The LHCb Eventbuilder: Design, Implementation and Operational Experience

Niko Neufeld, Markus Frank, Jean-Christophe Garnier, Clara Gaspar, Richard Jacobsson, Beat Jost, and Guoming Liu

***Abstract*—The LHCb event-builder is implemented using a large Gigabit Ethernet network using a push-protocol for a single stage read-out at a 1 MHz event injection rate. The destination assignment and dynamic load-balancing are facilitated by LHCb's Timing and Fast Control system. The assembly of event fragments is done on each event-filter farm node instead of having dedicated builder units. The design of the event-builder will be shortly described, followed by a description of the implementation, the protocol used and the performance during first data taking. The emphasis will be on the experience we gained during the running of such a large event-building network. We will discuss the problems we encountered and how we overcame them.**

***Index Terms*—Data acquisition, ethernet networks, high-speed networks.**

I. Introduction

LHCb [1] is one of four large experiments at the Large Hadron Collider (LHC). It is dedicated to the precise study of the difference between matter and anti-matter (technically called CP-violation) using the large amounts of B-mesons produced in the LHC. Operating at a lower luminosity than the other experiments means that per bunch-crossing there is less activity in the detector and consequently the size of an event is smaller. The complex signature of the decay of B-mesons is such that they can only be selected efficiently using high-level algorithms run on standard servers. Consequently a large number of events need to be read out into a computer farm.

The data acquisition is therefore characterised by a small event-size after zero-suppression of 35 kB and a high maximum rate of events of 1 MHz. The topography of the detector and different requirements on data-processing result in a relatively high number of data-sources (or front-end links) of approximately 300. Fragments from all read-out boards must reach one of up to 2000 servers that are required to select interesting physics events. The total amount of data makes a Local Area Network the only reasonable option for such a data acquisition. After a brief evaluation of the ATM [2] and Myrinet [3] technologies, Gigabit Ethernet [4] has been chosen as the link technology.

The following facts are important for the final architecture:

- LHCb settled on a common read-out board for all sub-detectors,[1] which was designed to be able to send its data via Gigabit Ethernet to the DAQ. This read-out board is called TELL1 [5].
- The physical installation of the entire system, *including the event-filter farm*, is in the underground area, close to the detector. The maximum distance between the off-detector read-out electronics and the event-filter farm is 36 m, well within the limit of 90 m between patch-panels for 1000 BaseT [4]. Thus we use 1000 BaseT for all connections.
- The small average event size per source of about 100 bytes is only slightly above the minimal Ethernet frame-size of 64 bytes. To send a 100 byte message over the Ethernet requires 126 bytes (i.e., a 26% overhead!). Adding a minimally useful header which is required for event-building, quickly brings this overhead to 40 or even 50%. The message size is therefore increased by merging event-fragments belonging to several consecutive collision events. This is done in the read-out boards, which are connected directly to the DAQ network. All merged fragments are sent to the same receiver.

Originally, it had been foreseen to use dedicated event-builder PCs to merge the fragments from read-out boards and send them to dedicated farmnodes for selection. The estimated data-rate for such a PC was about 2.4 Gb/s full-duplex, which seemed not quite feasible with PCs available in 2005. Later we could demonstrate about 1.6 Gb/s for a candidate system [6]. The event-builder PC was consequently removed and the event-building has been implemented as a dedicated process in each node of the event-filter farm.

II. The Architecture of the LHCb Event-Builder

The data-flow of the event-builder is shown in Fig. 1. The detector front-end cards send their data to the read-out boards. The data-flow on this link is completely deterministic as no zero-suppression is done on the front-end cards themselves. The buffer-status of the front-ends can thus be centrally *emulated* to avoid congestion. This is one of the core functionalities of the Timing and Fast Control system (TFC) [7]. Thus no flow-control is needed on the individual links.

In the read-out boards the data are zero-suppressed and re-formatted for transmission to the DAQ. Event-fragments from several triggers are merged into a single message. Since the processing-time on these boards is not deterministic no emulation

Manuscript received June 11, 2010; revised November 23, 2010, February 26, 2011; accepted March 28, 2011. Date of publication June 07, 2011; date of current version August 17, 2011.

The authors are with the CERN, PH Department, Geneva, Switzerland (e-mail: niko.neufeld@cern.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2011.2151876

[1]One sub-detector uses a functionally equivalent board.

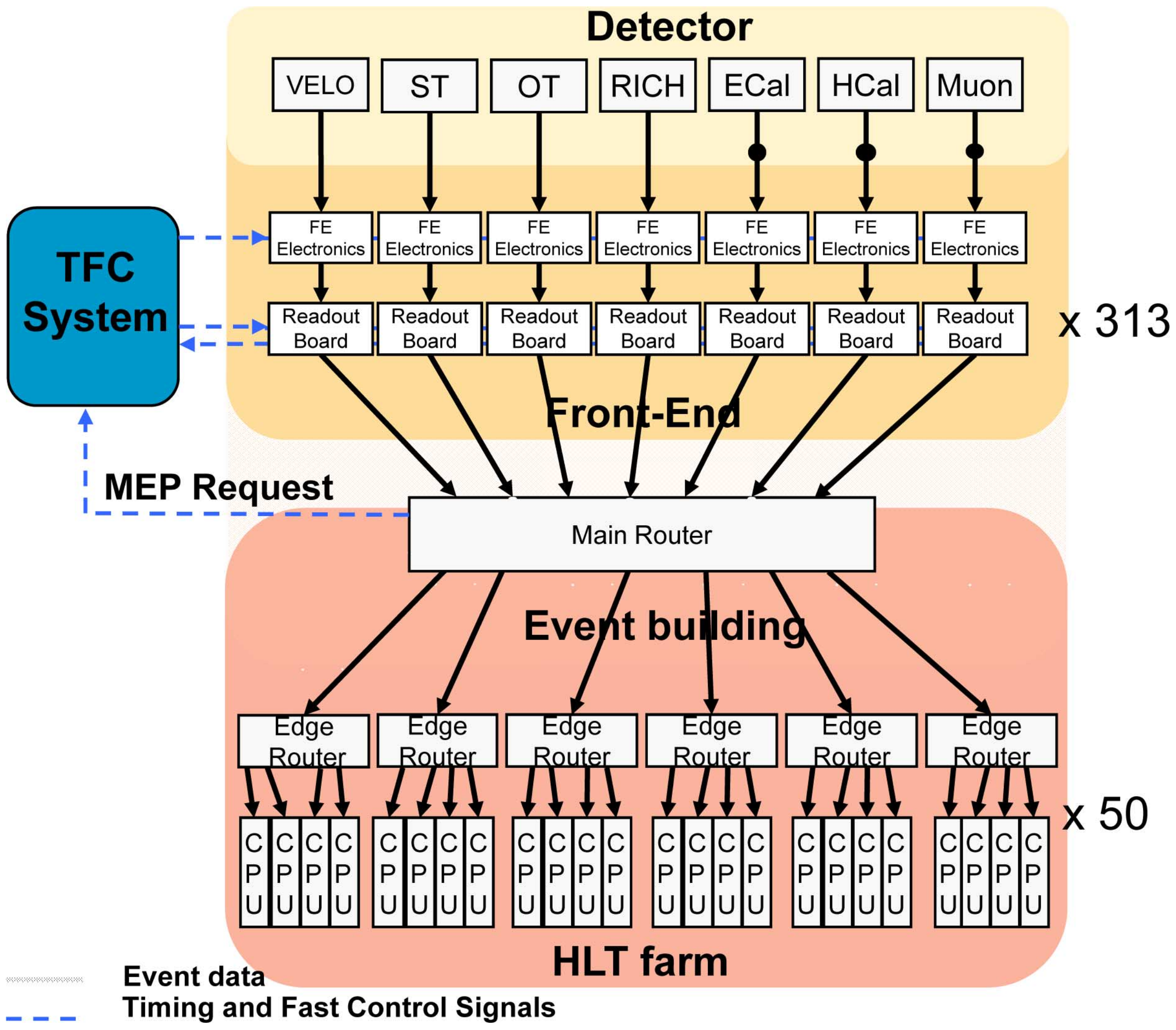

Fig. 1. Architecture of the LHCb event-builder.

is possible and the buffer status needs to be *protected* via a dedicated throttle network, which uses a simple differential line to transmit an active low signal. This throttle network feeds the logical OR of all read-out boards to the TFC system, which will disable the trigger until the throttle signal is lifted. It is the responsability of each board's implementation to make sure that the throttle signal is asserted early enough to avoid problems because of events arriving before the trigger has been disabled.

A. *Event-Building Protocol*

The read-out boards are driven by FPGAs and have no operating system to assist in handling a complex protocol such as TCP/IP. To keep the logic on the read-out boards simple, a very light-weight protocol is required. We therefore designed a simple push-protocol on top of the Internet Protocol, IPv4 [8], which we call MEP (Multi Event-fragment Packet). A MEP is a datagram which contains up to 16 event-fragments, that is data belonging to 16 triggers. MEP is similar to UDP but does not have its own header check-sum nor does it know the concept of ports. MEP contains three essential information items for event-building:

1) The number of event-fragments
2) The full event-ID of the first event-fragment,[2] which is a monotonically increasing number which identifies the event-fragment belonging to a certain trigger
3) The least significant 16 bits of every other event fragment in the packet
4) The size of the event-fragments in bytes

Following the OSI model [9], information that is available at the underlying IP layer is not repeated. For example, in the event-builder process on the farm the origin of the data is established by checking the IP source address of the received datagram. A MEP datagram must fit into a single IPv4 packet, resulting in a maximum MEP size of 64 kB.

The TFC system assigns the destination farmnode via the TTC link to each readout-board. For this purpose the TFC takes the role of a data-flow manager. The farmnodes announce their availability to the TFC by sending it a credit token via Ethernet that indicates the number of MEPs, i.e., event datagrams, they are willing to accept. After a certain number of triggers the TFC will broadcast the IP address of one of the nodes to all

[2]The event-ID is an unsigned 32-bit number. Wrap-arounds are allowed and are unproblematic as the event-building process does not correlate events with different event-IDs.

TABLE I
IMPORTANT NUMBERS ABOUT THE LHCB EVENT-BUILDER

Total average event-size	35 kB
Maximum event rate	1 MHz
Average message rate per read-out board	100 kHz
Number of read-out boards	308
Number of links into core DAQ router	1260
Number of edge routers	50
Number of links between edge router and core router	600
Number of farmnodes	550 (up to 2000)

the read-out boards via the TTC network. The boards will then send the MEP datagrams to this node without any further delay or back-pressure.

B. Readout Network

It is clear from the preceding discussion, that while the farmnodes can protect themselves from overload by sending or not sending credits to the TFC, the network has no such protection. It has to absorb the full in-rush of packets from all the read-out boards. The problem is exacerbated by the fact that the TFC system synchronously[3] distributes the packet destinations, which causes a lot of packets with the *same* destination address to arrive in the network at the same moment.

Modern Ethernet switches buffer data at the output[4] to avoid head of line blocking at the ingress. In the LHCb architecture at the output however is a computer, the farmnode, which has a single Gigabit connection. So 300 devices send at the same time to the same output destination, creating a 300 to one over-commitment on the output link, albeit for a relatively short duration. This almost ressembles a denial-of-service attack scenario and many Ethernet switches react by simply dropping packets, which is allowed by the Ethernet standard. The MEP protocol does not foresee any re-transmission or traffic modulation, so such a packet loss would be disastrous.

We could identify only one device, a very large core router[5] capable of sustaining this violent traffic. It is so large in fact that a single unit is sufficient to absorb the traffic from all read-out boards.

However, it has 1260 Gigabit Ethernet ports and thus it is not possible to connect all the required farmnodes. The resources of LHCb did not permit extending the core by buying more of the very expensive core devices. Starting from the observation that the bandwidth of a single core router is sufficient for LHCb's data traffic, but the number of ports over which this traffic is distributed is not, it was decided to put edge-switches between the core router and the farmnodes. These are connected with a number of up-links to the core and distribute the traffic to farmnodes. The ratio between up-link capacity and farmnode link-capacity is about three. Some key figures characterising the LHCb event-builder can be found in Table I.

[3]This is because the TFC system uses the TTC [10] technology, conceived for the distribution of timing and trigger signals in the LHC experiments.

[4]At least conceptually, there are many ways of organizing the buffer-memory.

[5]Routers forward packets based on the IP information and not based on the addresses in the Ethernet header.

III. IMPLEMENTATION

The read-out boards, the DAQ network and the event-filter farm are all installed in three floors of electronics barracks in the experiment cavern of the LHCb experiment 100 m underground. Each read-out board is equipped with four Gigabit Ethernet ports for sending data. The number of links used (between one and four) to connect a read-out board to the DAQ network depends on the expected amount of output data of the board. The links are treated in a way similar to a Link Aggregation Group (LAG). This means that each Ethernet frame that is transmitted to the DAQ network, will be sent in a round-robin fashion on one of the activated links. A different Ethernet source address is used on each link, but no attempt is made to fully implement link-aggregation according to IEEE 802.3ad. Each board is assigned a single fixed IP source address from a class-B subnet. The destination address of the farmnode is constructed from a constant pre-fix and 12 bits broadcast by the TFC system.

All links from the read-out boards are connected to the core router directly so that it can absorb the packet bursts in its large egress buffer memory. The core router is connected to each of the 50 edge routers via two link-aggregation groups (LAG) of six links each.[6] Cost prevented the use of a pair of 10-Gigabit connections. The edge routers have 48 Gigabit Ethernet ports.

In the configuration of the routers it was very important to provide the maximum amount of buffer memory for the output. This can be achieved by minimising the number of priority queues, because typically each queue will absorb a fixed amount of memory.

In each farmnode there is a single process which receives events from the network and puts them into a shared memory. This process is called MEPRx. The architecture of the tasks on the farmnode is described elsewhere [11]. Since MEP does not use ports MEPRx opens a raw IP socket and registers itself to a special IP protocol number (0xCB). Opening raw-sockets normally requires root privileges, which is inconvenient. To avoid this a small patch is deployed on the farmnodes as a kernel-module, which allows user processes to open raw sockets. The traffic coming to the farmnode is very bursty. Thus, in order not to loose packets in the kernel, in particular during IP re-assembly, we needed to tune the receive memory, several time-out parameters and also the number of descriptors for the Direct Memory Access (DMA) buffers in the Network Interface Card (NIC). This tuning has to be redone every time a major new operating system version is installed. Typically we have configured 8 MB of receive space and up to 4000 pending packets. With these settings we have not observed losses in the kernel itself. It is important to tune this correctly as unfortunately many of the drops in the kernel are silent, which is perfectly allowed by the IPv4 specifications, since IP is not a reliable protocol.

The MEPRx process does a lot of verification and syntactic checks on the data. It sends a request for new MEPs to the TFC via Ethernet whenever it has successfully acquired space in the shared memory to receive a complete set of MEP datagrams. Back-pressure is implemented indirectly via the TFC system: when the event-data on a farmnode cannot be processed quickly enough or accepted events cannot be sent onwards, the buffers

[6]Currently only eight links are in use.

in the node will fill up. The MEPRx process will then at some point fail to acquire memory and consequently not send a MEP request. When the number of available destinations reaches a low watermark, the TFC system, which is in control of the distribution of trigger decisions, will start to throttle the trigger. This stops the data-flow from the front-end electronics into the read-out boards.

Event-building is completed once MEP datagrams from all read-out boards have been received. On start of run several events are requested at once so to avoid idle-time waiting for data. Several events can be built in parallel. This allows coping with late arriving MEP datagrams, even though this should normally not happen in a large enough farm, provided that the distribution among farmnodes is flat. After a defined time-out has been reached or a corrupted or a truncated MEP is received the MEP is declared incomplete and is discarded. A single missing or corrupted MEP from any single source will cause the entire collection of events represented by the MEPs to be discarded. The LHCb trigger algorithms cannot work with incomplete information since data from the entire detector are required to identify interesting events. Currently it is up to the person on shift to judge when problems with discarded events require corrective action. Since the problems coming from the data acquisition system itself, such as the network, have been solved, in the vast majority of the cases the problem can be attributed to a specific sub-detector or read-out board. Most often the problem can be solved by pausing the trigger, resetting the corresponding part and re-enabling the trigger. The dead-time incurred by such an operation varies between a few seconds and a few minutes depending on the reconfiguration time of the sub-system in question.

IV. Performance

The system is designed for handling events at a maximum rate of 1 MHz and a size of 35 kB. Typically 10 events are packed into one MEP so that the total datagram rate from 300 read-out boards at nominal running is 30×10^6 s^{-1}, corresponding to an aggregated bandwidth of 35 GB/s. We do not yet have real collision data from a 1 MHz run, because the LHC up to now does not provide a sufficient number of collisions. For this rate, tests have been done using the data-generator mode of the read-out boards.

There are two main aspects to the performance of the system:

1) The loss-less transport of data at any rate
2) The resource-usage in the various system components

The loss-rate is currently one incomplete MEP per minute.[7] The reasons for packet loss that have been identified so far and their solutions will be discussed in the next section.

A. *Buffer and Memory Usage*

As is typical for a push-architecture buffer-sizes increase in the direction of the data-flow. Consequently the output buffer on the read-out boards is only 128 kB. The read-out boards however have a fast asynchronous method of disabling the trigger and with it the input data flow, so this buffer is well protected. The next buffer is in the main router. The effective buffer-size available for data is architecture specific and vendors are usually unwilling to disclose any details. When evaluating suitable devices, we have tried to measure their buffer-size with a simple minded approach based on flow-control. Experience has later taught us that this method tends to overestimate the effectively available buffer space. For our specific router we see 256 MB of shared buffer for a set of 48 ports, which corresponds to 3.5% of the total ports in the core router. In each-port set we connect both read-out boards and farmnodes. Ports connected to read-out boards do not use any buffer because this router uses virtual output queueing. So the shared output buffer is only used by the ports connected to farmnodes. Fig. 2 shows the buffer-occupancy from a high-rate test run.[8] It seems clear that even multiplying by three there is quite some margin and consequently there should be no packet drop due to buffer overruns.

In the farmnodes there are four buffers involved.

1) The packet buffers in the Network Interface Card (NIC)
2) The kernel buffers (`sk_buf` structures) which are allocated from the so-called slab allocator[9] and get the data via DMA. Each of them holds data belonging to one Ethernet frame.
3) The socket buffers associated to an application (in this case the MEPRx process). This buffer holds the complete re-assembled MEP datagrams.
4) The shared-memory buffer which MEPRx uses to distribute the event-data to the actual trigger processes

The different layers involved in the data transport in a farmnode are shown in Fig. 3. Item 1 is a hardware feature and cannot be changed, however it is important to make sure that a large number of buffers is configured. We set the number of buffers[10] to the maximum, which is supported by the network interface adapter we use. This maximum is around 1000. For item 2 there is no way to directly monitor the occupancy as it comes from a shared kernel pool. Experimentally we see no packet loss with 4000 packets available per network device while at values of 2000 packets losses were still observed. For item 3 again there is no real monitoring possible, but empirically a size of about six MB has proved to be sufficient. The buffer in item 4 has to absorb the fluctuations in processing time from the trigger processes and the time it takes to receive the MEP datagrams from all read-out boards. We have conservatively chosen it such that it has space for three worst-case events. A worst-case event is an event in which each read-out board sends the maximum size packet, that is 64 kB. The buffer is therefore set to 21 MB. In practice the occupancy is low, unless there is back-pressure from processes lower in the stack. In total one instance of the MEPRx process requires 79 MB of RAM out of which 68 MB is shared

[7]This has been measured while running at approximately 500 kHz event rate, corresponding to 50 kHz MEP rate.

[8]Unfortunately this measurement is destructive, because the firmware in the router must be frozen to read these internal registers. It is currently impossible to passively monitor the buffer occupancy. The figure therefore represents a snapshot, after the system has achieved a steady state.

[9]A memory management sub-system of Linux, which provides blocks of memory suitable for useage with DMA capable hardware.

[10]Technically this is the number of DMA descriptors handled by the card.

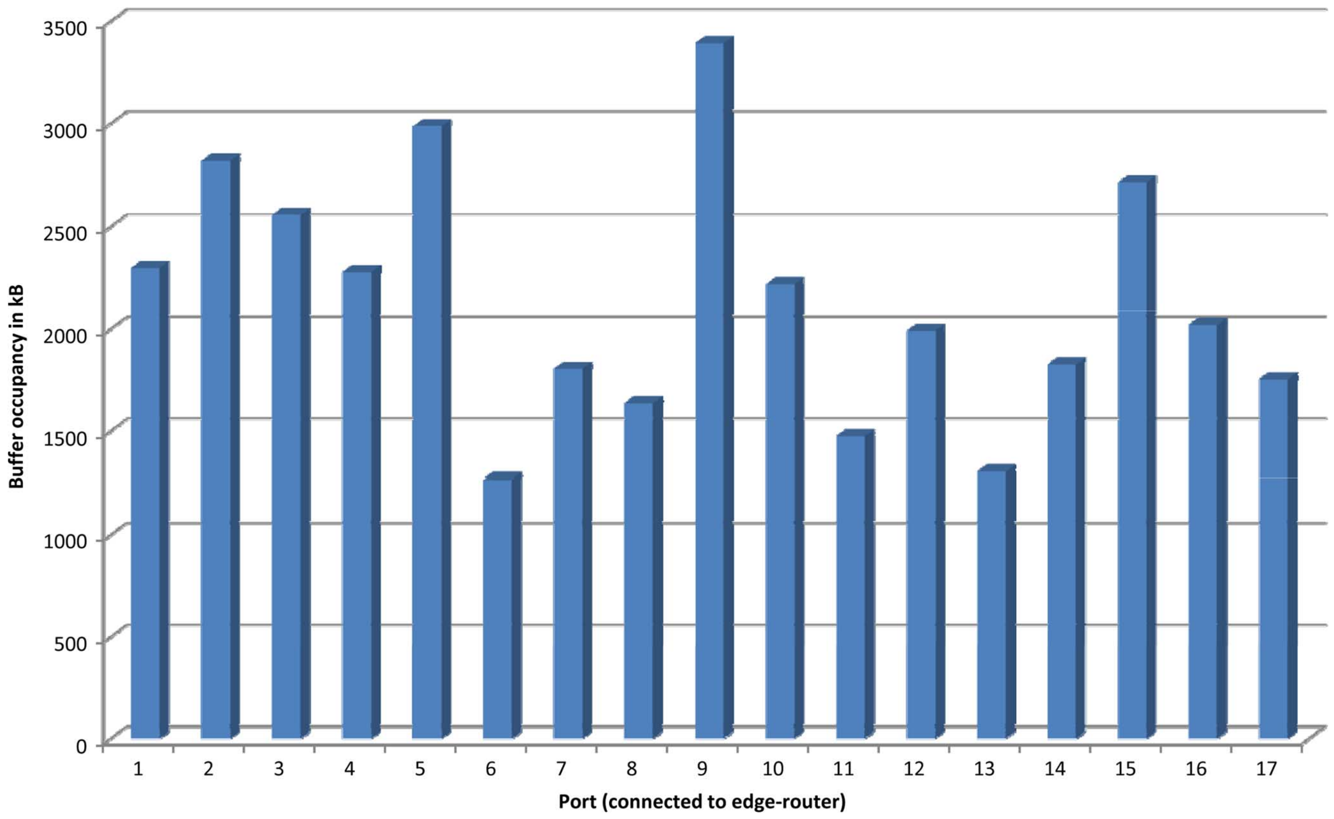

Fig. 2. Output-buffer usage from one port-set of 48 ports in the 18 ports connected to the farmnodes. In this test 350 kHz of events where sent from 270 read-out boards.

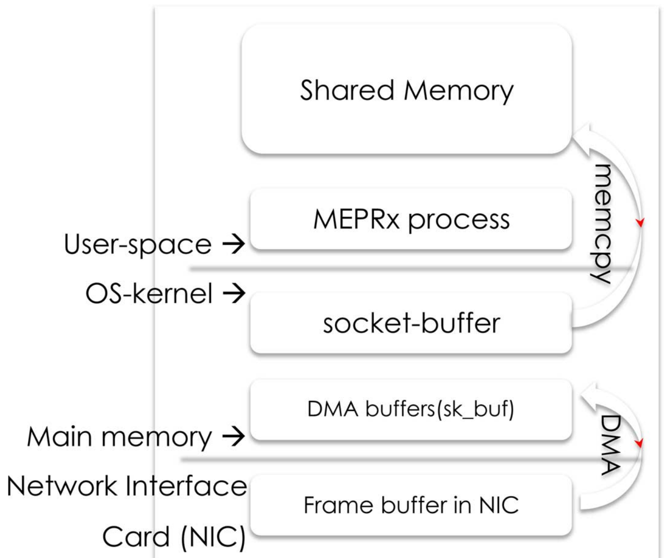

Fig. 3. Data transfer in a farmnode from the frame reception in the NIC to the shared memory used by the trigger application.

memory. These numbers are valid for a 64-bit implementation using gcc version 4.3.

B. CPU Load

The second resource which deserves interest and close monitoring is CPU usage. The read-out boards are driven by FPGAs, so there is no CPU in the usual sense. The actual data-formatting for sending over the Ethernet links is very simple and requires little FPGA resources, in particular since the Media Access Controller (MAC) is a separate external ASIC.

In the network devices the CPU load is independent of the traffic flowing through, however the monitoring of the traffic can cause heavy CPU load, which in turn can lead to performance problems. An overloaded CPU can lead to packet-loss both in servers and network devices. In servers this is caused by the CPU being too late or too slow to handle interrupts from the network adapter. In network switches the CPU can fail to update forwarding tables, thus causing packet loss. In this way fine-grained, high-rate monitoring can have undesired side-effects on switches.

Finally in the servers the CPU usage rises approximately linearly with the event rate as shown in Fig. 4. This is because the main contributions to CPU load are the memory copy operations and the bookkeeping, both of which scale almost linearly with the number of events per unit of time. As can be seen from the figure, for the servers currently in production, there is no problem from the CPU-load.

V. Problems and Solutions

Apart from trivial programming errors in the code of the event-builder or other parts of the readout system and simple configuration issues in the network devices (such as maximum transmission unit not uniformly set to 9000 octets), the only real problems are caused by packet-loss.

Despite initial doubts using an infrastructure all based on unshielded twisted pair (UTP) category 6 (TIA Cat6) cabling has

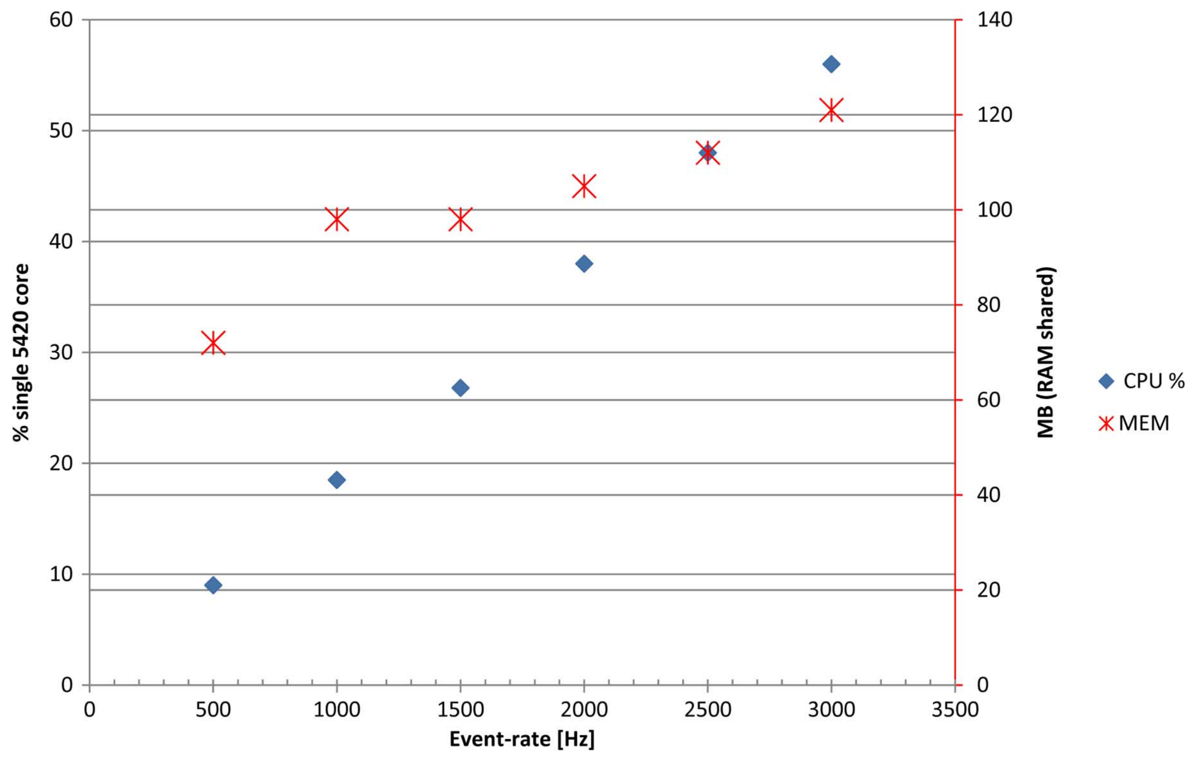

Fig. 4. MEPRx CPU load as a function of input rate.

not caused any problems. On more than 2500 links the only hardware problems observed were due to physical damage on the rear of the patch-panels. There are no checksum errors, jabbers [4] or other indications of problems on the link-layer. In the following we will present the many sources of packet loss and how they can be overcome.

A. Network Issues

1) MTU Issues: Choosing an as large as possible Maximum Transmission Unit (MTU) is very important for good performance in many layers of the data-flow, in particular in the receiving farmnode, because it reduces the number of frames which need to be handled. It also reduces protocol over-heads and thus saves band-width. Even though a MTU above 1500 bytes is not in any of the Ethernet standards, all enterprise class devices support at least 9000 bytes. We had an unpleasant surprise when we changed our protocol from being based on Layer-2 addresses (i.e., pure switching) to static routing (i.e., packet forwarding based on IP addresses instead of MAC addresses). Packet drop under load was observed but only for packets longer than 1500 bytes. In switching mode the same device had worked perfectly at any frame-size. For this limitation no work-around could be found and this device, which had passed all tests up to then, had to be discarded for the final system.

2) Backplane Usage in Switches: Large switches are usually of the packet switching type. They are not fully connected cross-bars. Consequently a scheduling algorithm determines which groups of ports can exchange data at any give time. The default setting for this period, sometimes called an "epoch", is optimised for random traffic, such as found in a full-mesh test scenario. In our device the default setting is 10.4 μs.

For event-building traffic however all groups of ports have data for a single specific egress port to which the farmnode is connected, which is the target for this event. Since buffering is done at the egress, and there is a lot of ingress traffic,[11] it is important that the scheduler grants access more frequently than in the default configuration so that the port-group can off-load their packets. A higher frequency for the scheduler will cause sub-optimal back-plane usage. But the total back-plane capacity is an order of magnitude above our needs. We chose a delay of approximately a third of the default, 3.2 μs.

3) Buffer Distribution in Switches: Cheaper switches, such as our edge routers implement almost all functionality in a single ASIC. This usually comes with some limitations in the way the buffer memory can be re-organized. For instance, any device destined for professional, data-center use, will support several priorities for traffic. These will be implemented in hardware by queues. In most edge routers we have encountered, it is not possible to attribute all memory to a single queue, which in the case of event-building traffic means that buffer-memory is actually lost. The edge routers in our system, which are the best devices we have found in our tests, have a comparatively large egress buffer of 512 kB per port. It is possible to attribute 90% of these 512 kB to a single queue. This is problematic when the total size of all MEPs destined to a specific egress port is larger than this limit. Since there are significant tails in the distribution of event-sizes, before corrective measures were introduced, we lost a MEP about once in 10 seconds, where the actual number

[11]Typically there will be between 300 and 500 frames with a total size of 400 to 600 kB for a single egress port.

depends on the event-size, which gets larger when beam-conditions create more noise hits, and trigger-rate. While it can be argued that this is a rather small loss, this loss introduces a bias in the event sample received by the high level trigger, which is highly undesirable.

4) Link-Aggregation: As described above, 50 edge routers connect the farmnodes to the core router. The edge routers use two link aggregation groups (LAGs) to connect to the core router. LAGs are defined by the IEEE 802.3ad standard, which, however, does not define how individual links within a LAG are used, i.e., according to which criterion which link in a LAG is chosen. It does however require that the temporal order of packets be preserved. Normally LAGs are used for performance improvement as well as for increased redundancy. Unfortunately in our setup this has the consequence that while a farmnode is connected with a single Gigabit link to the edge router, the edge router receives packets destined to this farmnode over six links in parallel. This is six-to-one overcommitment, which above a certain event-size, i.e., a certain number of read-out boards, causes packet-drops in the edge routers, whose output memory is limited to about 450 kB. In our case this could be overcome by using special link-selection algorithm. In this algorithm the link is chosen based on an arbitrary field in the IP header. We have programmed the read-out boards such that the least significant half-word of the event-ID, which is a strictly monotonically increasing number is used in the IP header of outgoing MEP datagrams. Since this number is always the same for MEP datagrams belonging to the same set and hence destined to the same farmnode, this will result in only one link out of the LAG being used for this specific event-number. In this way the six-to-one overcommitment is reduced to one-to-one scenario and packet loss is avoided. It should be noted that this specific LAG algorithm is quite unique.

5) Layer-2 Clock: To our surprise even in the one-to-one scenario described above, we still observed losses in the edge routers, albeit at lower rate. Tests with a traffic-generator confirmed that when a very long train of packets exactly back-to-back, i.e., with minimal inter-frame gap, is sent from one input port to only one output port, some of the aggregation routers (not all!) show packet drop. After long debugging we could trace this down to an interesting feature of the IEEE 802.3ab standard. In clause 40.6.1.2.6 the transmit clock frequency is defined to be 125 MHz $\pm 0.01\%$. The receiver, which recovers the clock from the transmitter, is required in 40.6.1.3.2 to have tolerance in accepting 125 MHz $\pm 0.01\%$. We could show that the clock used to transmit by our main router is 125.007, that is within the tolerance of the Ethernet standard. The edge router receives the packets without problem, however when transmitting itself it uses a 125 MHz clock. During a long train of packets, this can lead to a loss of a byte and consequently a packet. We solved this by increasing the inter-frame gap used by the main router.

B. Losses in the Receiver Farmnode—Kernel and Driver Parameters

It has been mentioned that kernel parameters such as the IP fragment re-assembly time need to be set to large values to cope with bursty traffic. This tuning has to be re-visited for every major OS release otherwise packet loss may happen. In the device driver the IRQ coalescence has to be set such that a maximum of packets is transmitted in one go, since latency is not important in our application, minimising the interrupt rate helps in the overall performance. This needs to counter-balanced by generous buffer parameters. Most of the parameters have to be set at least one order of magnitude larger than the defaults.

We have observed that even minor releases of software updates can have undesired side-effects. After an upgrade of the driver for some of our network cards, we observed frequent packet-loss. It turned out that this was due to a bug in the firmware of the NIC. This bug can cause the NIC to lock up and loose almost all packets. It is triggered by a long train of big packets which arrive with little or no separation. This happens of course frequently in the DAQ network. The earlier version of the driver used a work-around, which was apparently lost in the version upgrade. Currently we are using a RHEL5.4 kernel (2.6.18 series) in Ethernet drivers from Broadcom (tg3) and Intel (e1000). Many important driver parameters are of course hardware specific. The tuning of most of these parameters relies on a good understanding of the hard- and software and intuition, because very little detailed monitoring (for example the number of currently used DMA descriptors) is available.

VI. Conclusion

The LHCb event-builder embodies an almost ideal push-architecture. It is implemented as a large Gigabit Ethernet switching network where the sources are custom-build read-out boards and the receivers are standard PC-servers. The network is a two-stage one, with one large core router attached directly to the read-out boards and edge routers connecting to the PCs. An existing synchronous system, the Timing and Fast Control system, is used to distribute destination addresses and implements load-balancing using a simple credit-based mechanism. The MEP protocol implements no re-transmission, nor does it know dedicated event-builder units or an intermediate stage between the network and the read-out boards. The system is thus very lean, but very dependent on the devices downstream of the readout-board to be able to support the extremely bursty traffic pattern created by the synchronicity of the sending. When any out of 308 datagrams belonging to one event is lost, it will lead to the discarding of the entire event.

Stress tests using traffic generators up to 500 kHz of event-rate have shown that it is indeed feasible to build such a system with existing hardware. Numerous sources of packet-loss had to be overcome. The event-loss rate is now very low $(<10^{-8})$. The main strengths of the system are its extremely simple protocol, which is easy to implement in hardware, and the economy in terms of components, both in number of different kinds of hardware. There are essentially only five different device-types. Secondly the system is economical in the overall number of devices since there is no protocol adaptation layer, as the read-out boards already send their data in the final DAQ format and the final event-building is done on each receiver PC.

The main weakness is the dependence on the performance of the core router, which has to absorb the full inrush and a 300 to 1 overcommitment. Very few, if not only a single, currently available devices can achieve this. They are very expensive and not as perfect as originally hoped for. Experience suggests using two devices in parallel, even though just looking at the bandwidth available a single router is by far sufficient. For the near future a second core router will be a most reasonable consolidation. Even with a single device, the presented system is a reliable data acquisition for a detector producing 10^6 events of 35 kB per second.

The question, wether this system can meet the requirements to operate the data acquisition at 40 MHz for the upgraded LHCb experiment, is the subject of an exciting R&D programme.

References

[1] A. A. Alves *et al.*, "The LHCb detector at the LHC," *JINST*, vol. 3, p. S08005, 2008.

[2] F. Golshani and F. M. Groom, *The ATM Handbook,*1st ed. 2000, vol. 8, Nat. Commun. Forum.

[3] Open Specifications and Documentation [Online]. Available: http://www.myri.com/open-specs/ Myricom

[4] *Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,*, LAN/MAN Standards Committee, 2008, IEEE Computer Society, Ed..

[5] G. Haefeli *et al.*, "The LHCb DAQ interface board TELL1," *Nucl. Instrum. Meth. A*, vol. 560, pp. 494–502, 2006.

[6] B. Gaidioz, A. Barczyk, N. Neufeld, and B. Jost, "Evaluation of subfarm controllers candidates with an implementation of LHCb event-building," CERN-LHCB-2005-087.

[7] P. R. Barbosa-Marinho *et al.*, LHCb Online System Technical Design Report: Data Acquisition and Experiment Control, CERN-LHCC-2001-040, 2001.

[8] The Internet Protocol, Sep. 1981 [Online]. Available: http://tools.ietf.org/html/rfc791, RFC791

[9] X.200: Data Networks and Open System Communications Open Systems Interconnection—Model and Notation, vol. 7, 1994 [Online]. Available: http://www.itu.int/rec/T-REC-X.200-199407-I/en, ITU-T

[10] B. G. Taylor, "TTC distribution for LHC detectors," *IEEE Trans. Nucl. Sci.*, vol. 45, no. 3, pp. 821–828, Jun. 1998.

[11] M. Frank *et al.*, "The LHCb high level trigger infrastructure," in *J. Phys. Conf. Ser.*, 2008, vol. 119, p. 022023.