Parallel Cluster Computing with IEEE1394-1995

László Böszörményi, Günther Hölzl and Emanuel Pirker

Institut für Informationstechnologie, Universität Klagenfurt
Universitätsstraße 65–67, A–9020 Klagenfurt
{laszlo@itec,guenther@itec,epirker@edu}.uni-klu.ac.at

Abstract. *Diverging demands on computer networks, such as high bandwidth, guaranteed quality of service and low latency lead to growing heterogeneity. IEEE1394-1995 is a standardized low-cost high-performance serial-bus-system with both isochronous and asynchronous operation. It might be an interesting candidate for all-round local and system area networks, providing a good compromise in fulfilling the above demands for low costs. Beside providing some technical background we show the possibilities and advantages of building parallel clusters on top of IEEE1394-1995. The main advantage is that minimal speed-up can be guaranteed, as shown on the basis of the parallel implementation of discrete Fourier transformation.*

Keywords
IEEE1394-1995, cluster computing, parallel systems, isochronous transfers, Linux

1 Introduction

Computer networks have to face different, partly contradictory demands. *High bandwidth* seems to be still the measure most people believe in, and most companies use as selling argument for their networks. However, for multimedia (or, more exactly, for continuous data) *guaranteed quality of service* (QoS) at the user-level is more important than raw speed. For parallel applications, user-level *end-to-end latency* is the most important measure. These divergences in the demands lead to a growing heterogeneity in local area network technology. The only common denominator is the requirement for low prices. Currently, we have no standard (or quasi standard) solution that fulfills all requirements for a reasonable price.

We assume that in the coming years a new de facto LAN/SAN standard arises, which is able to provide a good compromise in supporting all the above mentioned requirements. We investigate the IEEE1394-1995 standard (the "FireWire[1]") from this point of view, i.e. its ability to serve as a basis for an "all-round" standard network.

Ethernet is definitely the leading LAN technology, at least regarding the number of installations. There are a number of efforts to raise the bandwidth of Ethernet. 100 Mbps FastEthernet can be regarded as the new standard LAN

[1] FireWire is a trademark of Apple Computer, Inc.

technology (as being already less expensive than 10 Mbps Ethernet). The Gigabit Ethernet technology also arises, however, not without contradictions. Latency times seem to get much slower lower than bandwidth is getting higher. The Virtual Interface Architecture (VIA) [4] addresses the problem of low latency and therefore, Ethernet-based systems combined with VIA might have excellent performance in the future. Currently, however, there are no appropriate products yet available.

Some efforts have also be done to provide guaranteed QoS in an Ethernet-based system. In [14] a solution is suggested which requires the replacement of the Ethernet drivers without any change in the hardware. In [15, 12] the solution is extended in a way that not even the drivers must be replaced, rather a simple Ethernet switch is used (consisting of a PC with a number of Ethernet cards). These solutions provide satisfactory but by no means optimal results. Summarizing we can say that there is a certain probability that Ethernet will remain the ultimate LAN technology in the future, but this is by no means sure.

Other networks, especially Myrinet [2], provide high bandwidth and good latency times, but provide no guaranteed QoS. Moreover, although Myrinet can be regarded as a cheap solution if it is compared to supercomputers, it is expensive if it is compared to Ethernet. One of the most attractive features of Myrinet is its flexibility, which might be used to add guaranteed QoS to Myrinet.

The IEEE1394-1995 standard [7] addresses first of all the question of multimedia support and QoS. It has, however, some features, that qualify it also as a good basis for parallel computing. In the following we elaborate these features. If IEEE1394-1995 proves as really suitable for parallel computing then it could be an interesting, low-price candidate for future LAN technology.

Previous work covered the design and implementation of a driver for Intel and Alpha based computing nodes running the Linux operating system and optimized for asynchronous data transfers [10]. We use the Adaptec AHA-8940 1394-to-PCI host adapter which supports 200 Mbps data transfer rate at present.

2 The IEEE1394-1995 High Performance Serial Bus

2.1 Relative place

We start the investigation with a placement of IEEE1394-1995 among the most important networks used for parallel cluster computing.

Network comparisons regarding pure technical aspects can be found e.g. in [1] and [13]). Table 1 provides a short comparison of essential network technologies. The table reflects beside basic technical characteristics such as network structure, minimal one way latency, maximal access bandwidth and the existence of support for isochronous communication, also aspects of operating system support, manufacturer support and price.

There is obviously no "best" network. Regarding the different operational areas each network has its pros and cons. IEEE1394-1995 belongs to the bests in price, supports isochronous communication, has high bandwidth and low latency. Exactly these are the features that make the further investigation worthwhile.

Table 1. Comparison of networks for parallel cluster computing

	Fast-Ethernet	Gigabit-Ethernet	SCI	ATM	Myrinet	IEEE1394-1995
Network structure[a]	bus	bus	ring	switched	switched	bus
Min. one way latency[b]	20 μs	20 μs[c]	5 μs	120 μs	5 μs	15 μs
Max. access bandw.	100 Mbps	1 Gbps	4 Gbps	155 Mbps	1.2 Gbps	400 Mbps
Isochronous cap.				✓		✓
Cable len. per link[d]	200 m	200 m	10 m	100 m	10 m	4.8 m
Windows-NT supp.	high	high	low	medium	high	medium
Solaris support	high	medium	high	medium	high	null
Linux support	high	low	low	medium	high	low[e]
Manufacturers	a lot	many	few	many	1	few
Costs per link[f]	$500	$1500	$1000	$3000	$1800	$500

[a] original network structure, busses can also be switched of course
[b] regarding only the hardware dependent layers
[c] estimated value
[d] when using standard low cost cables
[e] development in progress [11]
[f] host adapter inclusive switching hardware

2.2 Features

The IEEE1394-1995 is a low-cost, high-performance ergonomic serial bus [7]. It was designed for operation both in the areas of industrial as in consumer electronics. Its architecture is compatible with other IEEE busses and standards (e.g. [6]) and implements a *memory read/write communication architecture* in contrast to conventional I/O-based communication, so distributed systems with global memory architectures can be mapped without great translation efforts. The shared memory architecture of IEEE1394-1995 is an architecture to provide economic interfaces and low latency. The address model is based on IEEE Std 1212 [6] (CSR Control and Status Register architecture), thus being compatible with SCI (Scalable Coherent Interface). IEEE1394-1995 uses IEEE-1212 "64-bit fixed" addressing, where the first 16 bits are used to represent the *node_ID*, thus allowing up to 64k nodes. Further the *node_ID* is divided into the 10-bit wide *bus_ID* and a 6 bit-wide *physical_ID*. Therefore up to 1023 busses[2], each having up to 63 nodes[3], can be interconnected. The remaining 48 bits are used for addressing the node *memory space*, the *private space* or the *register space*[4]. One or more nodes of the serial bus can be combined to form logical *modules*. An addressing scheme for determining the module number is not provided.

[2] bus # 1023 refers to the local bus
[3] node # 63 refers to the broadcast-address
[4] The *private space* is characterized by having 0xFFFFE and the IEEE Std 1212 *register space* by having 0xFFFFF as the leading 20 bits, thus remaining 28 bits for the *private address* resp. the *register address*.

The main feature distinguishing from most other communication technologies is the capability of the *isochronous transfer* of digital data, which is needed when working under *guaranteed timing* or *guaranteed bandwidth* is required, e.g. transfer of multimedia data. Although this feature is also available in the *Asynchronous Transfer Mode (ATM)*, ATM has the disadvantage over IEEE1394-1995 of the requirement of additional expensive switches with high latency.

The serial bus protocols are a set of three stacked layers. The lower two layers can be compared to the ISO/OSI layers 1-2. The highest layer is the so called *transaction layer* and provides *read, write* and *lock transactions*. It provides a path to the *isochronous resource manager* (IRM), which is in fact part of the control and status register (CSR) structure [6]. The middle layer (*link layer*) provides a one-way data transfer with confirmation of request and provides its service to the *transaction layer*. In difference to the *transaction layer* it provides an *isochronous data transfer service* directly to the application using periodic cycles with a cycle time t_{isoc} of 125 μs. A link-layer-transfer is called a *subaction*. The lowest layer (*physical layer*) translates the logical symbols into electrical symbols, arbitrates the bus and defines the mechanical interfaces for the Serial Bus.

2.3 Performance Measurement

Though the Linux IEEE1394-1995 driver is in a preliminary state, we have done some basic performance measurements on asynchronous transactions. The *read quadlet transaction* is used for minimum information transfer to access a single quadlet (four bytes) aligned register of the CSR structure. We measured the performance of consecutive read quadlet transactions, which in turn consist of an acknowledged read request and an acknowledged read response subaction, with the packet length of four resp. five quadlets. The equipment consists of Intel Pentiums 200 MHz running Linux with the 2.0 kernel. Further we use the Adaptec AHA-8940 1394-to-PCI host adapters with the 400 Mbps Adaptec AIC-5800 link controller chip and the 200 Mbps IBM 21S750PFB PHY chip. The tests were done with a data transfer rate of 200 Mbps. A 100000 transaction test showed an average single transaction latency of 129 μs. The latency consists of the link layer round trip time plus the transaction layer software overhead. For packets with little payload the software and communication overhead is rather high and we expect notably higher throughput with isochronous transfers and asynchronous packets with higher block size.

3 Transaction model for group communication

As mentioned in section 2.2 the highest layer of the Serial Bus protocol stack is the *transaction layer* which provides three different reliable unicast transaction types: the *read*, *write* and *lock* transactions. For unicast messages services like flow and error control are provided by the underlying *link layer* with *asynchronous subactions*.

The idea of group communication is to let processes communicate with a *group* of other processes *simultaneously*. Examples of group communication are replicated file systems (with coherent caches), replicated program executions (SPMD machines), teleconferencing and financial computing systems. Hardware support influences the efficiency of group communication. The bus architecture, as used at IEEE1394-1995, is an ideal vehicle for one-to-all group communication (broadcasting). Reliable broadcasting, which is related to atomic messages, is an extension to broadcasting and is required for most parallel applications. IEEE1394-1995 provides broadcast write transactions with an unacknowledged request subaction – the standard doesn't define how to complete the transaction to make it reliable. In general it is required to check the needs of the applications for realizing the appropriate broadcast method. For example file-streams can be broadcasted more efficiently using forward error correction mechanisms. In contrast cache coherency protocols have to use broadcasting with causality and total ordering semantics since several processes can transmit "memory-invalid-messages" simultaneously.

Protocols for group communication have been investigated very exhaustively, e.g. [3, 5, 8]. In this section we want to demonstrate a realization of the BB-method [8], a reliable multicast method, on top of the IEEE1394-1995 link/transaction layer. The CSR architecture can be used very efficiently.

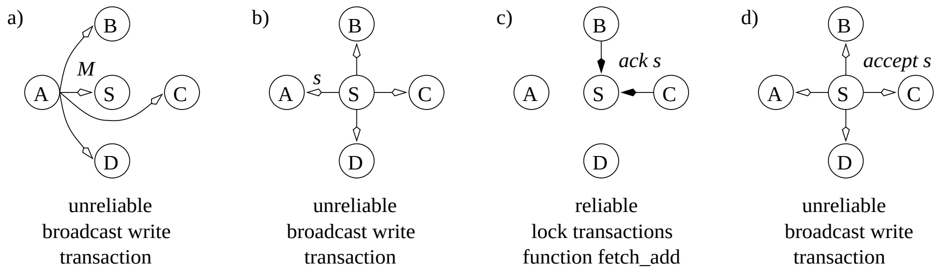

Fig. 1. Reliable broadcast method for IEEE1394-1995 using the BB method for five nodes with $r = 2$

The BB method assumes the existence of one processing node with an extra service called *sequencer* S. It is based on unreliable broadcast and reliable unicast messages. The procedure for a single broadcast is demonstrated in figure 1. After node A sends a broadcast of the message M (Fig.1a), S stores M and allocates a sequence number s, which is again broadcasted (Fig.1b). To cope with processor failures, r other processors also store the message and have to acknowledge s (Fig.1c). After receiving the r acknowledgements, S broadcasts an *accept s* message and the kernels are allowed to pass M to the application (Fig.1d). To assure that all processing nodes have received the broadcast messages they have to inform the sequencer about their highest received sequence number periodically, so the sequencer and the r other processors are allowed to discard messages from their buffers. In case of loss of a message by single receivers because of transmission errors or because of buffer overflows within the receivers, the lost message can be retransmitted using reliable unicast messages. The correct order of the received broadcast messages is checked by the receiver itself.

The read, write and lock transaction architecture of IEEE1394-1995 is well suited to efficiently implement the BB method. We suggest the following method:

Analogous to the bus manager election process a *sequencer* or *reliable broadcast manager* (RBM) is determined at startup or bus reset. The RBM implements two status registers, which can be read through the CSR address space, one for the actual sequence number and one for the sequence number of the oldest buffered message. These registers can be used to retransmit lost messages. Furthermore each node implements a *broadcast message counter* (BMC) in the RBMs address space, which holds the sequence number of the last successfully received broadcast message. The RBM has to analyze these locally stored BMCs periodically for being able to discard acknowledged messages. Faulty nodes can be also detected by this procedure.

For message broadcasting we use the standardized unacknowledged broadcast transactions. The "fetch_add" lock transaction adds a given value to a memory's value and can be efficiently used to implement the "*ack s*" messages. Every lock transaction decrements a counter, which was initialized with r. When the counter reaches zero, the RBM is allowed to complete the reliable broadcast transaction by broadcasting the "*accept s*" message.

4 Topologies

The obvious strategy of connecting all nodes to one serial bus is easy to realize, but can have major drawbacks regarding performance. One can separate bus segments by usage of *bus bridges*. Similar to Ethernet bridges, intra-bus traffic is not distributed across bridge ports. Usage of bus bridges means we have to keep the system free of cycles, thus disallowing to build complex communication networks. Therefore we suggest routing on higher layers and connecting nodes to more than one independent bus.

To minimize latency and maximize throughput we have to follow two conflicting targets:

– keep the number of nodes connected to a single bus as small as possible to avoid congestion on the bus
– keep the number of busses small to avoid having messages traveling over many hops which is time-consuming

We suggest building parallel computers by usage of nodes which are connected to more than one bus. Routing occurs by a bridge protocol or by yet a higher layer. We do not use bus bridges because they don't allow cycles and have no advantages over multi-homed nodes. Having the driver bridging seems to be the best solution since we don't need heavy-weight protocol stacks, we can implement our own bridge protocols and policies and can make use of the reliable group communication patterns directly.

For doing routing decisions, performance of standard drivers may be not adequate. In case of the Linux IEEE1394-1995 driver [10], a very low-level cut-through routing implementation, which bypasses the Serial Bus protocol stack is suggested. To achieve real high performance, additional hardware solutions (intelligent PCI-1394 adapters) are required.

The straightforward approach to build an IEEE1394-1995 based "supercomputer" is connecting individual nodes in the form of a grid. In this case vertical and horizontal connections are independent busses. Every host in the grid is therefore connected to two different busses and is capable of routing (figure 2).

Sending a message from node A to node B in a grid of N nodes can be accomplished by the network

- needing one routing decision if B is connected to one of the two busses A is directly connected to ($2\sqrt{N}-2$ nodes can be reached this way, without any intermediate hop)
- needing two routing decisions (one intermediate hop) if B is connected to any other bus in the network (the rest, or $N-2\sqrt{N}+1$ nodes are reached this way)

Routing is done, in the simplest case, only by that one such host which is connected both to the source and target node. This can be extended to arbitrary nodes, demanding a bridge protocol or advanced static policies.

Due to the efficiency of bus-based grids they may be completely satisfying for most networks. You can vary the grid structure by using a different – bigger or smaller – number of busses, thus leaving the "one row/column – one bus" policy.

However, as the node number increases we have to face other topologies, e.g. by connecting each node to three or more busses.

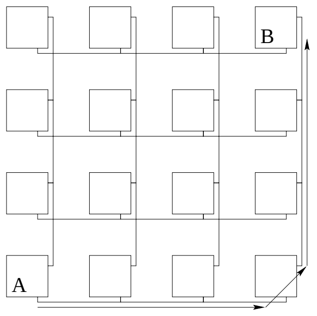

Fig. 2. Grid of 16 nodes; Node A sending a message to B

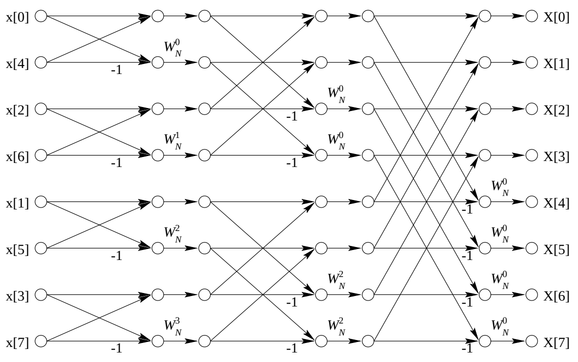

Fig. 3. Signal graph of a frequency decimated DFT-algorithm, $N = 8$

5 Parallel applications using isochronous transfers

One of the great characteristics of IEEE1394-1995 is the *isochronous transfer capability* of digital data. The original thought was to use this sort of transfer for time critical data such as video- and audio streams. The mechanism is to reserve guaranteed communication bandwidth and applications may transfer a limited size of information during time-slots. The idea was to provide guaranteed QoS and to reduce synchronization and buffer management tasks. In this section we show with the help of an easy example (parallel discrete Fourier transform (pDFT)) how isochronous transfers can be used to achieve a *guaranteed speedup* produced by parallel algorithms. Time critical applications (e.g. control systems) have to rely on end-to-end timing guarantees. In contrast to most of todays used networks (see table 1) IEEE1394-1995 is able to provide timely guaranteed delivery. In combination with real time task scheduling end-to-end scheduling can be realized, thus giving us the possibility to calculate the *guaranteed speedup*, which is indeed the lowest bound of the grade of parallelization.

5.1 DFT implementation

The discrete Fourier transform (DFT) is applied in the analysis, the design and implementation of time-critical signal processing procedures [9]. The goal is to calculate the DFT of limited sequences with the length N using the formula

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \qquad k = 0, 1, ..., N-1 \tag{1}$$

and $W_N = e^{-j\frac{2\pi}{N}}$. An optimal sequential algorithm, which takes advantage of the symmetrical and periodic characteristic of W_N^{kn} needs $O(N \log N)$ multiplications. The signal graph of an eight point frequency decimated DFT is shown in figure 3. The inputs are fed through a bit reversing operation (butterfly operation). The calculated results are spread to adders with a following multiplicator.

5.2 Parallel DFT implementation with isochronous channels

A parallelization can be realized by vertical distribution, so every processing element (PE) has to calculate $N/\#\mathrm{PE}$ complex sums and multiplications. For transmitting $N/\#\mathrm{PE}$ values at every isochronous cycle the PE has to reserve sizeof($value$) $* N/\#\mathrm{PE}$ channel capacity. For calculating the end-to-end processing time we assume a constant processing time t_p, which is guaranteed by the operating systems real time scheduler. The overall system's behavior is demonstrated in figure 4 for processing times smaller and larger than the isochronous cycle time t_{isoc}. As the result of delayed predecessors we observe that the maximum single idle time for synchronization is $2 * t_{isoc} = 250\mu s$. Although the average aggregate time will be much lower, a guarantee for the maximum aggregate time can be provided by $nt_p + 2(n-1)t_{isoc}$, where n is the

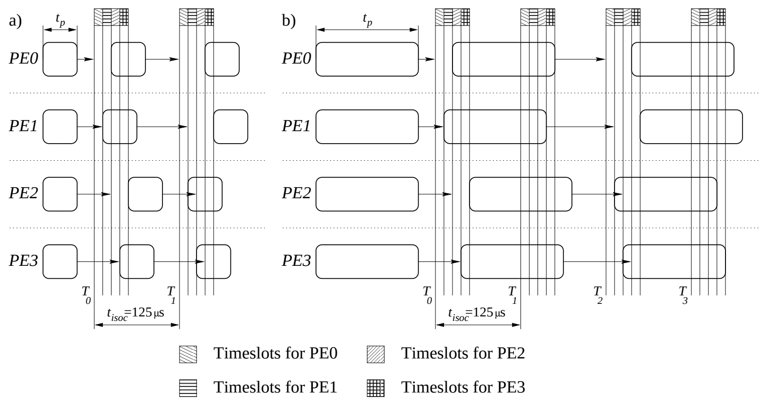

Fig. 4. Parallel implementation of the DFT using isochronous channels and $\#\text{PE} = 4$ a) $t_p < t_{isoc}$, b) $t_p > t_{isoc}$

number of processing steps. However, in the general case, when $N > \#\text{PE}$, only $\log_2 \#\text{PE}$ communication steps are needed, the maximum aggregate time will be $t_p \log_2 N + 2t_{isoc}(\log_2 \#\text{PE})$. The *guaranteed speedup* of the parallelization is shown in figure 5 and can be expressed by

$$guaranteed\_speedup_{isoPDFT} = \frac{\#\text{PE} \log_2 N}{\log_2 N + 2\frac{t_{isoc}}{t_p} \log_2 \#\text{PE}} \qquad (2)$$

6 Conclusion and further work

We have shown that the IEEE1394-1995 standard (and its followers) might play an important role in the LAN/SAN technology of the next decade. It not only supports the delivery of continuous data with guaranteed quality of service, but it also has the potential of serving as a high-performance, low-latency network, supporting cluster-based parallel computing. Even more, due to the combination of the above features, parallel clusters based on IEEE1394-1995 have the unique feature of *guaranteed speedup*.

We presented some basic figures about the performance of IEEE1394-1995. A next step should be to measure the performance with well-known benchmarks and with some typical applications.

It might be an interesting research to build an *IEEE1394-1995 switch*. Such a switch could be built quite simply with the help of a workstation with a greater number of IEEE1394-1995 adapter cards. With the help of such switches arbitrarily complex networks could be built cost effectively and efficiently.

References

1. Bal H.E., Hofman R., and Verstoep K., *A Comparison of Three High Speed Networks for Parallel Cluster Computing*, Workshop on Communication and Architec-

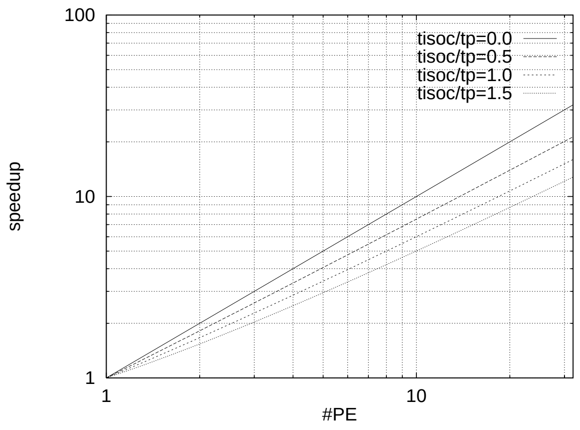

Fig. 5. Influence of the communication effort in the guaranteed speedup of the parallel DFT using isochronous channels. Using more PEs for parallelization means relative higher influence of the communication overhead.

tural Support for Network-based Parallel Computing (CANPC'97), pp. 184-197, San Antonio, Texas, February 1997

2. Boden N.J., Cohen D., Felderman, R.E., Kulawik A.E., Seitz C.L., Seizovic C.L., Su W., *Myrinet: A Gigabit-per-second Local Area Network*, IEEE Micro, 15(1): 29-36, February, 1995
3. Chang J., Maxemchuk N.F., *Reliable Broadcast Protocols*, ACM Transactions on Computer Systems, Vol.2, No.3, August 1984
4. Compaq, Intel, Microsoft, *Virtual Interface Architecture Specification Version 1.0*, http:www/viarch.org, Dec 16, 1997
5. Frank A.J., Wittie L.D., Bernstein A.J., *Multicast Communication on Network Computers*, IEEE Software, May 1985
6. IEEE Computer Society, *IEEE Std 1212, Control and Status Registers (CSR) Architecture for microcomputer buses*, New York, 1994
7. IEEE Computer Society, *IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus*, New York, August 1996
8. Kaashoek M.F., Tanenbaum A.S., *Efficient reliable group communication for distributed systems*, Rapport IR-295, Faculteit Wiskunde en Informatica, Vrije Universiteit, July 1992
9. Oppenheim A.V., Schafer R.W., *Discrete-Time Signal Processing*, Prentice Hall, 1989
10. Pirker E., Hölzl G., *The Design, Implementation and Operational Areas of the Linux IEEE-1394 Driver*, Institute of Information Technology, University Klagenfurt, Technical Reports, No 2-98, June 1998
11. Pirker E., *The Linux IEEE1394-1995 Subsystem*, `http://www.edu.uni-klu.ac.at/~epirker/ieee1394/`
12. Varadarajan S., Chiueh T., *EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch*, State University NY at Stony Brook, TR-45, January 1998
13. Varma A., Raghavendra C.S., *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, IEEE-press, 1993
14. Venkatramani C., Chiueh T., *The Design, Implementation and Evaluation of a software-based real-time Ethernet protocol*, ACM SIGCOM 95, 1995
15. Venkatramani C., Chiueh T., *Design and Implementation of a Real-Time Switch for Segmented E*, in International Conference on Parallel Processing (ICPP), August, 1998