immediately before a new test run is started.

Given a transmit rate, frame size and a number of packets to send, the sending node starts to transmit frames. The frame includes the time just before the transmission. When the receiving node receives the frame it calculates the latency for the frame and saves the value. After receiving all the frames the receiver calculates the average latency.

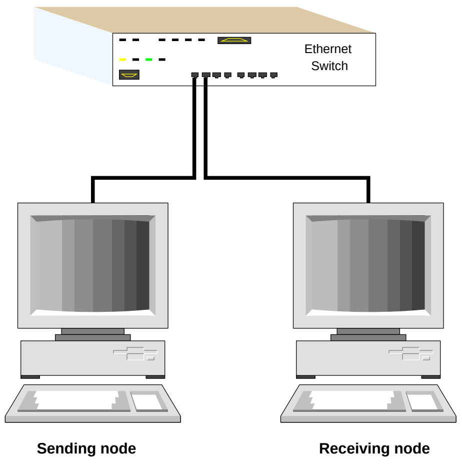

Figure 4.4 Test network for latency test

Result

The transmit rate for the test was one percentage of maximum rate. The test result is presented in Figure 4.5. As a reference a minimum latency is calculated with Equation 4.7 and included in Figure 4.5.

The transmission method for the switch is "store-and-forward", causing the factor 2 for LAN transmit time. The frame is also copied twice on the PCI-bus, one time in the sender and one time in the receiver. The 160 extra bits in the equations origin from the sum of bits for inter frame gap (IFG), preamble, and start frame delimiter (SFD).

$$\begin{aligned} &\textit{LAN transmit time}\ \ [\mu s] = \\ &= \frac{\textit{frame size}\ [\textit{byte}]\ *\ 8\ +\ 160\ [\textit{bits}]}{\textit{LAN speed}\ [\textit{Mbps}]} \end{aligned} \tag{4.5}$$

$$\begin{aligned} &\textit{PCI-bus copy time}\ \ [\mu s] = \\ &= \frac{\textit{frame size}\ [\textit{byte}]}{133\ [\textit{Mbps}]} \end{aligned} \tag{4.6}$$

$$\begin{aligned} &\textit{Minimum latency}\ \ [\mu s] = \\ &= 2 * \textit{LAN transmit time}\ [\mu s]\ +\ 2 * \textit{PCI-bus copy time}\ [\mu s] \end{aligned} \tag{4.7}$$

Figure 4.5 shows that the difference between measured average latency and minimum latency increases with larger frames. The trend for this extra latency can be expressed as:

$$\textit{Extra latency}\ = C_1 + C_2 * \textit{frame size}$$

The term "$C_2 * \textit{frame size}$" is probably caused by memory copy in the test node. The memory copy is introduced when the frame is passed through

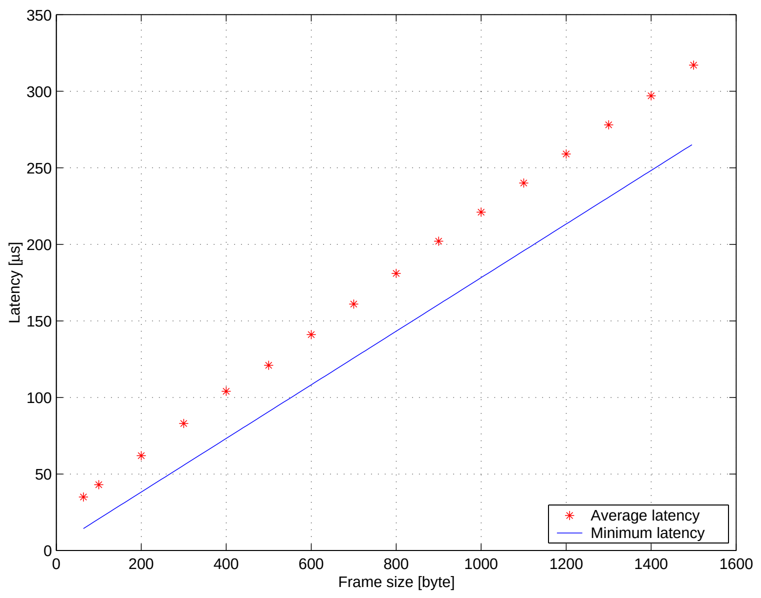

Figure 4.5 Switch latency [μs]

the UDP- and IP-layers in the test node. The term “C_1” is constant for every frame size and depends mainly on the necessary context switches when the frame is sent and received.

The conclusion is that the switch does not add more latency to the transmission than the transmission time caused by the "store-and-forward" function.

4.4 Summary

Both the throughput test and latency test show that the DES 1016D from D-Link performs as expected. The throughput test also shows that the node can become a weak link if there is a lot of traffic with small frame sizes.

5. Scheduling

5.1 Introduction

The goal for scheduling the traffic in a switched Ethernet LAN is to let ordinary traffic, such as telnet, ftp, and http traffic, coexist with the periodic real-time traffic. This means that the ordinary traffic has to be restricted so it does not interfere with the periodic real-time traffic. By this you can say that the ordinary traffic is scheduled as well.

The intention was to investigate different scheduling approaches, but the lack of time prevented the intention. So this chapter only investigates one type of scheduling, the worst case scheduling.

5.2 Definitions

This is a list of terms that are used in this chapter. The purpose of the list is to clearly define important term so there is no confusion.

Scheduler: See NetGuard.

NetGuard: The node which schedules the RT traffic. Since it handles more tasks than the scheduling, it is called the NetGuard. The following list contains example of the tasks that the NetGuard has to handle:

- Schedule the requested RT traffic.
- Update the nodes when the schedule is changed. If the schedule can change dynamically, the NetGuard has to send the new schedule to the nodes when it changes.
- Keep track of nodes connected to the LAN. The NetGuard has to check and update a list of nodes, that are connected to the network.
- Act as a router for non RT traffic. See non RT traffic.
- Keep a global real-time clock for the nodes. The global real-time clock can be used for clock synchronization.
- Convert broadcast to unicast. When the NetGuard routes a broadcast into the scheduled network, it is transmitted to all the connected nodes with their unique address. The transmission is controlled so it does not interfere with the scheduled RT traffic.

Node: A data terminal equipment connected to the LAN.

RT traffic: Scheduled periodic real-time traffic between the nodes.

Non RT traffic: Scheduled ordinary traffic, such as telnet, ftp, and http traffic. The non RT traffic is either sent or received by the NetGuard.

RTC: Periodic real-time channel, a collection of properties describing the RT traffic, used by the NetGuard to schedule the traffic. The following list is a minimum of properties needed for the schedule, but a implemented version of the list may look different.

- Send node. Can be identified by Ethernet address or IP-address.
- Receive node. Can be identified by Ethernet address or IP-address.
- Transmit time [μs]. The transmit time includes all overhead, also inter frame gap, preamble, and start frame delimiter. The minimum network latency, using a store-and-forward switch, is the transmit time multiplied by two.
- Frequency [Hz]. The frequency for the periodic update.
- Maximum latency [μs]. The maximum allowed network latency.

Minimum transmit time: Transmit time for a 64-byte Ethernet frame, including all overhead.

Maximum transmit time: Transmit time for a 1518-byte Ethernet frame, including all overhead.

5.3 Worst case scheduling

Assume that the RT traffic is forwarded to the network with the single restriction that the time between the frames has to be at least the period that is requested. This means that when a node sends a RTC frame it simply calculates the next time for this channel using Equation 5.1. Thus it is possible that all the RT traffic is forwarded at the same time. This is also known as the worst case. The scheduler only has to verify that the worst case is within the limits that the RTC specifies.

$$\begin{aligned} &RTC[j].Next\ send\ time = \\ &= Current\ time\ +\ \frac{10^6}{RTC[j].Frequency} \end{aligned} \tag{5.1}$$

By using worst case scheduling there is actually no need for synchronizing the nodes. This is of course only possible if the real-time clocks in the nodes are not drifting too much.

Example

It is always easier to understand using an example. Figure 5.1 shows how the network is interconnected. All the nodes and the NetGuard, are communicating with 100 Mbps in full-duplex with the switch. The transmission method for the switch is store-and-forward. This gives the following values for the network:

$$\begin{aligned} &Minimum\ transmit\ time = 6.72\mu s \approx 7\mu s \\ &Maximum\ transmit\ time = 123.04\mu s \approx 123\mu s \end{aligned}$$

Table 5.1 specifies an example of RT traffic that is used throughout this chapter.

Table 5.1 Requested RT traffic

	RTC[1]	RTC[2]	RTC[3]	RTC[4]
Send node	Node[1]	Node[2]	Node[2]	Node[4]
Receive node	Node[3]	Node[3]	Node[4]	Node[1]
Transmit time [μs]	50	50	10	40
Frequency [Hz]	1000	1000	10000	5000
Maximum latency [μs]	500	500	100	350
Ethernet frame size [$byte$]	605	605	105	480

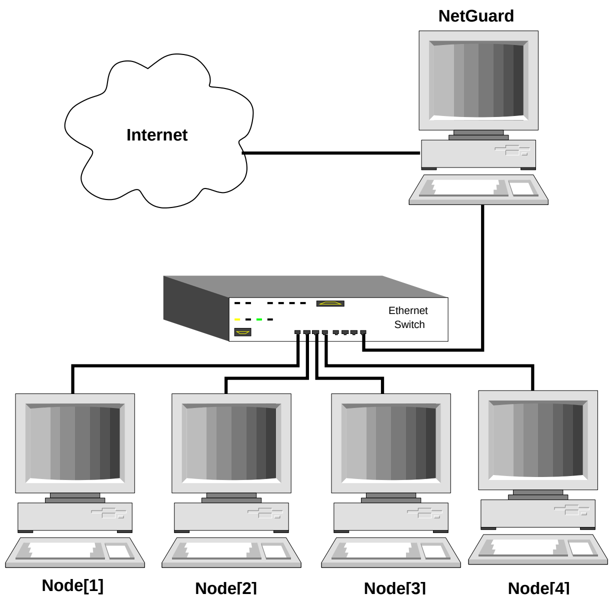

Figure 5.1 Network interconnection

5.4 Periodic update constraint

The minimum check to do for a worst case schedule, is that is possible so send and receive all the RT traffic at all. In addition to this the node has to be able to communicate with the NetGuard.

The worst case period is defined as the shortest periodic update time for sending and receiving. This gives us Equation 5.2 and 5.3. The amount of traffic that can be forwarded in the worst case period is defined as worst case duration. The worst case duration is calculated with Equation 5.4 and 5.5.

$$node[i].Worst\ case\ send\ period\ [\mu s] = \tag{5.2}$$
$$= \min_{\forall j | RTC[j].send\ node = node[i]} \left(\frac{10^6}{RTC[j].Frequency} \right)$$

$$node[i].\mathit{Worst\ case\ receive\ period}\ [\mu s] = \tag{5.3}$$
$$= \min_{\forall j | RTC[j].\mathit{receive\ node}=node[i]} \left(\frac{10^6}{RTC[j].\mathit{Frequency}} \right)$$

$$node[i].\mathit{Worst\ case\ send\ duration}\ [\mu s] = \tag{5.4}$$
$$= \sum_{\forall j | RTC[j].\mathit{send\ node}=node[i]} \Big(RTC[j].\mathit{Transmit\ time} \Big)$$

$$node[i].\mathit{Worst\ case\ receive\ duration}\ [\mu s] = \tag{5.5}$$
$$= \sum_{\forall j | RTC[j].\mathit{receive\ node}=node[i]} \Big(RTC[j].\mathit{Transmit\ time} \Big)$$

With the equations above it is possible to calculate how much unused time there is available, with Equation 5.6 and 5.7. To ensure the NetGuard communication the free periodic duration has to be greater than the minimum transmit time.

$$node[i].\mathit{Free\ periodic\ send\ duration}\ [\mu s] = \tag{5.6}$$
$$= node[i].\mathit{Worst\ case\ send\ period} - node[i].\mathit{Worst\ case\ send\ duration}$$

$$node[i].\mathit{Free\ periodic\ receive\ duration}\ [\mu s] = \tag{5.7}$$
$$= node[i].\mathit{Worst\ case\ receive\ period} - node[i].\mathit{Worst\ case\ receive\ duration}$$

Example

Table 5.2 shows the calculated values for the requested RT traffic, using the equations above. The ∞ for the period actually means that there is no requested RT traffic. All the free periodic durations are greater than the minimum transmit time, so the requested RT traffic passes the periodic update constraint.

Table 5.2 Free periodic duration for the requested RT traffic. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Worst case send period	1000	100	∞	200
Worst case receive period	200	∞	1000	100
Worst case send duration	50	60	0	40
Worst case receive duration	40	0	100	10
Free periodic send duration	950	40	∞	160
Free periodic receive duration	160	∞	900	90

5.5 Maximum latency constraint

At a quick glance the network latency for a RT frame, sent from node #1 to node #2, should be the worst case send duration for node #1 and the worst case receive duration for node #2. The worst case send duration means, in this case, that all the scheduled RT traffic was sent at the same time and is waiting in the output buffer for node #1. The frame we are looking at is the last one in this queue. When the frame we are looking at finally is transmitted, it arrives to the switch buffer at the same time as all the scheduled RT traffic to node #2 arrives. Let's assume that the frame we are looking at is the last one in the switch buffer queue. The time before our frame arrives to node #2, is of course the worst case receive duration for node #2. Figure 5.2 shows that there are exceptions from this assumption. In this case it is because there is multiple traffic between node #1 and node #2, so the network latency for frame A is not influenced by frame B and C in the switch buffer.

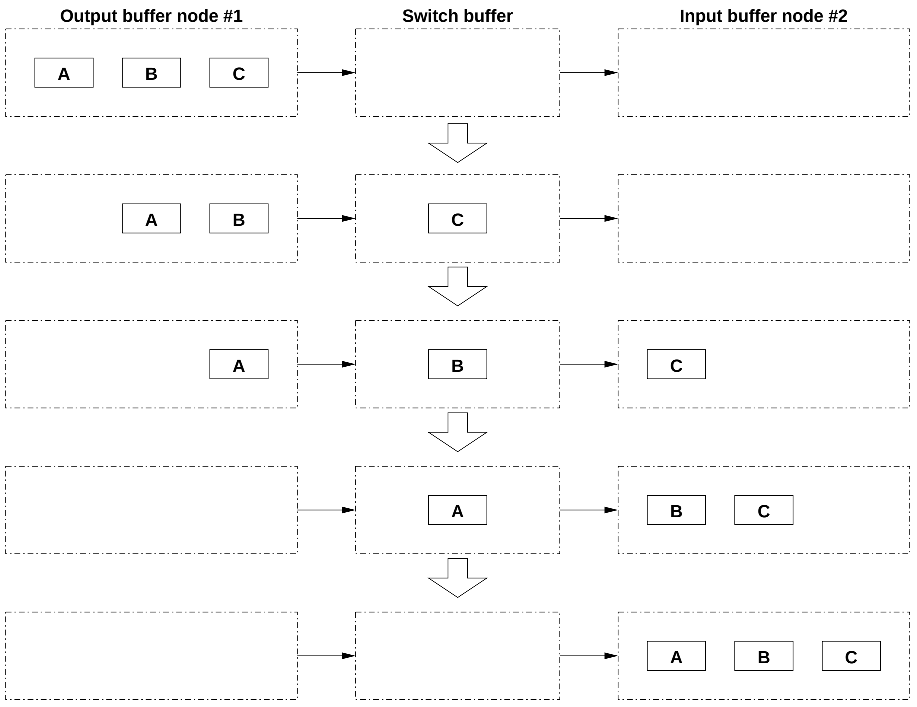

Figure 5.2 Example of switch latency

It becomes even more difficult to calculate a correct maximum latency, if there are intermixed RT frames from other nodes. So if this phenomenon is ignored, and the network latency is calculated as the sum of the worst case send duration and the worst case receive duration, it is possible to calculate a worst case network latency for each RT frame. By subtracting the worst case network latency from the allowed maximum latency, we get the unused duration. Equation 5.8 gives us the available latency duration for each RT channel.

$$\begin{aligned}&RTC[j].Available\ latency\ duration\ [\mu s] = \\ &= RTC[j].Maximum\ latency - \\ &\quad node[RTC[j].send\ node].Worst\ case\ send\ duration - \\ &\quad node[RTC[j].receive\ node].Worst\ case\ receive\ duration\end{aligned} \tag{5.8}$$

The RTC available latency duration can be used in the sending node or in the receiving node. To ensure the NetGuard communication, the assigned duration has to be greater than the minimum transmit time. To really calculate the correct free latency duration, a two-step procedure is used.

The first step calculates the free latency duration for all the sending nodes, with Equation 5.9. The division by two is done to divide the available latency duration equally between the sending node and the receiving node.

$$\begin{aligned}&node[i].Free\ latency\ send\ duration\ [\mu s] = \\ &= \min_{\forall j | RTC[j].send\ node=node[i]} \left(\frac{RTC[j].Available\ latency\ duration}{2} \right)\end{aligned} \tag{5.9}$$

The second step calculates the free latency duration for all the receiving nodes, with Equation 5.10. This step just assigns what is left of the available latency duration to the receiving node, and selects the worst case.

$$\begin{aligned}&node[i].Free\ latency\ receive\ duration\ [\mu s] = \\ &= \min_{\forall j | RTC[j].recive\ node=node[i]} \Big(RTC[j].Available\ latency\ duration - \\ &\quad node[RTC[j].send\ node].Free\ latency\ send\ duration \Big)\end{aligned} \tag{5.10}$$

Example

Table 5.3 shows the calculated values for the requested RT traffic, using Equation 5.8.

Table 5.3 Available latency duration for the requested RT traffic. Unit: μs

	RTC[1]	RTC[2]	RTC[3]	RTC[4]
Maximum latency	500	500	100	350
Worst case send duration	50	60	60	40
Worst case receive duration	100	100	10	40
Available latency duration	350	340	30	270

Table 5.4 shows the free latency duration for the different nodes with the requested RT traffic. All the free latency durations are greater than the minimum transmit time, so the requested RT traffic passes the maximum latency constraint.

Table 5.4 Free latency duration for the requested RT traffic. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Free latency send duration	175	15	∞	135
Free latency receive duration	135	∞	175	15

5.6 NetGuard communication

Now the scheduler has to decide how much time there is available for the NetGuard communication, e.g. the non RT traffic. This means that the maximum time for NetGuard communication can be calculated with Equation 5.11 and 5.12. The time has an upper bound in the maximum transmit time.

$$node[i].Node\ send\ time\ [\mu s] = \\ = \min \begin{pmatrix} Maximum\ transmit\ time, \\ node[i].Free\ latency\ send\ duration, \\ node[i].Free\ periodic\ send\ duration \end{pmatrix} \tag{5.11}$$

$$node[i].NetGuard\ send\ time\ [\mu s] = \\ = \min \begin{pmatrix} Maximum\ transmit\ time, \\ node[i].Free\ latency\ receive\ duration, \\ node[i].Free\ periodic\ receive\ duration \end{pmatrix} \tag{5.12}$$

Example
Table 5.5 shows the maximum time for the NetGuard communication. Only two of the values are less than the maximum transmit time. Not so surprisingly it is the RTC[3] who causes this, which is sent from node #2 and received by node #4. This can however be improved which will be investigated in the next section. Figure 5.3 shows the network latency for RTC[3] with the worst case traffic between node #2 and node #4.

Table 5.5 NetGuard communication for the requested RT traffic. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Node send time	123	**15**	123	123
NetGuard send time	123	123	123	**15**

Finally we can calculate the worst case network latency, as the sum of the worst case duration and the maximum values for the NetGuard communication. Table 5.6 show the calculated values for each RT channel.

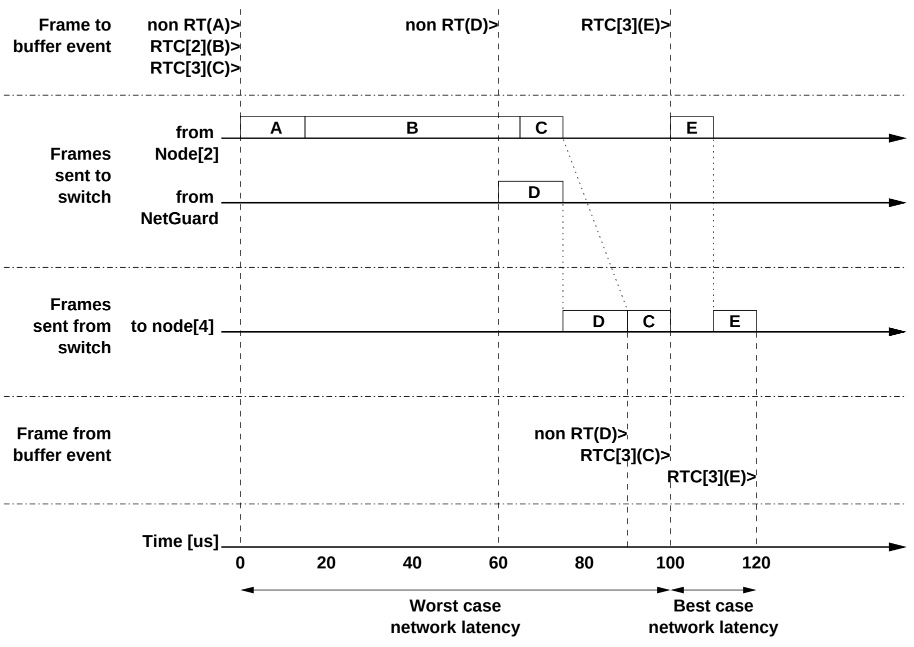

Figure 5.3 Network latency for RTC[3]

Table 5.6 Worst case network latency for the requested RT traffic. Unit: μs

	RTC[1]	RTC[2]	RTC[3]	RTC[4]
Worst case send duration	50	60	60	40
Worst case receive duration	100	100	10	40
Node send time	123	15	15	123
NetGuard send time	123	123	15	123
Worst case network latency	396	298	100	326

5.7 Fragmentation of the RT traffic

There are two reasons for fragmentation of the RT traffic:

1. The requested RT traffic does not pass the constraints stated earlier.
2. To improve NetGuard communication abilities.

Fragmentation can not be seen as a pure advantage. One of the drawbacks is that the overhead in the transmission increases. If the header for handling the fragmentation is 20 bytes, the overhead time in the example will become.

$$\textit{Overhead time} = 4.64\mu s \approx 5\mu s$$

Another drawback is that each fragment causes extra interrupts in the sending and receiving nodes. This will increase the system load for the nodes, and maybe jeopardize the whole RT function.

The equations in the periodic update constraint are still valid if, for the fragmented RT traffic, the values for frequency and transmit time are substituted with fragmentation frequency and fragment transmit time. The fragmentation transmit time is calculated with Equation 5.13. For the maximum latency constraint, Equation 5.8 has to be substituted with Equation 5.14, for the fragmented RT traffic.

$$\begin{aligned} &RTC[j].\mathit{Fragment\ transmit\ time}\ [\mu s] = \\ &= \frac{RTC[j].\mathit{Transmit\ time} - \mathit{Overhead\ time}}{RTC[j].\#\mathit{fragment}} + \mathit{Overhead\ time} \end{aligned} \tag{5.13}$$

$$\begin{aligned} &RTC[j].\mathit{Available\ latency\ duration}\ [\mu s] = \\ &= RTC[j].\mathit{Maximum\ latency} - \\ &\quad (RTC[j].\#\mathit{fragment} - 1) * \frac{10^6}{RTC[j].\mathit{Fragment\ frequency}} - \\ &\quad \mathit{node}[RTC[j].\mathit{send\ node}].\mathit{Worst\ case\ send\ duration} - \\ &\quad \mathit{node}[RTC[j].\mathit{receive\ node}].\mathit{Worst\ case\ receive\ duration} \end{aligned} \tag{5.14}$$

First fragmentation example

Let go back to the example. Assume that RTC[2] is fragmented three times with the fragmentation frequency 10 kHz. Table 5.7 shows the free periodic duration, using the transmit time for RTC[2] calculated in Equation 5.15. Notice that the fragmentation frequency for RTC[2], will also change the worst case receive period for node #3.

$$\begin{aligned} &RTC[2].\mathit{Fragment\ transmit\ time} = \\ &= (50\ -\ 5)/3\ +\ 5 = 20\mu s \end{aligned} \tag{5.15}$$

Table 5.7 Free periodic duration for the requested RT traffic, with RTC[2] fragmented three times at 10kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Worst case send period	1000	100	∞	200
Worst case receive period	200	∞	**100**	100
Worst case send duration	50	**30**	0	40
Worst case receive duration	40	0	**70**	10
Free periodic send duration	950	**70**	∞	160
Free periodic receive duration	160	∞	**30**	90

The available latency duration for RTC[2] will decrease due to the fragmentation. The new available latency duration is calculated in Equation 5.16. Table 5.8 shows the new values for all the free latency durations.

$$RTC[2].Available\ latency\ duration = \\ = 500 - (3-1) * 100 - 30 - 70 = 200\mu s \tag{5.16}$$

Table 5.8 Free latency duration for the requested RT traffic, with RTC[2] fragmented three times at 10kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Free latency send duration	**190**	**30**	∞	135
Free latency receive duration	135	∞	**170**	**30**

The final result for trying to improve the NetGuard communication is presented in Table 5.9. The fragmentation manages to increase the NetGuard communication for node #2 and #4, but it also decreases the NetGuard send time for node #3.

Table 5.9 NetGuard communication for the requested RT traffic, with RTC[2] fragmented three times at 10kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Node send time	123	**30**	123	123
NetGuard send time	123	123	**30**	**30**

The worst case network latency is calculated in Table 5.10, where the worst case send duration for RTC[2] is calculated as:

$$RTC[2].Worst\ case\ send\ duration = \\ = (3-1) * 100 + 30 = 230\mu s$$

Table 5.10 Worst case network latency for the requested RT traffic, with RTC[2] fragmented three times at 10kHz. Unit: μs

	RTC[1]	RTC[2]	RTC[3]	RTC[4]
Worst case send duration	50	230	30	40
Worst case receive duration	70	70	10	40
Node send time	123	30	30	123
NetGuard send time	30	30	30	123
Worst case network latency	273	360	100	326

Second fragmentation example

The first attempt to improve NetGuard communication was maybe not the best. If the fragmentation of RTC[2] instead is two times with the frequency 5kHz, Table 5.11 shows the new values for free periodic duration, using the transmit time for RTC[2] calculated with Equation 5.17.

$$RTC[2].Fragment\ transmit\ time = \\ = (50 - 5)/2 + 5 \approx 28\mu s \tag{5.17}$$

Table 5.11 Free periodic duration for the requested RT traffic, with RTC[2] fragmented two times at 5kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Worst case send period	1000	100	∞	200
Worst case receive period	200	∞	**200**	100
Worst case send duration	50	**38**	0	40
Worst case receive duration	40	0	**78**	10
Free periodic send duration	950	**62**	∞	160
Free periodic receive duration	160	∞	**122**	90

The available latency duration for RTC[2] can now be calculated with Equation 5.18. All the values for free latency duration are presented in Table 5.12.

$$RTC[2].Available\ latency\ duration = \\ = 500 - (2 - 1) * 200 - 38 - 70 = 192\mu s \tag{5.18}$$

Table 5.12 Free latency duration for the requested RT traffic, with RTC[2] fragmented two times at 5kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Free latency send duration	**181**	**26**	∞	135
Free latency receive duration	135	∞	**168**	**26**

Table 5.13 shows the final result for the second attempt to improve the NetGuard communication. The improvement for node #2 and #4 is not as good as in the first attempt, but the NetGuard send time for node #3 is a lot better.

The worst case network latency is calculated in Table 5.14, where the worst case send duration for RTC[2] is calculated as:

$$RTC[2].Worst\ case\ send\ duration = \\ = (2 - 1) * 200 + 38 = 238\mu s$$

Table 5.13 NetGuard communication for the requested RT traffic, with RTC[2] fragmented two times at 5kHz. Unit: μs

	node[1]	node[2]	node[3]	node[4]
Node send time	123	**26**	123	123
NetGuard send time	123	123	**122**	**26**

Table 5.14 Worst case network latency for the requested RT traffic, with RTC[2] fragmented two times at 5kHz. Unit: μs

	RTC[1]	RTC[2]	RTC[3]	RTC[4]
Worst case send duration	50	238	38	40
Worst case receive duration	78	78	10	40
Node send time	123	24	26	123
NetGuard send time	122	122	26	123
Worst case network latency	373	462	100	326

5.8 Traffic control

Next time for RT traffic

If there is fragmented RT traffic the simple restriction stated in Equation 5.1, has to be modified. The modification can be expressed in three steps.

1. When the node sends the first fragment, the current time is saved for later use. The next time is then calculated with Equation 5.20

$$RTC[j].Last\ time = Current\ time \tag{5.19}$$

2. When the node sends a fragment except the first and the last, the next time is calculated with Equation 5.21
3. When the node sends the last fragment, the next time is calculated with Equation 5.22

$$RTC[j].Next\ time = Current\ time + \frac{10^6}{RTC[j].Fragmentation\ frequency} \tag{5.20}$$

$$RTC[j].Next\ time = RTC[j].Next\ time + \frac{10^6}{RTC[j].Fragmentation\ frequency} \tag{5.21}$$

$$RTC[j].Next\ time = RTC[j].Last\ time + \frac{10^6}{RTC[j].Frequency} \tag{5.22}$$

Next time for NetGuard communication

The calculation of the next time for the NetGuard communication is more complicated than for the RT traffic. First the bandwidth for NetGuard communication has to be divided among the nodes. The reason for doing this is to ensure that not too much non RT traffic is forwarded. This leads to Equation 5.23. The NetGuard can change the division by allowing the nodes to request a preferred fraction of the bandwidth. The best fairness is to divide the bandwidth equally.

$$\begin{aligned} \sum_{\forall i} node[i].Node\ bandwidth\ fraction\ &\leq\ 1 \\ \sum_{\forall i} node[i].NetGuard\ bandwidth\ fraction\ &\leq\ 1 \end{aligned} \tag{5.23}$$

Another problem when calculating the next time is that the non RT traffic does not have a specific transmit time. A non RT traffic frame can either have longer or shorter transmit time than the scheduled send time. If a frame has longer transmit time than the send time it has to be fragmented, before it is sent. By considering the actual time it takes to send a frame (fragmented or not) and to ensure that the node or the NetGuard does not forward too much traffic, the next time for non RT traffic can be calculated with Equation 5.24 and 5.25. The first part of the maximum expression ensures that the worst case RT traffic can pass before new non RT traffic is sent. The second part ensures that not more than the allowed bandwidth is used.

$$node[i].Node\ next\ send\ time = Buffer\ free\ time + \max\left(\begin{array}{c} \left(\begin{array}{c} Frame\ transmit\ time\ + \\ node[i].Worst\ case\ send\ duration \end{array} \right), \\ \\ \left(\dfrac{Frame\ transmit\ time}{node[i].Node\ bandwidth\ fraction} \right) \end{array} \right) \tag{5.24}$$

$$node[i].NetGuard\ next\ send\ time = Buffer\ free\ time + \max\left(\begin{array}{c} \left(\begin{array}{c} Frame\ transmit\ time\ + \\ node[i].Worst\ case\ receive\ duration \end{array} \right), \\ \\ \left(\dfrac{Frame\ transmit\ time}{node[i].NetGuard\ bandwidth\ fraction} \right) \end{array} \right) \tag{5.25}$$

The buffer free time used in the Equation 5.24 and 5.25, means the time when the output buffer on the network card is empty. For every frame that is sent the buffer free time is updated in two steps. Equation 5.26 is used before the calculation of next time and Equation 5.27 is used after.

$$\begin{aligned}&\textit{Buffer free time} = \\ &= \max(\textit{Buffer free time, Current time})\end{aligned} \tag{5.26}$$

$$\begin{aligned}&\textit{Buffer free time} = \\ &= \textit{Buffer free time} \; + \; \textit{Frame transmit time}\end{aligned} \tag{5.27}$$

5.9 Summary

The simple restriction for forwarding the frames stated in Section 5.8, is the result of the worst case scheduling. This control has to be implemented in every node connected to the switch and in the NetGuard. Design ideas and implementation problem are discussed in the next chapter.

The observant reader has noticed that there is no algorithm for the fragmentation of the RT traffic. To develop an algorithm you have to decide which parameter you what to optimize. The two fragmentation example shows that the NetGuard communication can be improved. There is however a dilemma to this improvement. The NetGuard does not know if there really is any non RT traffic that needs to be improved. If the NetGuard had more information about the amount of non RT traffic that can be expected to be sent and received by the nodes, a better optimized decision can be made. I wish there was more time to investigate the problem more thoroughly, but for now I can only postpone the problem.

The worst case scheduling theory has some properties that is interesting for distributed control system. The following list of "pros and cons" summarizes these properties:

+ The frequency for periodic updates can be chosen more freely for distributed control system.
+ The system load for forwarding network traffic is not so high.
+ Synchronization is not so important. Since the worst case is allowed the frames can be sent without mutual synchronization.

– The network latency is not constant.
– Mixing high and low frequencies for periodic updates, could lead to poor network utilization.

6. Test implementation

The intention with the test implementation was to test the worst case scheduling before a real version was implemented. Unfortunately there was no time to make a complete test implementation. Therefore this chapter will focus on the design ideas and implementation problems.

6.1 Introduction

We assume that all applications running in the nodes use the TCP/IP protocol suite for network communication. The traffic control for the worst case scheduling can then be implemented as an extra layer. Let us denote the extra layer as the RT-layer. The RT-layer is added between the Internet-layer and the Ethernet hardware interface, see Figure 6.1. By adding the RT-layer an application running in the node does not have to be modified.

The RT-layer also adds a header when an IP-frame is forwarded to the Ethernet-layer. The header is mainly used for handle the fragmentation caused by the worst case scheduling traffic control.

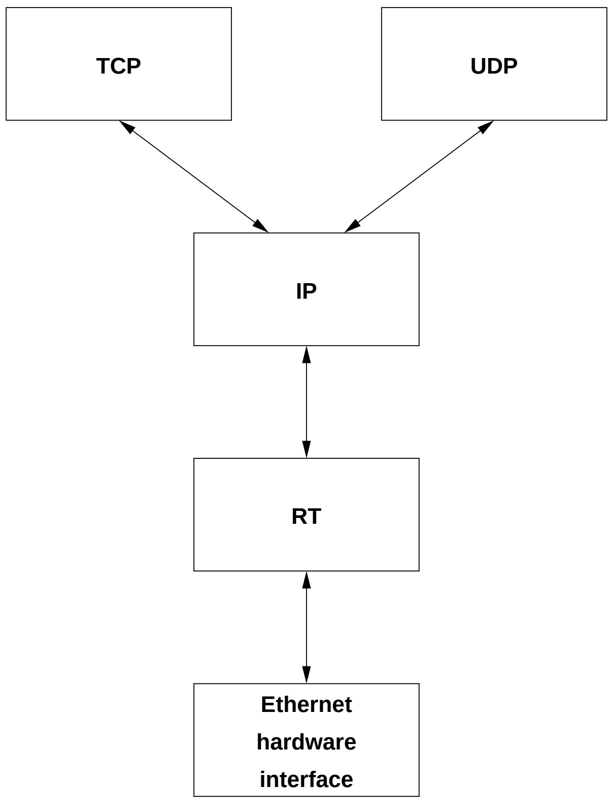

Figure 6.1 Protocol layer model

6.2 Fragmentation of the RT traffic

Since the previous chapter did not include any algorithm for fragmentation of the RT traffic, the fragmentation decision has to be made manually. It can be done by adding the fragmentation frequency and the number of fragment to the RT traffic request. The advantage by doing the fragmentation decision manually, is that the impact on the target system can be tested in a controlled way.

6.3 RT-layer

In the RT-layer there is a set of send channels and receive channels. The NetGuard assigns one send channel and one receive channel for each RT traffic request. Since the RT-layer has to identify if the IP-frame is scheduled or not, the request for RT traffic to the NetGuard has to include:

- Source IP-address
- Destination IP-address
- Source port
- Destination port

When the NetGuard changes the schedule due to a new request, the RT-layer has to receive the new schedule. This can be solved by sending a predefined frame to the nodes, and when the frame passes the RT-layer, the information is extracted.

Since the traffic control for non RT traffic is different from the traffic control for the RT traffic, the RT-layer in the NetGuard looks a little bit different. In the NetGuard RT-layer all send channels are used for non RT traffic and in the node RT-layer only one send channel is used for non RT traffic. The easiest thing to do is to implement two types of RT-layer, but this is maybe not so appealing.

Frame buffers

The RT-layer needs to be able to buffer frames, both when sending and receiving frames. When the RT-layer receives frames it needs one buffer per receive channel, due to the fragmentation ability. The send channels need at least one buffer per channel, due to the traffic control which includes fragmentation.

No matter how many buffers per send channel that are chosen there is still a possibility to run out of buffer space. If this happens the only thing to do for the RT-layer is to throw away the IP-frame. If the non RT traffic uses the TCP-protocol, the frame will be retransmitted again. For the RT traffic it only means that one periodic update is lost.

If the RT-layer uses many buffers for each send channel, there is a possibility that this will cause a lot of cache memory misses. This will add more latency, when the RT traffic is supposed to be forwarded. Considering this it maybe is best to only have one buffer per send channel.

Node traffic identification

First the source IP-address, the destination IP-address, the source port, and the destination port have to be extracted from the IP-frame. Then the values are compared to the scheduled RT traffic. If the IP-frame is scheduled it is put in the corresponding send buffer, otherwise the IP-frame is put in the send buffer for non RT traffic.

NetGuard traffic identification

The RT-layer in the NetGuard only needs to handle non RT traffic. This means that only the destination IP-address needs to be extracted. The IP-address is compared with a list of nodes, and if the IP-address is found in the list, the IP-frame is put in the corresponding send buffer.

RT-frame identification

To make a fast identification of the RT-frame content, the receive channel number should be included in the RT-frame. If the RT-frame is fragmented, the receiver must buffer all fragment until the last fragment arrives. When the frame is complete it is forwarded to the IP-layer.

6.4 Clock synchronization

Even though clock synchronization is not important for worst case scheduling, there are some reasons for implementing this function. The first reason is that if the nodes have a global time, each RTC-frame can be time stamped. By doing this the network latency can be checked when the frame is received. This information can be used for statistics and detection of network problem.

The second reason is switch related. The automatic address learning function in the switch only keeps the information for a short period. If a node in the network only receives frames, the switch will forget on which port the node is connected. So for every frame sent to the node, the switch will forward the frame on all the other ports. This phenomenon will jeopardize the real-time function. This problem is avoided, if the node is forced to send periodic message. So why not use the periodic message for clock synchronization.

6.5 IP fragmentation

Figure 6.2 shows a fragmented UDP/IP datagram. Notice that the UDP header, which includes the fields for source port and destination port, only is sent in the first frame. This could be a problem in the RT-layer when the IP-frame is supposed to be identified. Using the restriction that the IP-frame for the RT traffic never is fragmented, the RT-layer can then start the identification by looking on the fragmentation bits for the IP-frame, and if the IP-frame is fragmented, assume that it is a non RT traffic.

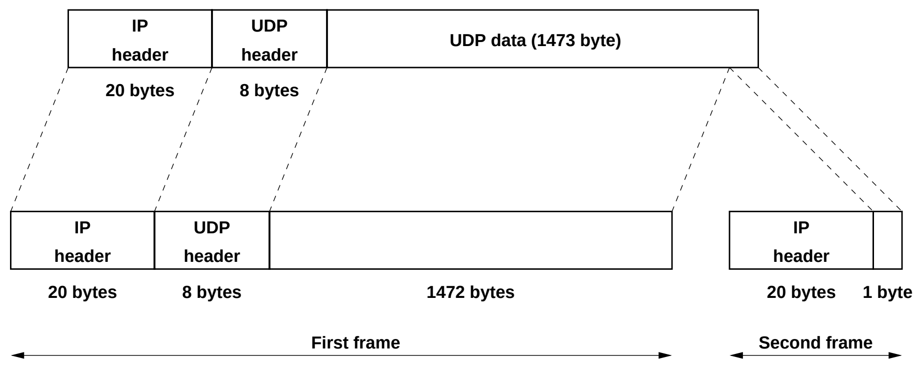

Figure 6.2 Example of UDP fragmentation

6.6 Dynamic vs Static

So far there has been no discussion about whether the scheduling should be dynamic or static. Of course a dynamic solution is more appealing.

If a dynamic implementation is considered, there are some problems to be discussed. If neither the NetGuard nor a node knows that they are in the same network, the only way is if the NetGuard or the node sends a broadcast trying to find the other. Let's assume that the node is allowed to send this broadcast. The switch forwards the broadcast on all channels, except on the channel where it received the broadcast. This means that this broadcast should not be generated too often, since it interferes with the RT traffic. There also has to be a specific broadcast channel in the RT-layer, since no send channel or receive channel has been assigned to the node yet.

6.7 Summary

As argued above the RT-frame for the test implementation should contain the following fields:

- Fragmentation information
- Receive channel number
- Time stamp

The following information is necessary for the test implementation to schedule the RT traffic:

RT traffic request

- Source IP-address. The IP-address for the sending node.
- Destination IP-address. The IP-address for the receiving node.
- Source port. The socket port used by the sending application.
- Destination port. The socket port used by the receiving application.

- Transmit time [μs]. The transmit time for the periodic RT frame, including all overhead, when the frame is sent as a single fragment. The minimum network latency, using a store-and-forward switch, is the transmit time multiplied by two.
- Frequency [Hz]. The periodic update frequency.
- Fragment frequency [Hz]. The send frequency that is used when the RT traffic is fragmented.
- Number of fragment. The number of fragments that the RT-layer should divide the periodic RT frame into, before it is transmitted.
- Maximum latency [μs]. The maximum allowed network latency.

7. Future work

The first step is to implement a real version in order to verify that the theory works in practice. One thing that could jeopardize the function is the increase of system load that the worst case scheduling traffic control fragmentation adds to the target system.

The second step is to develop fragmentation algorithms for the RT traffic. Since fragmentation is not purely an advantage, and it is not only one parameter to optimize, I would like to characterize this as a complex problem.

The third step would be to add more switches to the network. Figure 7.1 shows a example of an expanded network with four switches. The top switch can be considered as a backbone for the network. The connections between the switches are potential bottlenecks for the network. One solution which makes the situation better, is to use a Gigabit switch as backbone. This will decrease the network latency between the sub switches. Another thing to do is to add routers for the non RT traffic to each sub switch. By doing this the control of traffic in the bottlenecks will be better and more predictable.

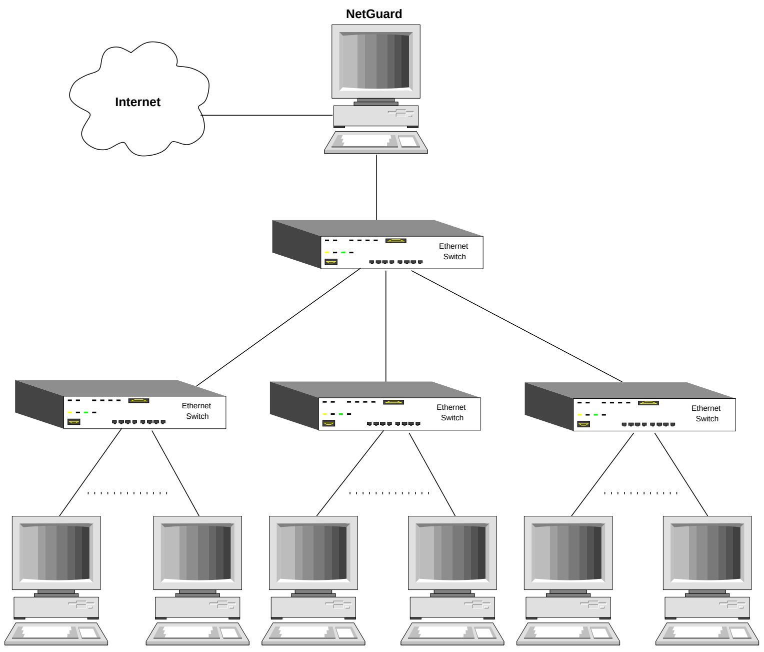

Figure 7.1 Ethernet network with four switches

Figure 7.2 shows the suggested changes to the network. The SubNet-Guard is responsible for retransmitting the non RT traffic, sent by the nodes in the sub-network, in a controlled way so that the RT traffic in the up-link for the sub-switch is not interfered. Some of the problems that remain to be investigated are:

- The impact of broadcast in the network.
- Identification of what sub switch a node is connected to.

If the network has a static configuration, it should be possible to avoid the problems above. So the last thing to find out would be a dynamic solution for the expanded network.

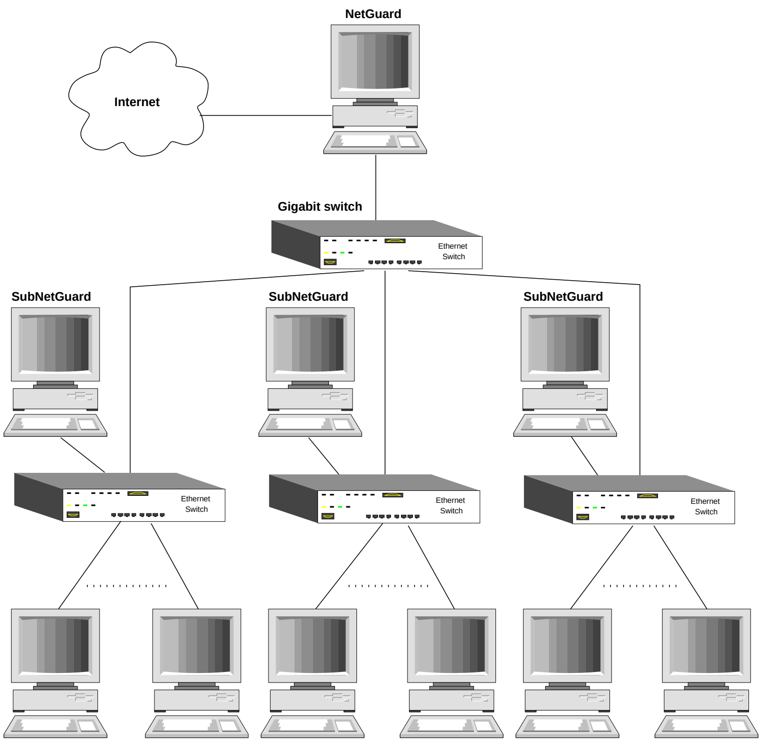

Figure 7.2 Future real-time switched Ethernet LAN

8. Conclusions

The test result in Chapter 4 shows that the tested switch performs as expected. However, it also reveals that the node can become a weak link if there is a lot of traffic with small frame sizes. If the node system load gets close to 100% the RT behavior for a scheduled switched Ethernet network can be jeopardized.

Chapter 5 investigates how the traffic in the network can be controlled by using worst case scheduling. The scheduler takes the buffers in the switch and in the network interface card into account to guarantee that the maximum allowed network latency is not exceeded. The result of the scheduling is a number of simple equations that calculate the time for when it is allowed to send another frame through the switch. The problem with the node system load can be avoided if a lower utilization of the bandwidth is acceptable.

Finally, Chapter 6 shows that the worst case schedule traffic control can be implemented as an extra layer in the TCP/IP protocol suite.

9. References

[1] William Stalling. Data & Computer Communication Sixth Edition, Prentice-Hall Inc, 2000.

[2] W. Richard Stevens. TCP/IP Illustrated Volume 1, Addison Wesley, 1994.

[3] RFC2889, Benchmarking Methodology for LAN Switching Devices.