An implementation of a reliable message broadcast for the CMS event builder system

I. Suzuki[1], *M. Litmaath*[1], *V. O'Dell*[1], *S. Pavlon*[2], *K. Sumorok*[2]

[1] Fermi National Accelerator Laboratory, USA
[2] Massachusetts Institute of Technology, USA

Abstract

The CMS DAQ system utilizes a dedicated network to distribute commands to all the DAQ units responsible for reading out the detector Front-Ends. This network must have low latency (∼ms) since it must signal the presence of a new event (at a maximum rate of 100 kHz) well before the Front-Ends run out of their limited buffering capability. A reliable broadcast functionality is necessary for this. Although many implementations of reliable broadcast/multicast protocols exist and are available, the requirements of total reliability and short latency have kept us from using any of these. A NACK-based reliable broadcast protocol with Forward-Error Correction was implemented on a system of PCs connected via Fast-Ethernet. The implementation was tested in various system configurations and proved its technical feasibility.

Keywords: CMS, Event Manager, reliable broadcast, UDP

1 Introduction

The design of the trigger and data acquisition system for the CMS experiment[1] is diagrammed in Fig. 1. The event manager (EVM)[2] controls and monitors all event traffic in the bottom half of the data acquisition system, which is a large network switch fabric surrounded by hundreds of readout units (RUs) and builder units. One of the functions of the EVM is to distribute the first level (L1) trigger decision to the RUs reliably via the readout control network (RCN). The L1 trigger information is utilized by RUs to read buffered data from the front-end modules, that are named detector dependent units (DDUs).

The RCN should support reliable broadcast or multicast. The L1 trigger is expected to accept events at a rate of up to $100\,\mathrm{kHz}$. Assuming the size of the information sent via the RCN to be three 32-bit words per L1 accept, the required bandwidth is $96\,\mathrm{bits} \times 100\,\mathrm{kHz} \approx 10\,\mathrm{Mbps}$. The required latency for a transaction is $10\,\mu\mathrm{s}$ when the EVM sends packets on every L1 accept. It is, however, difficult to broadcast a message reliably within such a short time period using currently available commercial networks. Therefore, many (O(100)) messages are packed and sent out as one packet. The buffer depth in the DDUs gives the actual limit for the latency. The current CMS design expects the depth to be about a thousand events. Then, to keep the buffers only halfway filled, less than $5\,\mathrm{ms}$ latency is required.

The design of the final EVM and RCN system has not been determined yet. Developments of prototype systems and extensive performance studies are necessary to finalize the system.

2 Protocol on the RCN

Many reliable broadcast/multicast protocols have been proposed or implemented[3]. Most of them aim to serve multimedia (audio/visual) data streams, which do not need total reliability, *i.e.* some

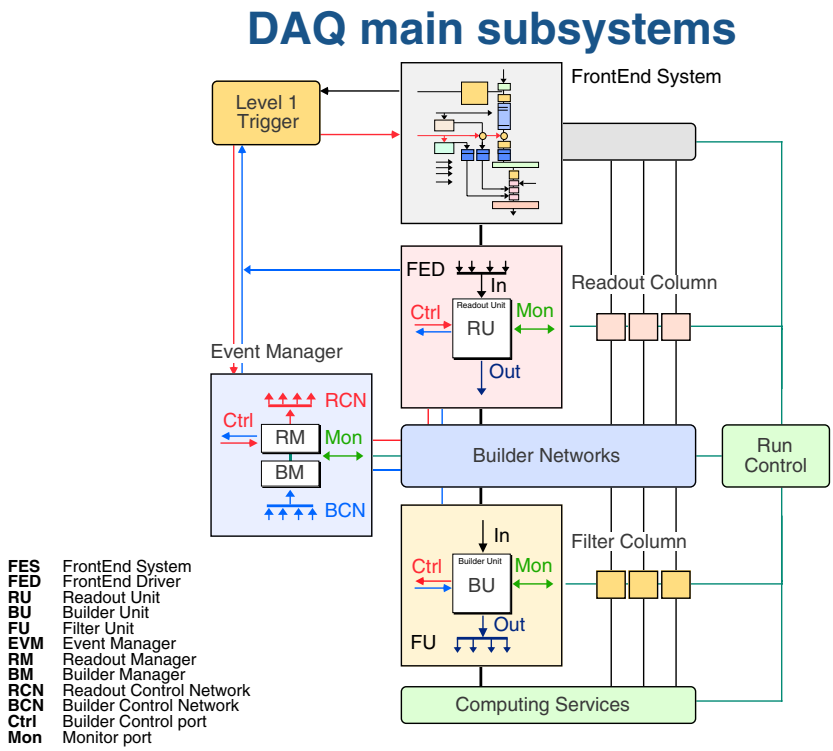

Figure 1: CMS trigger and data acquisition system

information may be lost. Some of them are totally reliable, but are not fast enough. The protocol used on the RCN must be totally reliable and have a limited latency of several milliseconds at most. We propose a preliminary protocol specialized for the RCN. It is a combination of popular techniques[4] and a simple design.

Key features of the protocol are;

- NACK based,
- Receiver side packet loss detection,
- Redundant traffic to reduce packet repairs, and
- ACKs from a few receivers as a pacemaker.

ACK(Acknowledge)-based mechanisms, in which each receiver sends back an ACK packet in principle for every data packet, are not applicable to the RCN. A flat network would be flooded with ACK packets and a tree-based ACK mechanism introduces a significant latency. In a NACK (Negative ACK)-based mechanism, a receiver sends back a repair request only when it detects packet losses or unrecoverable corruptions.

Data corruptions inside a packet are detected utilizing CRC (cyclic redundancy check-sum) words included in each packet. Packet losses are detected as jumps between the sequential numbers of adjacent packets. The sequential number is managed by the EVM in our case.

With these mechanisms the data can already be transmitted reliably. The process of repairing packets, however, is expensive in terms of latency. Therefore a Forward-Error Correction (FEC) technique is introduced to keep the packet retransmissions as rare as possible. In this scheme the EVM sends redundant data and any receiver can reconstruct the original data allowing

for packet losses up to the redundancy limit. A simple example is to add one extra packet, that is the XOR of the original packets, for every three data packets. In that case the receiver misses the whole packet sequence only when there are two or more packets lost from these four packets. Any fraction of redundancy is possible regarding the network reliability.

The last point is congestion control. There is no means to detect buffer overflow on the receiver side, because hardware flow control is not possible (or not practical) in broadcasts. Keeping the transfer rate below a level acceptable for the RUs is necessary to avoid buffer overflows and resulting NACK floods. A possible method is that the EVM waits for an ACK packet from only one or a few RUs and sends out the next packet after receiving it. Using this method the EVM applies an optimized delay between sends. Having the EVM sleep through a kernel system call cannot be used to control the delay with adequate precision. Fig. 2 illustrates the protocol described above.

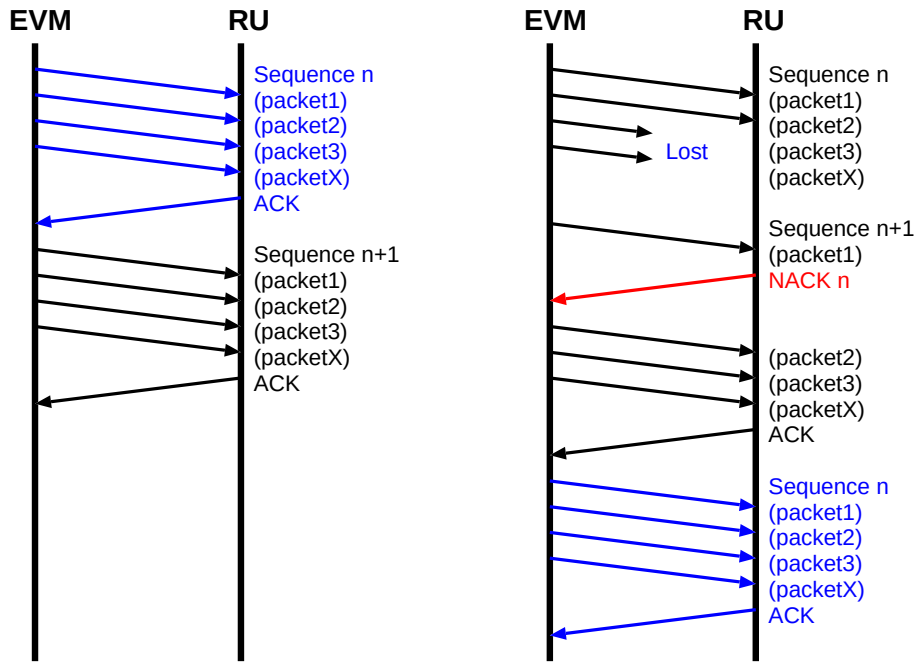

Figure 2: Sequence diagram of the RCN transactions.; Left: Normal handshake with ACK packets.; Right: Packet repair. Packet loss detected by the first packet in the next sequence. A RU requests the missing sequence by NACK and repair packets are sent back.

3 Implementation and benchmark tests

Each mechanism in the protocol described in the previous section has its advantages and disadvantages on performance and stability. We made test programs on a system of PCs connected via Fast-Ethernet to evaluate the protocol. The PCs were running the Linux OS. UDP/IP was used as a base protocol and the reliable broadcast protocol was implemented on top of it. This test setup required from us a minimal cost and effort while it maintained a reasonably realistic environment.

Two redundant traffic mechanisms were tried. One was the 3 + 1(XOR) FEC scheme and the other was a simple packet duplication. For a random packet loss probability of μ, both mechanisms have a probability of $\mathcal{O}(\mu^2)$ to lose a packet sequence. The fractions of the redundancy traffic are 25% and 50% for 3 + 1(XOR) FEC and duplication, respectively.

Bandwidth, time spent for the packet repairs and timing jitter of the packet receipts were

measured under various conditions. The target values are $10\,\mathrm{Mbps}$ for the bandwidth and less than $1\,\mathrm{ms}$ of time for packet repairs and reception jitter. The packet length should be small enough compared with the Front-End buffer depth of 1000 events. Zero packet loss, 1×10^{-3} and 1×10^{-6} packet loss rates for 500 RU nodes were emulated. The CRC was not implemented at this time.

In general, the results of the tests performed using this simple hardware/software configuration already satisfied the requirements for the RCN. There was, however, a problem in the very long tail of the packet reception jitter. Detailed results were presented in the talk. Effects of packet redundancy and rate regulation by ACK packets were also manifested.

4 Summary and future

The implementation of the protocol is still preliminary. There are several problems to be solved, in particular the long tail of the packet reception jitter. The bandwidth is enough for the L1 trigger distribution, but may be too narrow when we include trigger throttling information from the RUs to the EVM, which in turn informs the L1 trigger. In any case the generally positive benchmark results encourage further development of this reliable broadcast protocol.

Possible future options for improvements will be an implementation on a real-time OS like RT-Linux or VxWorks, a lower-overhead base protocol for Ethernet (VIA, or custom) and variations on the FEC scheme. For the hardware we have already started studies of an IEEE1394 (Firewire) network. This may give us significantly lower latency and enough room for extra traffic.

Acknowledgements

We would like to thank Mark Bowden for suggestions on the broadcast protocol, Don Holmgren and Ron Rechenmacher for their Linux kernel level message logging system, TRACE, used in the benchmark tests.

References

1 The Compact Muon Solenoid, Computing Technical Proposal, CERN/LHCC 96-45, 1996
2 E. Barsotti, M. Bowden, R. Kwarciany, M. Litmaath, V. O'Dell, Preliminary Specifications for the CMS Event Manager, CMS internal note, draft
3 K. Obraczka, Multicast Transport Mechanism: A Survey and Taxonomy, IEEE Communications Magazine, 1998
4 M. Handley, B. Whetten, TR. Kermode, S. Floyd, L. Vicisano, The Reliable Multicast Design Space for Bulk Data Transfer, IETF internet draft, 1999