Event Building in an Intelligent Network Interface Card for the LHCb DAQ System

J-P. Dufey, B. Jost, N. Neufeld and M. Zuin

Abstract—**LHCb is an experiment being constructed at CERN's LHC accelerator for the purpose of studying precisely the CP violation parameters in the B-B meson system. Triggering poses special problems since the interesting events containing B-mesons are immersed in a large background of inelastic p-p reactions. Therefore, a 4 level triggering scheme (Level-0 to Level-3) has been implemented. Full event building is performed between Level-1 and Level-2.**

Powerful embedded processors, used in modern intelligent Network Interface Cards, are attractive to use to handle the event building protocol in the high-speed data acquisition system. The implementation of an event building algorithm developed for a specific Gigabit Ethernet NIC is presented, and results from performance measurements are discussed.

I. Introduction

LHCb is one of the four experiments being constructed at CERN's LHC accelerator. It is a special purpose experiment designed to precisely study the CP violation parameters in B-meson decays by detecting many final states. The LHCb detector is a forward single dipole spectrometer, consisting of a micro-vertex detector, a tracking system, aerogel and gas RICH detectors, electromagnetic and hadron calorimeters, and a muon detector. The experiment is described in [1].

The expected b-quark production cross-section (500 μbarn, at a luminosity of 1.5×10^{32} cm^{-2}s^{-1}) leads to a rate of about 75 kHz of B-meson events which is immersed in a total inelastic rate of some 15 MHz. Typical branching ratios for the interesting final states of B-meson events lie between 10^{-5} and 10^{-4} leading to a rate of interesting events of ~5 Hz. For rare decay modes the branching ratios are as low as 10^{-9}.

After the selection done by the Level-0 and Level-1 triggers, implemented in hardware, the role of the Event Builder (EVB) system is to collect the data from the Front-end electronics, and assemble complete events in a "commodity processor" for further data reduction by the Level-2 and Level-3 software triggers.

The EVB system will be built around a switching network and will have to deal with a high rate (~40 kHz) of relatively small data packets (typically ~ 200-300 bytes). We are presently evaluating the use of Gigabit Ethernet as a possible technology for the implementation of this EVB system.

We present here a new solution for the implementation of the event building protocol, which takes advantage of the recent emergence of a new generation of Network Interface Controller (NIC) boards that include powerful programmable RISC processors. A simple event building protocol has been developed and implemented in the processor of a specific Gigabit Ethernet NIC card from Alteon [5]. The performance has been measured and a comparison with the more traditional implementation on a host computer is presented.

Manuscript received November 5th, 2000.

J-P. Dufey, B. Jost, N. Neufeld and M. Zuin
EP Division, CERN, CH-1211 Geneva, Switzerland
(Phone: +00 41 22 767 2130, e-mail: Niko.Neufeld@cern.ch).

II. Trigger and Data Acquisition System Overview

A four-level trigger scheme has been adopted. Level-0 and Level-1 triggers are implemented in hardware and reduce the event rate from 40 MHz down to 40 kHz. Level-0 rejects events on the basis of calorimeter and muon detector information while Level-1 uses data from the tracker to find indications for B decay vertices. The data stored during the fixed latency of those triggers are then zero-suppressed and undergo a first aggregation in the front end electronics. At this stage, the data are delivered over some 500 links in the form of relatively small (typically 200-300 bytes) data packets.

The task of the Event Building system is to transfer, for each event, the whole of data into a processor where 2 levels of software trigger are executed: the *Level-2* trigger algorithms, using the track vertex guidance from Level-1, are expected to reduce the rate to some 5 kHz by using part of the event data. The *Level-3* algorithms, based on full event reconstruction and applying physics cuts appropriate to the CP violation channels will deliver some 100-200 Hz of events that will be stored permanently.

More details on the LHCb data acquisition system can be found in [1]-[3].

III. Architecture of the Event Builder

The requirements imposed to the Event Building system are to collect, for each event, some 500 small (200-300 bytes) data packets produced at a rate of 40 kHz, and to deliver a complete event (100-150 Kbytes) to one processor in a large farm of some 4000 processors of 1000 MIPS. The aggregate data throughput is ~6 GB/s. A large switching network is required to cope with those requirements.

The Event Building system is composed of: (Fig. 1)

- The *Readout Units* (RUs) that interface the Front-end links with the Readout Network (RN). Each RU aggregates the data from several front-end links (up to 4) in order to adapt the data throughput to the capacity of a switching network port. Each *event fragment* built by the RU is shipped to the destination selected for each event (destination assignment function).
- The *Readout Network*, which provides support for event building by routing all fragments belonging to an

event to a particular destination port. The size of the switching network is approximately 125 X 125.

- The *Sub-Farm Controllers* (SFCs), which interface each output port of the RN with a local farm of processors that will run the higher-level triggers (Level-2 and Level-3). An SFC assembles the event fragments into complete events and delivers them to a selected processor, possibly controlling the load and activity of each processor in the sub-farm. It must handle the cases where event fragments are lost.

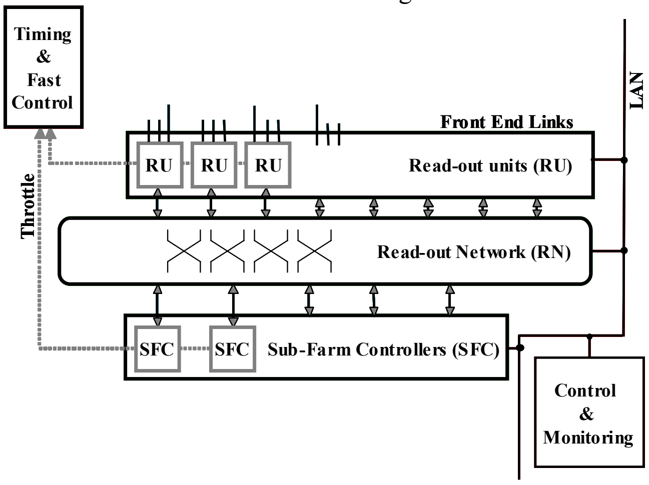

Fig. 1: Overall Architecture of the LHCb Event-Builder.

The event building architecture has been fixed as follows:

1. The *full event data are* transferred although Level-2 algorithms are based on partial data.
2. The *readout protocol* is a pure push-throughout protocol, where each RU pushes data through the Readout Network to a destination SFC, as soon as an event fragment is available.
3. The *destination assignment* is static. The algorithm governing the destination selection must be based on the event number and must be identical in all RUs.

This scheme has several nice features:

- No central control to communicate with sources and destinations on an event-by-event basis is needed. This leads to perfect scalability.
- The functionality of the RU is very simple in that it only has to multiplex the input links onto an output link using a FIFO to de-randomize the input traffic.
- Simple functionality of the SFC: it assembles event fragments arriving from RUs and sends complete events to one of the trigger processors, taking care of balancing the load.

The price to pay for the simplicity is:

- An elevated sustained bandwidth across the readout network.
- No direct feedback between sources and destinations of the RN. If anywhere in the system a buffer becomes too occupied, a general throttle signal is issued to the trigger to disable the flow of events.
- Overall performance is determined by the lowest performing sub-farm, hence some balancing of the processing power is required.

IV. Implementation of the Event Building System

A Readout Unit (RU) must handle up to 4 front-end links, each one delivering data packets of some 200 to 300 bytes at a rate of 40 kHz. The total rate of incoming packets can then reach 160 kHz. The RU aggregates those packets into an event fragment of up to typically 1200 bytes that is submitted to the switching network at a rate of 40 kHz. The resulting bandwidth is ~50 Mbytes/s or 400 Mbit/s.

As of today, the implementation of the aggregation of front-end data packets in the RU is foreseen to be in hardware in order to cope with a high rate up to 160 kHz. At the output of the RU, a software implementation of the event fragment delivery to the network is foreseen. This is the *upstream* or *source* part of what we call the *event building protocol* that will be executed by a processor in the RU.

The number of RU ports is 125 if we assume 500 front-end links and an aggregation of 4 links in each RU.

At the output of the switching network, the Sub-Farm Controllers receive event fragments at a rate, which depends on the relative numbers of SFCs and RUs. If the numbers are equal, the rate of event fragments arrival is 40 kHz. It can be reduced if needed by increasing the number of SFC ports. The number of SFCs also determines the rate of complete events produced by each SFC. This rate is 320 Hz for a "square" network (125 SFCs).

The software controlling the assembly of event fragments in the SFCs, for several events concurrently, and taking into account possible fragment losses, constitutes the *downstream* or *destination* part of the event building protocol.

As yet, no choice of technology has been made for the switching network. A likely candidate is Gigabit Ethernet. This paper presents a development carried out under the assumption that this technology is adopted.

Assuming switching network ports with 1 Gbit/s link speed, 4 front-end links per RU, and a uniform distribution of data on the front-end links, then the load factor per switch port would be ~25%. In reality the distribution of data will not be uniform so we shall at least impose an upper limit to the maximum load (e.g. 50%).

V. Event Building Protocols

The *source protocol* (implemented in the RU) performs only the fragment labelling and the destination assignment based, in its simplest form, on the event number.

The *destination protocol and algorithm* must be capable of handling several events concurrently since the random latencies of the event fragments (in the RUs and across the network) are not correlated.

Several event-building algorithms have been studied [4] implementing various strategies for fragment loss detection and recovery by the SFC. A simple detection of fragment loss can be implemented using a timeout based on the event number that can act as a clock. The arrival of a fragment belonging to an event not yet seen triggers an “ageing” of all events already in the process of being built. Normally, an event is complete as soon as the expected number of fragments has arrived. An event can run into “time-out” when its "age" exceeds some limit, in which case the algorithm considers the missing fragments as lost and ships the incomplete event, with a suitable warning to a processor. Late fragments are simply discarded. The upper limit for the maximum "age" is determined by the amount of buffer-space available.

The event building protocol comes on top of other standard network protocols that may include a *transport protocol* such as TCP/IP (Transmission Control Protocol over Internet Protocol) or UDP/IP (User Datagram Protocol over IP). TCP/IP guarantees the delivery of data packets whereas UDP does not. However it is well known that those protocols have a cost in terms of processing overhead and, consequently, lead to lower bandwidth utilisation. For packets with a transfer time smaller than the overhead, the frequency is limited to the inverse of the overhead time.

VI. Implementation of the Event Building Protocol

High rates of small data packets is the most challenging problem in the implementation of the event building protocols as the overheads due to their execution, both in the RUs and in te SFCs, will determine the performance of the whole system.

It has been mentioned that the transport protocol TCP/IP, while guaranteeing packet transmission, has a high cost in terms of overheads. It turns out that TCP/IP, and even UDP, in their current implementation, are not compatible with the requirements that we have to fulfil. Consequently we have decided to bypass those transport protocols, relying only on the "raw Ethernet" format. Considering the low load on the switching network and the fact that event building is the only (mostly 1-directional) traffic, we estimate that the probability to lose packets should be minimal. Investigations are currently under way to clarify this aspect of the problem.

The advent of Gigabit technology to the desktop necessitates a different approach from the hitherto per packet interruption of the host by the Network Interface Controller (NIC). For packets of the order of 500 bytes, at full bandwidth utilisation, the host would need to be interrupted every 4 μs, which, in addition to protocol handling, can either be impossible or would consume an unacceptable amount of system resources.

The industry has proposed a new concept of NICs to remedy this situation by providing strong support for those functionalities directly on the card: the interrupt rate in the host may be reduced by allowing packets on the receiving link (Rx) to be grouped when transferred to the host. This is referred to as *interrupt coalescence*. Multipurpose CPUs or ASICs are provided to calculate the layer 3/4 checksums (TCP or UDP and IP). Examples of those products can be found in [5] to [7].

NICs implementing multipurpose CPUs offer the possibility to upload protocol functions, normally performed by the operating system, to the NIC with the aim to enhance the performance, security, etc. of the high throughput links [8]. In our application, only one process needs to communicate with the NIC. We then proposed to upload to the NIC most of the event building protocol as well.

We present the results from a research aimed at testing this solution. We have chosen to implement the event building protocols in the CPU of a NIC, directly over the Ethernet protocol.

The protocols were first tested on PCs in order to determine the software and standard protocol (UDP or TCP) overheads. Then the code was adapted and ported to run on the processor imbedded in the NIC card and the performance has been measured. We present the NIC and its embedded processors, as well as the modifications to the firmware that we have implemented and give the results from performance measurements. A comparison with an implementation on the host computers is made.

The measurements were done over a point to point connection. No attempt has been made so far to interconnect the end nodes across a switch (which should not affect our present measurements) nor to measure the impact of parallel traffic in the switch.

A. The NIC architecture

We have selected a NIC based on the Alteon Tigon 2 ASIC [5]. Block diagrams of the NIC and Tigon are shown in Fig. 2

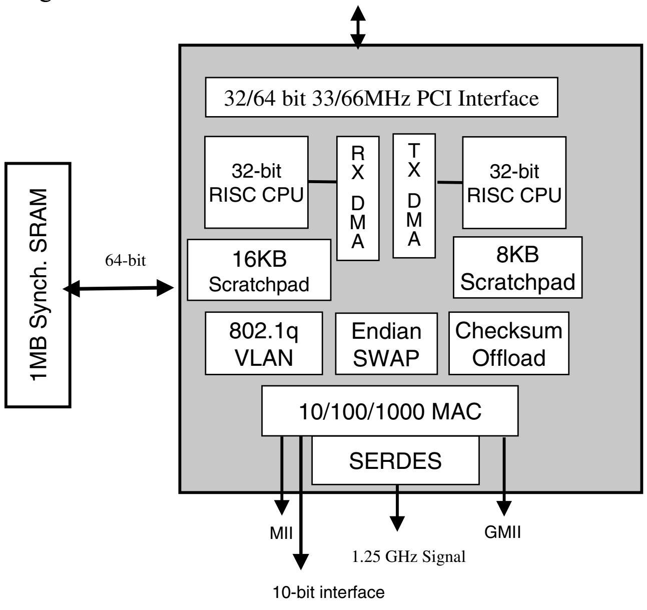

Fig. 2: Block Diagram of the Alteon Tigon 2 NIC

The Tigon 2 implements two R4000 MIPS type CPUs running at 88 MHz and two DMA engines which support scatter/gather at any byte boundaries. In the standard version of the firmware, one processor is devoted to Ethernet traffic management, to and from the MAC, while the second processor takes care of DMA (both directions) between the host and the NIC. The external SRAM, is accessed via a 64 bit wide memory bus. The Tigon is connected to the host via a PCI interface which can operate at 66 MHz, 64 bit wide. The chip incorporates an Ethernet MAC (Media Access Control) interface that supports 2 media attachment interfaces: MII (Media Independent Interface) for 10/100 Mbit/s and GMII (Gigabit Media Independent Interface) for 1000 Mbit/s. For more details on the physical interface, see [13].

The firmware provides an efficient API (Application Programmer Interface) for Ethernet. It supports auto-negotiation of link-speed at 10, 100 and 1000 Mbit/s and flow-control. It provides support for checksum calculations required for IP and TCP. It also manages a rather large buffer space, which can be used to collect several packets before interrupting the hosts, thus lowering the frequency of those interrupts.

There is no interrupt mechanism to interrupt the processors. Communication with the hardware is by means of event flags. It is the responsibility of the firmware to respond to events in due time. Events can be disabled and/or ignored.

The Tigon chip is implemented on the NICs produced by several manufacturers such as Netgear [9] and 3Com [10]. Intel's NIC [6] implements a different CPU.

B. *Test set-up and performance of the standard firmware*

A system has been set-up to test the performance of the NIC cards and to measure the performance of the event building code. It is shown in Fig. 3.

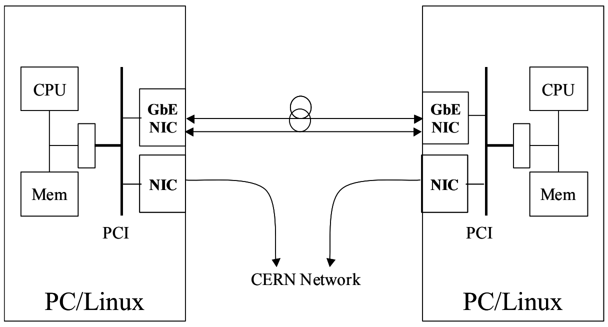

Fig. 3: Test set-up for Alteon NIC measurements

Two PCs, each one equipped with a Tigon 2 based NIC (labelled GbE in the figure), are inter-connected by optical links. The PCs run Linux and a dedicated device driver is used to download the cross-compiled firmware into the NIC. The driver also supports an interface to the GNU debugger (gdb) operated as a remote debugger.

As a first test of the capabilities of the NIC processors and the firmware development environment, a measurement of the raw Ethernet throughput, as a function of the packet size, has been done. The Ethernet packets are generated in the NIC acting as sender and are discarded by the receiver NIC, so no traffic goes over the host-NIC interface. Ethernet packets with a size > 1500 bytes are managed by means of the non-standard "jumbo frames" technology.

The results are presented in Fig 4. The minimum frame size is 64 bytes. A fit to the formula (1) for the effective throughput y was performed, where x is the packet size, b the nominal link bandwidth (125 MB/s) and a the overhead time during which no data can be transferred:

$$y = \frac{x}{a + x / b} \qquad (1)$$

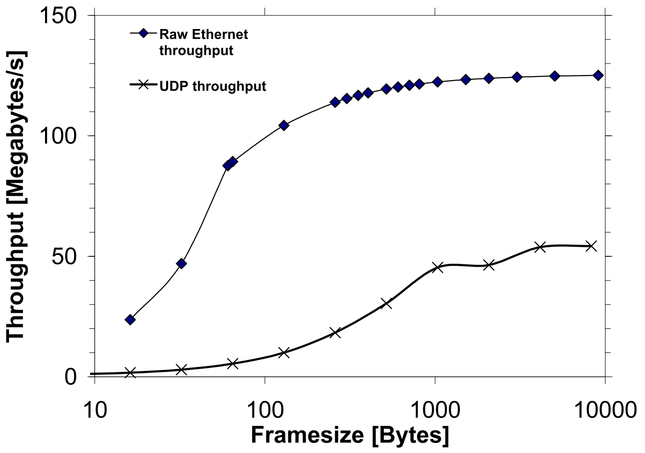

Fig. 4: Effective throughput using raw Ethernet frames and UDP, as a function of packet size. Diamonds are throughput measurements obtained for raw Ethernet frames. Stars show the throughput using UDP .

The fitted value for the overhead a is 0.2 µs. This overhead consists mostly of the preamble, the Start-of-Frame delimiter and the inter-frame gap, as described, for example, in [12]. The maximum achievable frequency of frame transfer is therefore approximately 1.4 MHz for the minimum size packets. For frames of 1000 bytes, 97% of the full bandwidth can be used.

Throughput has also been measured from host to host, using the "light" UDP/IP transport protocol. The throughput for packets of 1000 bytes does not exceed 40% of the available bandwidth, due to the protocol overheads. However it should be noted that the UDP performance depends also on the operating system (in this case Linux 2.2.12), on the PC hardware and its setting (most importantly the PCI bus, its setting and speed) and also on the various tuning parameters available (buffer size, IRQ coalescence etc.). No attempt was made to tune the UDP parameters for the throughput measurements presented here.

C. *Implementation of the Event Building Protocol in the NIC*

In the event-building application the “infrastructure” part of the firmware was only slightly modified, namely the initialisation, event dispatching, auto-negotiation and auto-sensing of link changes. The reception and transmit handlers were replaced by the entry points for the traffic generator in the sending node, and the event-builder in the destination node.

The tasks assigned to the 2 processors are the same as in the standard firmware, namely one taking care of the Ethernet frames to and from the MAC and the other one controlling the DMA with the host. It is recalled that in one NIC the traffic is unidirectional: from host to network in a RU's NIC and from network to host in a SFC's NIC.

Instead of the standard way of transmitting consecutive Ethernet packets into pre-allocated buffers in the host we execute, for each event, a chained DMA transaction, which gathers the ordered sequence of event fragments in a contiguous memory in the host. This avoids time-consuming copying around of data during event building.

The process in the NIC is never interrupted, but it can interrupt the host when new data are being transferred. The input and output buffers are implemented as circular buffers of descriptors and the relative position of consumer and producer pointers allow to locate new data.

In order to optimise the performance of the event building protocol, the most time critical parts of the code and the administrative data structures have been moved into the internal "scratch-pad" memory of the Tigon (Fig. 2) which can be accessed at processor speed, without any wait cycles.

D. Performance measurements in the NIC

For the evaluation of the event building protocol embedded in the NIC cards, the test set-up described in Fig. 3 was used. Event building is emulated on a point-to-point connection. The sending node (NIC) emulates multiple sources and the receiving node performs the event building of the fragments received. Apart from the protocol headers, the data contained in the packets is irrelevant. For this measurement, no data transfer to the NIC's host occurs, in particular, the complete events are discarded without being transferred to the host. The reason for this restriction is that we are interested in the measurement of the protocol overhead due to the NIC's processor only, while the transfer to (from) the host would distort the result.

This measurement has shown that the average overhead time to handle one event fragment (independent of its size) is ~9 µs in the receiver node. Considering that the NIC processor can handle a fragment concurrently with the data input of the next fragment, in the receiver, this value of overhead means that fragments, up to ~ 1100 bytes, can be received at a rate of more than 100 kHz. The performance is well beyond the requirement of achieving 40 kHz for 1000 bytes event fragments.

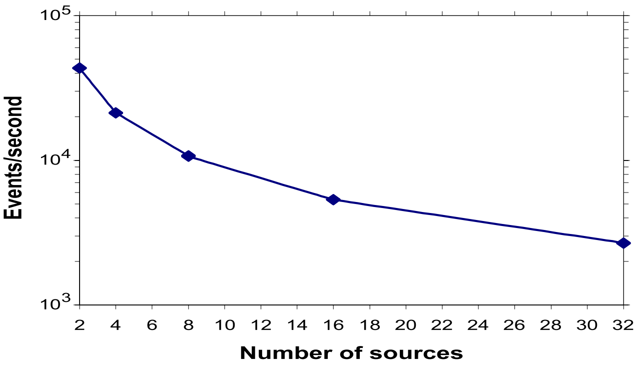

Fig. 5: Measurement of the maximum number of events built per second in one destination, as a function of the number of sources. The fragment size is constant and equal to 1'500 bytes.

Fig. 5 shows a measurement, on the point to point emulation of an N X N event builder, of the maximum rate at which full events could be delivered at a destination, for an event fragment size of ~1500 bytes, as a function of N. This rate decreases as 1/N, from ~ 42000/s for 2 sources to 2600/s for 32 sources.

E. Performance measurements for the event building protocol running on the host computers.

The same event building code, as used for the test of embedded event building, has been run on the 2 host processors of the test bed (233 MHz, running Linux). One processor generates the event fragments, emulating a variable number of sources, the other performs the event building. The data transfer used TCP/IP sockets. In this configuration we measured the pure software overhead, for handling one fragment, to be 3.2 µs, (to be compared with 9 µs on the 88 MHz processor of the NIC). However, the average total time to handle a minimal fragment (64 bytes), including overheads coming from the TCP/IP protocol and the operating system, is 9.3 µs, thus showing that the cost of the transport protocol and processor interrupt is 6.2 µs for the minimum size packets. It should be noted that the TCP/IP part of the overhead benefits from the support of the NIC, as mentioned earlier. For larger packets, the overhead, not counting the data transfer, is expected to increase proportionally to the packet size, due, in particular, to data copy between kernel and user space [11].

VII. Comparison and Conclusion

We compared one implementation of the event building protocol in the host computers, using TCP/IP (supported by the firmware running in the NIC) with an implementation directly in the NIC relying only on the Ethernet protocol.

The pure software overhead due to the execution of the event building protocol is larger in the NIC than on the host, due to the slower speed of the embedded processor. Nevertheless, the required performance is achieved and is even exceeded by a factor > 2 for packets of 1000 bytes.

The most important benefit when embedding the event building protocol in the NIC is to protect the host computer from the heavy load of handling the fast traffic of small data packets in terms of system overhead. Thus the SFC processor can be devoted to management tasks of the sub-farm of processors, remembering that this processor has to control some 30 processors.

Future revisions of the requirements might lead to an increased rate of event triggers delivered by the Level-1 trigger, probably by a factor ~ 2. In this case the NIC implementation would offer the additional advantage of a smaller overall overhead that permits to support a rate of more than 100 kHz of fragments of 1100 bytes.

The good performance of the embedded event building relies, for a large part, on the assumption that no transport protocol is required. This hypothesis still needs to be confirmed by the study of the traffic in the switching network.

VIII. Summary

Embedded event building has been studied for a specific Gigabit Ethernet NIC that implements a general purpose CPU. The feasibility of event building at a rate of 100 kHz, for fragments of 1000 bytes, has been demonstrated. The average time for processing one fragment in the destination NIC is of the order of 9 μs. The implementation of the event building protocol in the NIC processor results in a considerable offload for the host processor, which is interrupted only at the rate of completed events (a few hundred Hz). Consequently host CPU power is saved that can be devoted to the management of the sub-farm processors and to the data distribution.

IX. References

[1] *LHCb Technical Proposal*, CERN/LHCC 98-4, Feb. 1998.

[2] J-P. Dufey, M. Frank, F. Harris, J. Harvey, B. Jost, P. Mato and H. Mueller, "The LHCb Trigger and Data Acquisition System," *IEEE Trans. Nucl.. Sci.*, vol. 47, no. 2, pp. 86-90, April 2000.

[3] B. Jost, "The LHCb DAQ system," in Conference Record of 2000 IEEE Nuclear Science Symposium and Medical Imaging Conference, 2000 [CD-ROM].

[4] M. Zuin, "Embedded Event Building on a Gigabit Ethernet Network for the LHCb Experiment," Tesi di Laurea, Nov. 2000, Università Ca`Foscari – Venezia, unpublished.

[5] Alteon WebSystems, *Tigon/PCI Ethernet Controller,* rev 1.4, Aug. 1997, [Online]. Available: www.alteonwebsystems.com (from April 2001 this site just provided a link to a 3Com site).

[6] Intel Gigabit Ethernet NIC PRO/1000. [Online]: Available www.intel.com/network/products/server_adapters.htm

[7] SysKonnect SK-9843-SX Gigabit Ethernet Adapter. [Online]. Available: www.syskonnect.com/index.htm

[8] I. Pratt and K. Fraser, "Arsenic: a User-accessible Gigabit Ethernet Interface," presented at IEEE INFOCOM 2001. Available: infocom.ucsd.edu/papers/394-3981268191.pdf

[9] Netgear GA620 Gigabit Ethernet Adapter. [Online]. Available: www.netgear.com/adapters_main.asp.

[10] 3Com 3C985-SX Gigabit Ethernet Adapter. [Online]. Available: www.3com.com.

[11] D. Clark, L. van Jacobson, J. Romkey and H. Salwen, "An Analysis of TCP processing overhead," *IEEE Commun. Mag*., vol. 27, no. 6, pp. 23-29, June 1989.

[12] R. Seifert, *Gigabit Ethernet.* Reading MA, 1998.

[13] Ch.E. Spurgeon, *Ethernet, the Definitive Guide*. O'Reilly & Associates Inc, CA, 2000.