Dataflow Monitoring in LHCb

D Svantesson¹,²,³, R Schwenmer¹, G Liu¹ and N Neufeld¹
¹ CERN PH-Department, Geneva, Switzerland
² Chalmers University of Technology, Göteborg, Sweden
E-mail: niko.neufeld@cern.ch

Abstract. The LHCb data-flow starts from the collection of event-fragments from more than 300 read-out boards at a rate of 1 MHz. These data are moved through a large switching network consisting of more than 50 routers to an event-filter farm of up to 1500 servers. Accepted events are sent through a dedicated network to storage collection nodes which concatenate accepted events in to files and transfer them to mass-storage. At nominal conditions more than 30 million packets enter and leave the network every second. Precise monitoring of this data-flow down to the single packet counter is essential to trace rare but systematic sources of data-loss. We have developed a comprehensive monitoring framework allowing to verify the data-flow at every level using a variety of standard tools and protocols such as sFlow, SNMP and custom software based on the LHCb Experiment Control System frame-work. This paper starts from an analysis of the data-flow and the involved hardware and software layers. From this analysis it derives the architecture and finally presents the implementation of this monitoring system.

1. Introduction
The LHCb experiment [1] at CERN searches for tiny differences in the decay of B and anti-B hadrons, which are produced in proton-proton collisions in the Large Hadron Collider (LHC). The complexity of the events requires that a large amount of events, a million every second, is reconstructed and filtered. Because of this almost 60 Gigabyte of data need to be moved every second. This is achieved by a large local area network (LAN) implemented using Gigabit Ethernet. This network consists of more than two thousand active devices with even more Gigabit Ethernet interfaces. The reliable performance of all these interfaces is crucial for the good functioning of the data acquisition. Monitoring of all these high-speed interfaces is therefore very important but at the same time not easy, because many packets travel in the networks and traverse many different layers. In the following the data-flow with all its layers and the essential features of the protocol used for transporting the data is explained first. Then the monitoring framework for the data-flow is described in detail. The paper concludes looking at further possible developments.

2. The LHCb Dataflow
The LHCb data acquisition system is realized as a Gigabit Ethernet local area network with 307 data-sources, one core-router, 50 fan-out switches and almost 1400 servers. In total there are almost 3000 switch-ports. Data are pushed top-down using an unreliable datagram protocol (MEP). More than 300 TELL1/UKL1 readout-boards receive the data from the detector front-
ends. Data are pre-processed (zero-suppression, encoded, noise-removal, compression) and sent to the DAQ network. As usual in an event-building network, data-grams from all readout-boards must reach the same destination in order for the complete event to be assembled. In LHCb the destination is one out of 1500 server PCs. On the way to the server PC the datagram (often called MEP-packet) will first be forwarded by the main router to one of 50 distribution switches, which are connected directly to the PCs. In network language there are two “hops” between source and destination. The network setup is schematically shown in Figure 1.

The maximum size for a MEP is 64 kB. This means, that when it is sent using Ethernet, it can be necessary to split the payload over mutliple frames. In monitoring this fact can lead

Figure 1. The LHCb dataflow from the readout boards to the event filter farm. Data flows the readout-boards shown on the top to farm-servers shown at bottom.
to complications because the higher level applications and the users normally only care about the number of MEPs sent and received. Network devices such as routers and switches normally keep statistics of Ethernet frames as required by the Ethernet standard.

3. Monitoring

Monitoring the data-flow means first of all collecting all available information at all possible data-sources in the system. Often the amount of transmitted information can be reduced by summing up counters at various levels. Finally an interpretation or analysis of the monitored counters needs to be given. The challenges for any such monitoring system are the following:

- the number of counters (see Table 1 The total number of counters in the system (including all error counters) is well over $10^6$
- the heterogeneity of the source data
- the wide-spread lack of any counters other than those mandated by IEEE 802.3 (Ethernet)

The following key technical solutions have been used and will be explained more fully in the following:

- a "divide and conquer" approach to sum counters at multiple levels
- republish all information in an open, light-weight protocol, DIM [2] and display using LHCb’s standard SCADA system PVSS2
- use real-time access-control list functionality originally developed for fire-walls to create counters for custom protocols

| stage            | # counters | OSI-layer | protocol |
|------------------|------------|-----------|----------|
| readout-boards   | 1200       | 2 (Ethernet) | DIM      |
|                  | 300        | 4 (MEP)   | DIM      |
| core network     | 1280       | 2 (Ethernet) | SNMP    |
|                  | 350        | 4 (MEP)   | ACL      |
| edge network     | 2400       | 2 (Ethernet) | SNMP    |
| farm-node        | 1500       | 2 (Ethernet) | DIM      |
|                  | 1500       | 3 (IP)    | DIM      |
|                  | 450000     | 4 (MEP)   | DIM      |

3.1. Data sources and access methods

All devices involved in the dataflow produce counters. An overview is given in Table 1. In terms of the OSI model three network layers need to monitored: Ethernet (Layer 2), IP (Layer 3) and MEP (Layer 4). Every device supports the basic Ethernet counters as defined by the Ethernet standard [3]. These counters are implemented by the Ethernet hardware (the Media Access Controllers (MAC)). One problem with these counters is that they are only 32 bits wide, because of which they will wrap around very quickly when they count bytes or any other very common occurrence. Nevertheless these counters are very useful because they are absolutely reliable. On network devices such as routers and switches they can be reasonably efficiently retrieved using the Simple Network Monitoring Protocol (SNMP). On the readout boards and the farm-servers we export the counters using DIM.

---

4 On a Gigabit link it takes less than 35 seconds to transmit $2^{32}$ bytes.
The IP [4] standard does not define a standard set of counters. We are usually interested in how many packets are sent and/or how many have been received. The number of dropped packets is of particular interest in switches. Corrupted or truncated packets are another also of interest. On the readout boards we export the counters via DIM, they are counted by the FPGA driving the transmission. In the routers there are no specific counters, however we can use Access Control Lists (ACL) to count them. ACLs were actually invented for the implementation of firewalls. In all modern routers there is hardware support for ACLs. Since they are implemented directly by the ASICs moving the packets, the counting is very reliable and incurs no overhead. Unfortunately however while the actual counting using ACLs is for “free”, the acquisition of the values of the counters in the ACLs is not. They do not form part of the SNMP standard (because their configuration is highly customisable) so they need to be read via the control processor of the router. This can be done either via the management console (terminal) or in some cases via a usually proprietary API. Due to CISCOs enormous market dominance the command line interface to switches, while it is not standardized, looks and behaves at least quite similarly. We use the expect [5] library to query the counters and export them via DIM.

On the servers we use a very similar technique by using the iptables firewall mechanism for counting. The readout uses the standard Linux tools for reading information.

For MEP we use the same techniques as for IP. The only difference is that in the case of a MEP spanning \( n > 1 \) IP-packets, all these packets will be individually counted by the IP counters, while only one MEP will be counted, if the MEP could be successfully re-assembled by the IP layer. The ACLs in most of our routers are not flexible enough for this kind of operation requiring stateful information and deep packet processing so this can be done only approximately for MEPs fitting into a single IP packet. Comparing with the IP counters in the servers can be used to fix this difference. As for IP these counters are re-exported using DIM to the higher-level processing.

3.2. Implementation

The technicalities of how to access various counters have been briefly mentioned already in the preceding section. The difficulties are purely technical and mostly have to do with efficient string-processing. These collectors are written in C++ and python. All collectors export their data in a standardised form via DIM.

In order to reduce the number of counters for processing the collectors can be used also to sum counters from other collectors. This is extensively used in the event-filter-farm, where normally one does not care about the individual nodes. This divide-and-conquer approach is very efficient and the CPU load created by the collector processes is negligible. The overall software architecture is shown in Figure 2.

The result of all (summed-up) counters are fed into a dedicated process running in LHCb’s SCADA system PVSS2. The process does some final pre-processing of the data and fills data-structures (tables) made for easy graphical display. This process also selects the correct sub-sets of the detector and the event-filter-farm, which form a partition.

3.3. Results and performance

In LHCb the top-most layer, which presents information to the user is defined, because every high-level monitoring and GUIs are implemented using PVSS2. A screen-shot can be seen in Figure 3.

The information on the level of an individual server is of course still available in case a problem on an individual node is being debugged.
The software architecture of the monitoring system, showing the hierarchy of collectors used to reduce the number of individual counters.

The CPU load of the various processors is very modest. The counters cannot be updated very frequently however because the polling creates a load on the devices under monitoring. For some of the routers this could even impact forwarding performance! The polling rate is therefore set to 30 seconds. In any case in order to compare the counters to look for real matches and mis-matches the system needs to be halted as there is no way to read all this information synchronously.
4. Conclusions and further developments

Monitoring the data-flow in LHCb means counting network packets at Layers 2, 3 and 4 at every stage of a packet’s life in a network with more than 4000 links and almost 2000 active devices. Since an unreliable protocol is used, detection of systematic packet-losses is crucial, even when the losses are small. Commercial hardware usually provides only Layer-2 counters. Software counters are easy to implement, but their huge number (more than $10^6$) is still a problem. We have used ACLs to count Layer-3 and Layer-4 packets with 100% accuracy at a reasonably small overhead.

Among the open issues are the following:

- Currently the data-taking needs to be paused to obtain a consistent set of counters: a snapshot function would be very useful
- “Noise” on the network: Currently there are a few protocols active on the network (ARP, LLDP, SNMP) whose activity can falsify the packet counts. By explicitly counting these packets with ACLs one could correct for these
- In the current system hardware and software limitations do not allow using ACLs for counting in the edge-layer. This is subject to an upgrade of the edge layer (and will be a requirement for the next generation of devices)
- SNMP readout: the SNMP readout is very slow and moreover creates a high CPU load on the switches. Using a direct API (or the command-line) can potentially reduce the CPU load and provide faster read-out of the counters.
References
[1] Alves A A et al. (LHCb) 2008 JINST 3 S08005
[2] C Gaspar M Dönszelmann P C 2000 DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (Padova, Italy)
[3] LAN/MAN Standards Committee (ed) 2008 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications (IEEE Computer Society)
[4] 1981 The Internet Protocol RFC791, [Online]. available: http://tools.ietf.org/html/rfc791
[5] Harrison M (ed) 1997 Tcl/Tk Extensions (O’Reilly) chap Expect