# Efficient many-to-many real-time communication using an intelligent Ethernet switch

Xing Fan, Magnus Jonsson and Hoai Hoang



Fan X, Jonsson M, Hoang H. Efficient many-to-many real-time communication using an intelligent Ethernet switch. In: Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004.. IEEE Computer Society; 2004. p. 280-287. International Symposium on Parallel Architectures, Algorithms and Networks, 7.

DOI: http://dx.doi.org/10.1109/ISPAN.2004.1300493
Copyright: IEEE

Post-Print available at: Halmstad University DiVA
http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-387

# Efficient Many-to-Many Real-Time Communication Using an Intelligent Ethernet Switch

Xing Fan, Magnus Jonsson, and Hoai Hoang

*School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden, Box 823, S-301 18, Sweden. {Xing.Fan, Magnus.Jonsson, Hoai.Hoang}@ide.hh.se*

## Abstract

*This paper presents a solution for efficient many-to-many communication over switched Ethernet. The performance of using an ordinary switch and an intelligent switch is compared, and the results of the analysis based on 100 Mbit/s Fast Ethernet show that an intelligent switch used to handle many-to-many communication can give better performance (shorter latency and higher utilization) than an ordinary switch. We also extend the network to serve many-to-many traffic with real-time demands by adding a thin software layer to the intelligent Ethernet switch and the end-nodes. Earliest Deadline First (EDF) scheduling is used.*

## 1. Introduction

Modern and future parallel processing applications, such as radar signal processing applications, often involve many-to-many communication [1] [2]. Much work has been done on many-to-many communication [3-8], but has thus far primarily been on the software level with algorithm solutions for scheduling, routing, scalability, security etc. Work on the network level was proposed in [1] [9] [10], but is developed for other types of networks and differ significantly from our strategies of the many-to-many communication for switched Ethernet networks. A reliable many-to-many multicast protocol for group communication was described in [10], but aimed for wide-area ATM networks.

Since switched Ethernet uses a cost-effective off-the-shelf technology, it is now so prevalent and frequently used that it will probably take over much of the industrial bus and network markets in the future. With increased supported bit rates, switched Ethernet is also a good candidate for parallel processing systems, including clusters of workstations.

Some research has been done based on switched Ethernet with a similar idea of enhancing the switch with some intelligence, e.g. supporting guarantees for real-time traffic without modifications to the Ethernet hardware [11] [12]. However, the main aim of the work we present here is to add intelligence to the switch to give efficient support for many-to-many communication over switched Ethernet networks, without requiring significant modifications of the switched Ethernet specifications. Thinking the benefits of combining switches [13] [14], a heave researched area in parallel computers, we use the technique of message combining to improve the Ethernet switch. The particularly attractive feature proposed in this work is using intelligent switches because of the higher utilization and shorter latency, as compared to ordinary switches.

Only a thin layer is added between the Ethernet protocols and the TCP/IP suite in the end stations. The switch and the end-nodes control the many-to-many real-time traffic with EDF (Earliest Deadline First) scheduling on the frame level. The switch is responsible for admission control, where a feasibility analysis is made for each link and direction between end-nodes and the switch.

The paper is organized as follows. Section 2 discusses the network architecture and the properties of the ordinary switch relative to those of the intelligent switch while handling many-to-many communication. A comparative analysis is given in Section 3. Section 4 describes how to offer real-time support to many-to-many traffic, and the paper is concluded in Section 5.

## 2. Network description

Switched Ethernet offers some key benefits over traditional Ethernet, such as full duplex, flow control etc, because the switched technology makes it possible to describe a switch port to a single end-node. It also directs network traffic in an efficient manner, establishing a sort of direct line of communication between two ports for each frame, and maintains multiple simultaneous links between various ports. Taking advantage of this progress, we now focus on switched Ethernet to find efficient support for many-to-many communication.

Many-to-many communication can be generally described as more than one source node sending out data at the same time, and every source node sending data to more than one destination node. Here, personalized communication is considered, which means that the source node sends out different messages to different destination nodes. In an ordinary switched Ethernet without special support for many-to-many communication, a many-to-many stream is handled such that, once the switch receives the messages from the source node, it sends to the

Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)
1087-4089/04 $ 20.00 © 2004 IEEE

destination nodes immediately. However, separate small messages are sent between each source/destination pair.

The intelligent Ethernet switch proposed here provides control to enhance the many-to-many operation of the network. The many-to-many communication is divided into two phases: the first phase when all the sending nodes send data to the switch and the second phase when the switch sends the reorganized data to the destination nodes. The intelligent switch can store the data in the incoming messages into a data matrix during phase 1, and reorganizes it before sending it in the outgoing messages to the destinations during phase 2. The reorganizing process done by the intelligent switch is a so-called corner-turn, which can be explained as a transposition of the data matrix when the data matrix is in two dimensions. With the intelligent switch, several messages from the same source node in phase 1 or with the same destination node in phase 2 can be included in the same Ethernet frame if the length limit of the Ethernet standard allows this. For the same amount of data, less overhead (header information) is added using the intelligent switch as compared to using the ordinary switch. The intelligent switch can thus offer higher performance, which is verified by the analysis in Section 3.

## 3. Analysis

This section gives a utilization and latency analysis of many-to-many communication in the ordinary and intelligent switch, where Subsection 3.1 describes the assumptions and parameters in the analysis, 3.2 makes a utilization calculation and 3.3 a latency calculation. The results and analysis are in 3.4.

### 3.1 Utilization calculation

Our analysis is made for a full-duplex switched fast Ethernet (100 Mbits/s). For simplicity, we assume that all of the end-nodes are clock synchronized and that every node starts sending data at the same time and send to all other nodes except itself. The detailed definitions of all the parameters used for the utilization and latency analysis are explained in Table 1. The data matrix is used to describe how the intelligent switch stores and processes the many-to-many data, which has fixed-length data elements. The rows of the data matrix show the data to the switch, while the columns of the data matrix show the data from the switch. Two many-to-many communication data matrixes and some related parameters in two different cases, when each end-node is responsible for the same amount of traffic ($M/N$ is an integer) and when each end-node is not responsible for the same amount of traffic ($M/N$ is not an integer), are shown in Figures 1a and 1b, respectively.

### 3.2 Utilization calculation

The utilization is highly dependent on the relative amount of header information (we include, e.g, frame check sequence and inter-frame gap in the term header). In order to calculate the utilization, we consider the IEEE 802.3 MAC header, inter-frame gap (12 bytes), LLC header (4 bytes), IP header (20 bytes) and UDP header (8 bytes). The length of the MAC header changes relative to the length of the UDP data. When the UDP data are not less than 14 bytes, the MAC header is 70 bytes (Figure 2a). Otherwise a pad field is added to the header to meet the minimum frame size required by IEEE 802.3; thus the whole frame header should be 84 bytes minus the length of the UDP data (Figure 2b). The maximum length of the UDP data field in an Ethernet frame is 1468 bytes, following the IEEE 802.3 standard, which permits a maximum frame length of 1538 bytes.

The utilization analysis equations of many-to-many communication for the ordinary switch are described by Equations 1-15, resulting in $U_O$.

$$P_{BO1} = \left((MdivN + 1)^2 E\right) div 1468 \quad (1)$$

$$L_{BO1} = \left((MdivN + 1)^2 E\right) \bmod 1468 \quad (2)$$

$$U_{BO1} = \begin{cases} 1468/1538, & L_{BO1} = 0 \\ \left(\frac{1468}{1538} P_{BO1} + \frac{L_{BO1}}{84}\right)/(P_{BO1}+1), & 0 < L_{BO1} < 14 \\ \left(\frac{1468}{1538} P_{BO1} + \frac{L_{BO1}}{70 + L_{BO1}}\right)/(P_{BO1}+1), & 14 \le L_{BO1} \le 1468 \end{cases} \quad (3)$$

$$P_{BO2} = \left((MdivN)(MdivN + 1)E\right) div 1468 \quad (4)$$

$$L_{BO2} = \left((MdivN)(MdivN + 1)E\right) \bmod 1468 \quad (5)$$

$$U_{BO2} = \begin{cases} 1468/1538, & L_{BO2} = 0 \\ \left(\frac{1468}{1538} P_{BO2} + \frac{L_{BO2}}{84}\right)/(P_{BO2}+1), & 0 < L_{BO2} < 14 \\ \left(\frac{1468}{1538} P_{BO2} + \frac{L_{BO2}}{70 + L_{BO2}}\right)/(P_{BO2}+1), & 14 \le L_{BO2} \le 1468 \end{cases} \quad (6)$$

$$U_B = \left(U_{BO1}(N_B - 1) + U_{BO2} N_A\right)/(N - 1) \quad (7)$$

$$P_{AO1} = \left((MdivN)(MdivN + 1)E\right) div 1468 \quad (8)$$

$$L_{AO1} = \left((MdivN)(MdivN + 1)E\right) \bmod 1468 \quad (9)$$

| | |
|---|---|
| $N$: | Number of participating end-nodes in a many-to-many communication session |
| $M$: | Number of data elements per row/column of the data matrix |
| $E$: | Length (in bytes) of each data element |
| $A$: | A set including the nodes that are each responsible for $MdivN$ of rows/columns of elements in the data matrix |
| $B$: | A set including the nodes each responsible for more than $MdivN$ of rows/columns of elements in the data matrix |
| $N_B$: | Number of end-nodes in set $B$, $N_B = M \bmod N$ |
| $N_A$: | Number of end-nodes in set $A$, $N_A = N - N_B$ |
| $P_{BO1}$: | Number of frames (1468 bytes data) transmitted from each node in set $B$ to another node in set $B$ |
| $L_{BO1}$: | Length (in bytes) of data in the last frame transmitted from each node in set $B$ to another node in set $B$ |
| $P_{BO2}$: | Number of frames (1468 bytes data) transmitted from each node in set $B$ to another node in set $A$ |
| $L_{BO2}$: | Length (in bytes) of data in the last frame transmitted from each node in set $B$ to another node in set $A$ |
| $U_{BO1}$: | Utilization of the many-to-many communication from each node in set $B$ to another node in set $B$ |
| $U_{BO2}$: | Utilization of the many-to-many communication from each node in set $B$ to another node in set $A$ |
| $U_B$: | Average utilization of the many-to-many communication from each node in set $B$ to all other nodes |
| $P_{AO1}$: | Number of frames (1468 bytes data) transmitted from each node in set $A$ to another node in set $B$ |
| $L_{AO1}$: | Length (in bytes) of data in the last frame transmitted from each node in set $A$ to another node in set $B$ |
| $P_{AO2}$: | Number of frames (1468 bytes data) transmitted from each node in set $A$ to another node in set A |
| $L_{AO2}$: | Length (in bytes) of data in the last frame transmitted from each node in set $A$ to another node in set A |
| $U_{AO1}$: | Utilization of the many-to-many communication from each node in set A to another node in set $B$ |
| $U_{AO2}$: | Utilization of the many-to-many communication from end node in set $A$ to another node in set $A$ |
| $U_A$: | Average utilization of the many-to-many communication from each node in set $A$ to all other nodes |
| $U_O$: | Utilization of the whole many-to-many communication over the ordinary switched Ethernet |
| $P_{BI}$: | Number of frames (1468 bytes data) transmitted from each node in set $B$ to all other nodes |
| $L_{BI}$: | Length (in bytes) of data in the last frame transmitted between from each node in set $B$ to all other nodes |
| $P_{AI}$: | Number of frames (1468 bytes data) transmitted from each node in set $A$ to all other nodes |
| $L_{AI}$: | Length (in bytes) of data in the last frame transmitted from each node in set $B$ to all other nodes |
| $U_I$: | Utilization of the whole many-to-many communication over the intelligent switched Ethernet |
| $C$: | A set including all of the nodes if $M/N$ is an integer, otherwise including the nodes that are each responsible for more than $N$ of rows/columns of elements in the data matrix |
| $N_C$: | Number of end-nodes in set $C$, $N_C = \begin{cases} N & (M \bmod N = 0) \\ M \bmod N & (M \bmod N \neq 0) \end{cases}$ |
| $P_{O1}$: | Number of frames (1468 bytes data) transmitted from each node in set $C$ to another node in set $C$ |
| $L_{O1}$: | Length (in bytes) of data in the last frame transmitted from each node in set $C$ to another node in set C |
| $P_{O2}$: | Number of frames (1468 bytes data) transmitted from each node in set $C$ to another node not in set $C$ |
| $L_{O2}$: | Length (in bytes) of data in the last frame transmitted from each node in set $C$ to another node not in set C |
| $Q_1$: | Length (in bytes) of data and header from each node in set $C$ to all other nodes in set $C$ |
| $Q_2$: | Length (in bytes) of data and header from each node in set $C$ to all nodes not in set $C$ |
| $P_I$: | Number of frames (1468 bytes data) transmitted from each node in set $C$ to all other end nodes |
| $L_I$: | Length (in bytes) of data in the last frame transmitted from each node in set $C$ to all other nodes |
| $Q$: | Length (in bytes) of data and header from each node in set $C$ to all other nodes |
| $T_{link\_prop\_delay}$: | Maximum propagation delay over a link between an end-node and the switch |
| $T_{ct-delay}$: | Corner-turn delay inside the intelligent switch |
| $T_1$: | The whole many-to-many traffic transmission delay from the source nodes to the switch |
| $T_2$: | The whole many-to-many traffic transmission delay from the switch to the destination nodes |
| $T_O$: | Latency experienced by the whole many-to-many traffic on an ordinary switched Ethernet |
| $T_I$: | Latency experienced by the whole many-to-many traffic on an intelligent switched Ethernet |

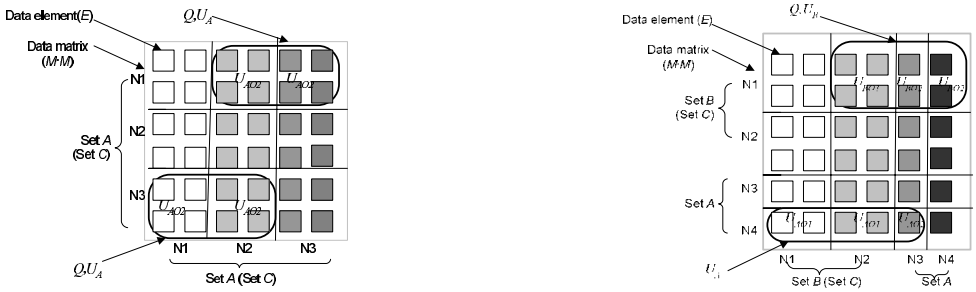
**Table 1. Notations used in describing the analysis equations.**

$$U_{AO1} = \begin{cases} 1468/1538, & L_{AO1} = 0 \\ \left(\frac{1468}{1538} P_{AO1} + \frac{L_{AO1}}{84}\right) / (P_{AO1} + 1), & 0 < L_{AO1} < 14 \\ \left(\frac{1468}{1538} P_{AO1} + \frac{L_{AO1}}{70 + L_{AO1}}\right) / (P_{AO1} + 1), & 14 \le L_{AO1} \le 1468 \end{cases} \quad (10)$$

$$P_{AO2} = \left((MdivN)^2 E\right) div 1468 \quad (11)$$

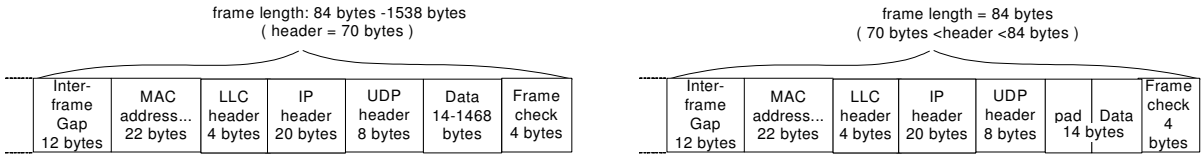
$$L_{AO2} = \left((MdivN)^2 E\right) \bmod 1468 \quad (12)$$

$$U_{AO2} = \begin{cases} 1468/1538, & L_{AO2} = 0 \\ \left(\frac{1468}{1538} P_{AO2} + \frac{L_{AO2}}{84}\right) / (P_{AO2} + 1), & 0 < L_{AO2} < 14 \\ \left(\frac{1468}{1538} P_{AO2} + \frac{L_{AO2}}{70 + L_{AO2}}\right) / (P_{AO2} + 1), & 14 \le L_{AO2} \le 1468 \end{cases} \quad (13)$$




**(a)** **(b)**

**Figure 1. Data matrix of the many-to-many communication. (a) *M*=6, *N*=3, (b) *M*=6, *N*=4.**




**(a)** **(b)**

**Figure 2. Data frame. (a) without pad, (b) with pad.**

$$U_A = \left(U_{AO1} N_B + U_{AO2}(N_A - 1)\right)/(N-1) \quad (14)$$

$$U_O = \left(U_A N_A + U_B N_B\right)/N \quad (15)$$

The utilization analysis equations of many-to-many communication for the intelligent switch are described by Equations 16-22, resulting in $U_I$.

$$P_{BI} = \left(\left(M(MdivN+1) - (MdivN+1)^2\right)E\right) div 1468 \quad (16)$$

$$L_{BI} = \left(\left(M(MdivN+1) - (MdivN+1)^2\right)E\right) \bmod 1468 \quad (17)$$

$$U_B = \begin{cases} 1468/1538, & L_{BI} = 0 \\ \left(\frac{1468}{1538}P_{BI} + \frac{L_{BI}}{84}\right)/(P_{BI}+1), & 0 < L_{BI} < 14 \\ \left(\frac{1468}{1538}P_{BI} + \frac{L_{BI}}{70+L_{BI}}\right)/(P_{BI}+1), & 14 \le L_{BI} \le 1468 \end{cases} \quad (18)$$

$$P_{AI} = \begin{cases} ((MdivN)(M - MdivN)E) div 1468, & M \bmod N \neq 0 \\ (M^2(N-1)E/N^2) div 1468, & M \bmod N = 0 \end{cases} \quad (19)$$

$$L_{AI} = \begin{cases} ((MdivN)(M - MdivN)E) \bmod 1468, & M \bmod N \neq 0 \\ (M^2(N-1)E/N^2) \bmod 1468, & M \bmod N = 0 \end{cases} \quad (20)$$

$$U_A = \begin{cases} 1468/1538, & L_{AI} = 0 \\ \left(\frac{1468}{1538}P_{AI} + \frac{L_{AI}}{84}\right)/(P_{AI}+1), & 0 < L_{AI} < 14 \\ \left(\frac{1468}{1538}P_{AI} + \frac{L_{AI}}{70+L_{AI}}\right)/(P_{AI}+1), & 14 \le L_{AI} \le 1468 \end{cases} \quad (21)$$

$$U_I = \left(U_A N_A + U_B N_B\right)/N \quad (22)$$

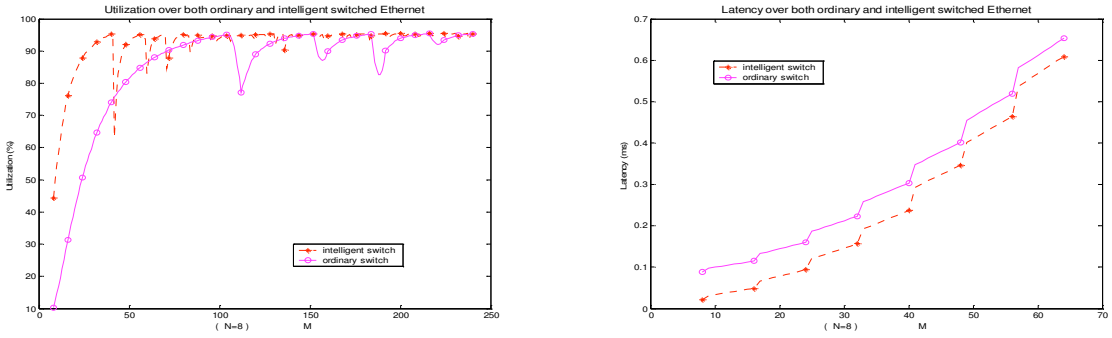
### 3.3 Latency calculation

The latency is defined as the time from the generation of the data in the source nodes until the data is received at all destination nodes. In the analysis, we assume that the propagation delay over a link between an end-node and the switch, $T_{prop\_del}$, is 500 ns. Moreover, since the intelligent switch processes the packets instead of blindly routing them, we account for the corner-turn delay in the intelligent switch. In our analysis the corner-turn delay, $T_{ct\_del}$, is assumed to be 1 s, since the corner-turn is also done during reception in parallel with the other latency and the 1 s here is only the delay of arranging the last received data and initiating transmission. A sufficiently powerful processor is assumed to reside in the intelligent switch. Earlier results of low-complexity data rearrangements in a switch (deadline sorting) showed by experiments that a microprocessor can handle this [12]. In the case of many-to-many communication, the task is only to copy the incoming data into memory in a rearranged, but regular, order.

The latency analysis equations of many-to-many communication for the ordinary switch are described by Equations 23-31, resulting in $T_O$.

$$P_{O1} = \begin{cases} \left(M^2/N^2 E\right) div 1468, & M \bmod N = 0 \\ \left((MdivN+1)^2 E\right) div 1468, & M \bmod N \neq 0 \end{cases} \quad (23)$$

$$L_{O1} = \begin{cases} \left(M^2/N^2 E\right) \bmod 1468, & M \bmod N = 0 \\ \left((MdivN+1)^2 E\right) \bmod 1468, & M \bmod N \neq 0 \end{cases} \quad (24)$$




**(a)** **(b)**

**Figure 3. Utilization (a) and latency (b) for a network with 8 nodes.**

$$Q_1 = \begin{cases} 1538P_{O1}(N_C - 1), & L_{O1} = 0 \\ (1538P_{O1} + 84)(N_C - 1), & 0 < L_{O1} < 14 \\ (1538P_{O1} + 70 + L_{O1})(N_C - 1), & 14 \le L_{O1} \le 1468 \end{cases} \quad (25)$$

$$P_{O2} = (((MdivN)(MdivN + 1)E)div1468 \quad (26)$$

$$L_{O2} = (((MdivN)(MdivN + 1)E) \bmod 1468 \quad (27)$$

$$Q_2 = \begin{cases} 1538P_{O1}(N - N_C), & L_{O2} = 0 \\ (1538P_{O1} + 84)(N - N_C), & 0 < L_{O2} < 14 \\ (1538P_{O1} + 70 + L_{O1})(N - N_C), & 14 \le L_{O2} \le 1468 \end{cases} \quad (28)$$

$$Q = Q_1 + Q_2 \quad (29)$$

$$T_1 = T_2 = 8Q/100Mbit/s \quad (30)$$

$$T_O = 2T_{prop\_del} + T_1 + T_2 \quad (31)$$

The latency analysis equations of many-to-many communication for the intelligent switch are described by Equations 32-36, resulting in $T_I$.

$$P_I = \begin{cases} (M^2(N-1)E/N^2)div1468, & M \bmod N = 0 \\ (((MdivN + 1)(M - MdivN - 1))E)div1468, & M \bmod N \neq 0 \end{cases} \quad (32)$$

$$L_I = \begin{cases} (M^2(N-1)E/N^2) \bmod 1468, & M \bmod N = 0 \\ (((MdivN + 1)(M - MdivN - 1))E) \bmod 1468, & M \bmod N \neq 0 \end{cases} \quad (33)$$

$$Q = \begin{cases} 1538P_I, & L_I = 0 \\ 1538P_I + 84, & 0 < L_I < 14 \\ 1538P_I + 70 + L_I, & 14 \le L_I \le 1468 \end{cases} \quad (34)$$

$$T_1 = T_2 = 8Q/100Mbit/s \quad (35)$$

$$T_I = 2T_{prop\_del} + T_1 + T_2 + T_{ct\_del} \quad (36)$$

### 3.4 Results and analysis

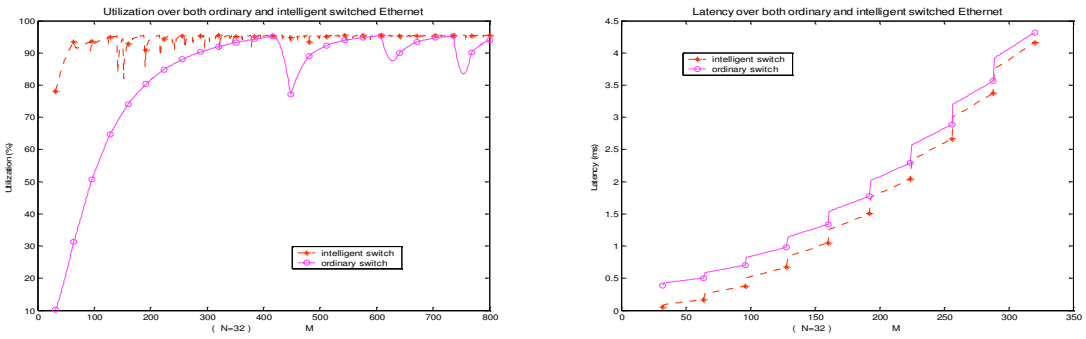
Figures 3-5 show the analysis results comparing the utilization and latency of many-to-many communication of an ordinary switch and an intelligent switch. The length of each data element, $E$, is 8 bytes. The following points can be observed from these results:

(1) The intelligent switch gives better performance (higher utilization and lower latency) than the ordinary switch in nearly every case, because the corner-turn process done by the intelligent switch adds less header to the data than an ordinary switch, as explained in Section 2. The performance improvement is obvious, e.g., when $N$ is 32 and $M$ is 64, the utilization in the case of the intelligent switch is three times the utilization with the ordinary switch, while the latency is only 50 %.

(2) There is a trend of increasing utilization for both the ordinary and the intelligent switches, as the traffic increases, arriving at peak values higher than 90 %. Sudden decreases occasionally happen on the curves, which are caused by the short frames (Figure 2b) having a pad field whose utilizations are lower than 17 %.

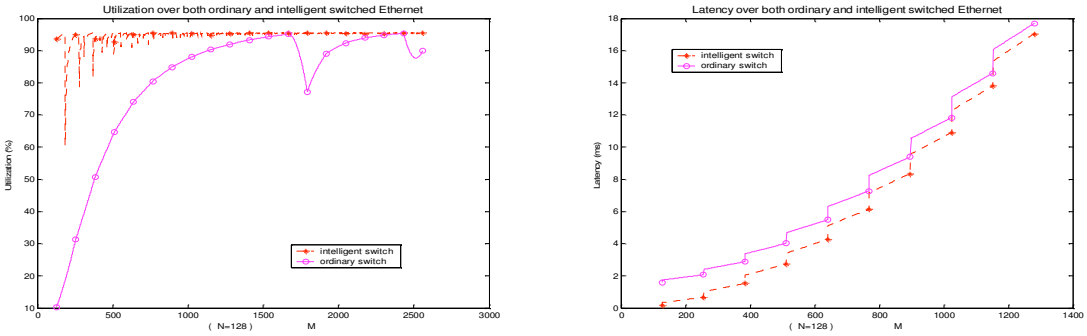
(3) There is a trend of increasing latency for both the ordinary and the intelligent switches, as the traffic increases. The sudden increases on the curves take place when the increase in $M$ makes a node $i$ responsible for one more row (column after the corner-turn) in the matrix than the other nodes. In this case, the many-to-many communication does not finish until the traffic related to node $i$ finishes. A relatively stable period of increasing takes place after a sudden increase, starting at the point when only one such node $i$ exists and ending at the point when each node is responsible for the same amount of data ($M/N$ is an integer), which is marked on the curves and shown in Figure 1a. A next sudden increase then happens, because the change in $M$ makes such a node $i$ appear again. The different amount of the increased traffic makes the different increasing degree (sudden or stable) in the analysis figures.

(4) In contrast to the stair-like latency curves, the utilization curves are smoother, because we assume that the unused bandwidth left by many-to-many traffic when waiting for the last node(s) to finish can be used by other kinds of traffic, e.g., non-real-time traffic. When $M/N$ is not an integer, there exists at least one such node $i$ that is responsible for more traffic than other nodes.




**(a)** **(b)**

**Figure 4. Utilization (a) and latency (b) for a network with 32 nodes.**




**(a)** **(b)**

**Figure 5. Utilization (a) and latency (b) for a network with 128 nodes.**

## 4. Real-time support

Based on the same network architecture as described above, we first discuss how to support guarantees for periodic real-time traffic in the network in general, and then how to extend this to many-to-many traffic with real-time demands.

### 4.1 Protocols and architecture

We add software (RT layer) to both the switch and the end-nodes to support guarantees for periodic real-time traffic. All nodes are connected to the switch and nodes can communicate with each other over logical real-time channels (RT channels), each being a virtual connection between two nodes in the system.

The function of, and interaction with, the RT layer etc. shown in Figure 6 is explained below, while the many-to-many layer, as described later, is assumed to be unused now. When an application wants to set up an RT channel, it interacts directly with the RT layer (1). The RT layer then sends a question to the RT channel management software in the switch (2). Outgoing real-time traffic from the end-node uses UDP and is put in a deadline-sorted queue in the RT layer (3). Outgoing non-real-time traffic from the end-node typically uses TCP and is put in an FCFS-sorted (First Come First Serve) queue in the RT layer (4). In the same way, there are also two different output queues for each port on the switch (5).

### 4.2 Single destination real-time communication

The real-time guarantee is upheld by an RT channel with index $i$ which is characterized by:

$$\{T_{period,i}, C_i, T_{deadline,i}\} \quad (37)$$

where $T_{period,i}$ is the period of data generation, $C_i$ is the amount of data per period and $T_{deadline,i}$ is the relative deadline used for the end-to-end EDF (Earliest Deadline First) scheduling.

$$T_{deadline,i} = T_{D1,i} + T_{D2,i} \quad (38)$$

is assumed, where $T_{D1,i}$ and $T_{D2,i}$ are the deadlines for real-time frames from the source node to the switch and from the switch to the destination node, respectively. $T_{deadline,i}$ can be partitioned in a number of ways. For simplicity, we here assume that

$$T_{D1,i} = T_{D2,i} = T_{period,i} / 2 \quad (39)$$

The scheduling of RT frames in the switch (and of outgoing RT frames in the end-nodes) is made according to EDF, i.e. all incoming RT traffic is served in deadline order to guarantee the worst-case delay. A detailed description of the RT channel establishment and RT traffic handling is given in [11].

### 4.3 Many-to-many real-time communication

Below we explain how to implement real-time support for many-to-many traffic. As shown in Figure 6, the many-to-many layer is based on UDP, so the many-to-many frames can be put in the RT traffic queue. After the switch has reorganized these many-to-many real-time frames coming from the source nodes, it sends out the reorganized frames to the destination nodes.

Considering the many-to-many characteristic, we divide the end-to-end many-to-many logical RT channel with index *i* into two channels, the RT channel from the source to the switch (actually one channel from each source node) and the RT channel from the switch to the destination (actually one channel to each destination node), which are characterized respectively by:

$$\{T_{perio1,i},\ C_{1i,j},\ T_{D1,i}\} \tag{40}$$

$$\{T_{period,i},\ C_{2i,j},\ T_{D2,i}\} \tag{41}$$

where *j* is the index of an end-node, and $C_{1i,j}$ (*j* denotes the source node) and $C_{2i,j}$ (*j* denotes the destination node) are the maximum amount of data per period from the source node to the switch and from the switch to the destination node, respectively. $C_{1i,j}$ and $C_{2i,j}$ can have different values, for instance, if there are fewer source nodes than destination nodes.

### 4.4 Feasibility analysis

As mentioned above, we use EDF [15] as the schedule algorithm for both the switch and the end-nodes. The feasibility test of accepting a new connection (admission control) is done in two steps, utilization constraint and workload constraint, each step being a test of its own.

According to basic EDF theory the worst-case maximum utilization for a physical link between a source node and the switch, $U_{\max 1}$, and for a physical link between the switch and a destination node, $U_{\max 2}$, are both 100%. To meet the utilization constraint, the utilization of RT traffic from/to an end-node *k* must not exceed these levels:

$$U = \sum_{i,j=k} \left(C_{1i,j} / T_{period,i}\right) \le U_{\max 1}$$
$$U = \sum_{i,j=k} \left(C_{2i,j} / T_{period,i}\right) \le U_{\max 2} \tag{42}$$

In order to describe the second step of the feasibility test, we establish the following definitions, by viewing the channel as a periodic task.

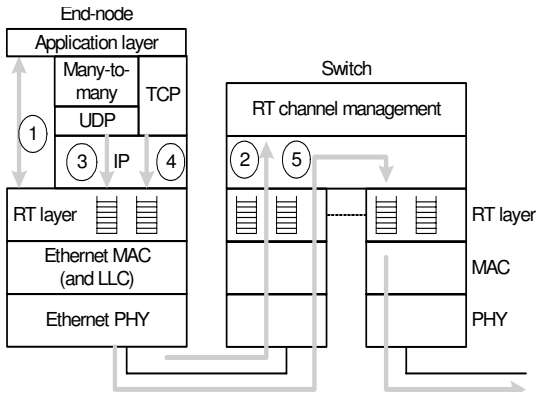



**Figure 6. Layers and output queues.**

The ***Hyperperiod*** for a set of periodic tasks is defined as the length of time from that when all tasks' periods start at the same time until they start at the same time again. A ***BusyPeriod*** is any interval of time in which a link is not idle. The ***workload function h(t)*** is the sum of all the capacities of the tasks with an absolute deadline less than or equal to *t*, where *t* is the number of time slots (maximum sized frames) that have elapsed from the start of the hyperperiod. Many different many-to-many communication operations, or other channels, with real-time demands might exist in the network at the same time. *h(t)* is calculated as follows for each physical link from an end-node to the switch and for each physical link from the switch to an end-node, respectively, for (from/ to) an end node:

$$h(t) = \sum_{i,j=k} (1 + \lfloor (t - T_{D1,i}) / T_{period,i} \rfloor) C_{1i,j},$$
$$h(t) = \sum_{i,j=k} (1 + \lfloor (t - T_{D2,i}) / T_{period,i} \rfloor) C_{2i,j} \tag{43}$$

According to the second constraint, the workload function *h(t)* must be less than or equal to *t* for all values of *t*. To check only the following range of *t* is an improvement of the algorithm above [16]:

$$1 \le t \le BusyPeriod \tag{44}$$

where *BusyPeriod* is the first BusyPeriod in the schedule at the start of the hyperperiod. Furthermore, one need not check every integer from the first time slot, but only the integers *t*:

$$t \in \bigcup_{i=1}^{n} \{mT_i + d_i : m = 0,1,\ldots\} \tag{45}$$

where each *i*, $1 \le i \le n$ is the index of an RT channel traversing the considered physical link in the considered direction.

When an RT channel has been established, the network guarantees delivery of each generated message with a bounded delay, $T_{max\_delay,\ i} = T_{deadline,i} + T_{lat}$. The worst-case situation is when all RT-channels start at the same time, and the maximum allowed capacity for each RT channel is

used. The worst-case latency is, for all these many-to-many logical RT channels, characterized by:

$$T_{lat} = 2T_{prop\_del} + T_{node\_acc} + T_{ct\_del} + T_{switch\_acc} \quad (46)$$

where $T_{prop\_del}$ and $T_{ct\_del}$ are explained in Section 3. $T_{node\_acc}$ and $T_{switch\_acc}$ are the worst-case latencies for a frame with the earliest deadline for leaving the source node and for leaving the switch, respectively, since we assume that we cannot interrupt the transmission of frames that have been stored on the network interface card, even though they might have later deadlines than other frames.

## 5. Conclusions

We have investigated the performance of switched Ethernet when transporting many-to-many streams. The study covers both ordinary and intelligent switched Ethernet with different parameters and shows that the intelligent switch achieves higher utilization and shorter latency than the ordinary switch. This efficient many-to-many communication is good for many parallel processing applications with high demands on many-to-many communication performance, e.g. for use in a radar signal processing systems. The so called corner-turn (involving many-to-many communication) in radar signal processing applications is often the most demanding communication action in this kind of systems [17]. Furthermore, we have proposed a solution to enhance the intelligent switch to give real-time support for many-to-many traffic, by adding software between the network layer and the link layer to dynamically set up real-time channels.

## References


[1] M. Jonsson, "Fiber-optic interconnection networks for signal processing applications," *4th International Workshop on Embedded HPC Systems and Applications (EHPC'99) held in conjunction with the 13th International Parallel Processing Symposium & 10th Symposium on Parallel and Distributed Processing (IPPS/SPDP '99),* San Juan, Puerto Rico, Apr. 16, 1999. Published in Lecture Notes in Computer Science. vol. 1586, Springer Verlag, pp. 1374-1385, 1999.

[2] K. Konstantinides, "VLIW architectures for media processing," *IEEE Signal Processing Magazine,* vol. 15, no. 2*,* pp. 16-19, Mar. 1998.

[3] S.Ranka, R. V. Shankar, and K. A. Alsabti, "Many-to-many personalized communication with bounded traffic," *Fifth Symposium on the Frontiers of Massively Parallel Computation (Frontiers'95)*, Feb. 06 - 09, 1995, McLean, VA, USA.

[4] W. Yoon, D. Lee, and H. Y Youn, "On the scalability of many-to-many reliable multicast," *Proceedings of the 16th IEEE IPDPS 2002 Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS '02),* Fort Lauderdale, FL, USA, Apr. 2002.

[5] M. Baldi, Y. Ofek, and B. Yener, "Adaptive real time group multicast," *16th Joint Conference of the IEEE Computer and Communication Societies (INFOCOM '97),* Kobe, Japan, Apr. 1997.

[6] Y. Sun, P. Y. S. Cheung, X. Lin, and K. Li: "Fault tolerant all-to-all broadcast in general interconnection networks," *ICPADS 1998*, pp. 240-247.

[7] Y. Yang and J. Wang, "Pipelined all-to-all broadcast in all-port meshes and tori," *IEEE Transactions on Computers*, vol. 50, no. 10, pp. 1020-1032, Oct. 2001.

[8] A. Borodin, Y. Rabani, and B. Schieber, "Deterministic many-to-many hot potato routing", *IEEE Trans. Parallel Distributed systems*., vol. 8, no.6, pp. 587—596, June 1997.

[9] C. Bergenhem and M. Jonsson, "Fibre-ribbon ring network with inherent support for earliest deadline first message scheduling," *Proc. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2002)*, Fort Lauderdale, FL, USA, Apr. 15-16, 2002.

[10] R. R. Koch, L. E. Moser, P. M. Melliar-Smith, "A reliable many-to-many multicast protocol for group communication over ATM networks," *Proc. International Conference on Dependable Systems and Networks (DSN'2000)*, New York, NY, USA, June 25-28, 2000.

[11] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, "Switched real-time Ethernet with earliest deadline first scheduling − protocols and traffic handling," *Proc. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS'2002)*, Fort Lauderdale, FL, USA, Apr. 15-16, 2002.

[12] H. Hoang, M. Jonsson, A. Larsson, R. Olsson, and C. Bergenhem, "Deadline first scheduling in switched real-time Ethernet-deadline partitioning issues and software implementation experiments". *Proc. Workshop on Real-time LANs in the Internet Age (RTLIA 2002),* June 2002, Vienna, Austria.

[13] S. R. Dickey and O. E. Percus, "Performance differences among combining switch architectures", *Proc. International Conference on Parallel Processing (ICPP'1992),* An Arbor, MI, USA, Aug. 17-21, 1992.

[14] G. H. Lee, C. P. Kruskal, and D. J. Kuck, "The effectiveness of combining in shared memory parallel computers in the presence of 'hot spots'", *Proc. International Conference on Parallel Processing (ICPP'1986),* University park, PA, USA, Aug. 17-21, 1986, pp. 35-41.

[15] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real-time traffic environment", *Journal of the Association for Computing Machinery*, vol. 20, no. 1, Jan. 1973

[16] J. A.Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo, "Deadline Scheduling for Real-Time Systems - EDF and Related Algorithms", *Kluwer Academic Publishers,* 1998.

[17] K. Teitelbaum, "Crossbar tree networks for embedded signal processing applications," *Proc. Massively Parallel processing using Optical Interconnections (MPPOI'98)*, Las Vegas, NV, USA, June 15-17, 1998, pp. 200-207.