Available on CMS information server **CMS IN 2005/000**

CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

1 April 2005

Simulation of the CMS Event Building System using the EVBSIM Package

M. Litmaath

Fermilab

K. Sumorok

MIT

Abstract

This document describes the functionality and results of the EVBSIM package used in the simulation of the CMS event building system.

DRAFT, version 0.5, November 1, 2001

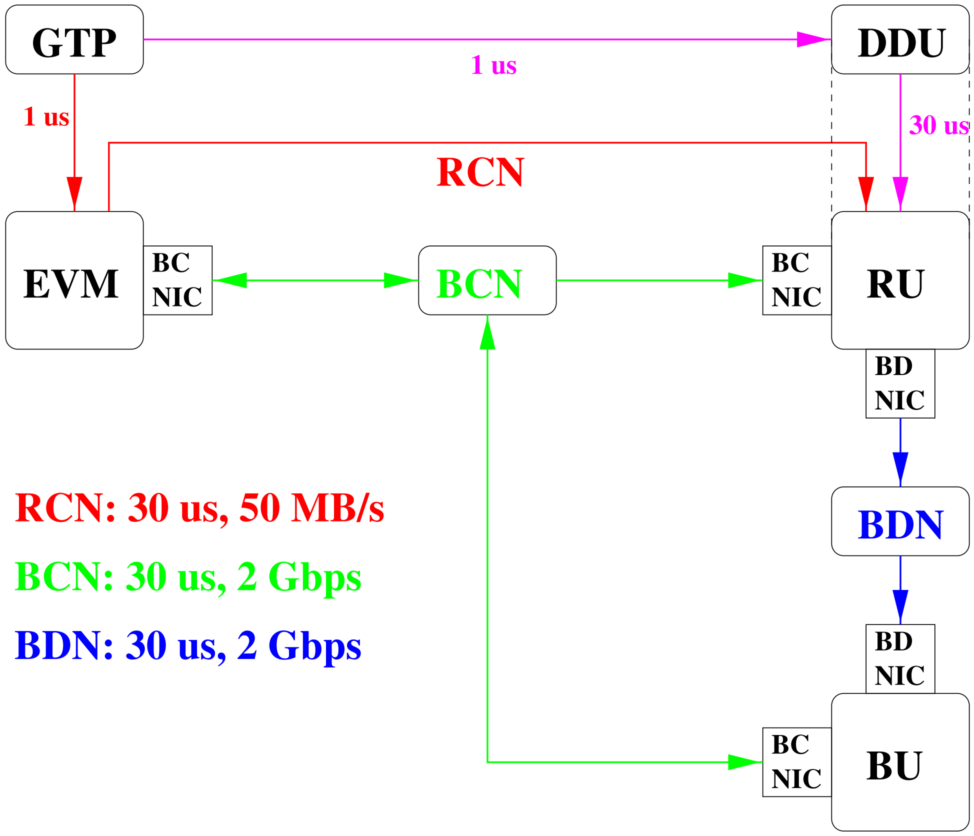

Figure 1: Players in the event building system and their interconnections with typical bandwidths and latencies.

1 Introduction

The goals of the EVBSIM simulation of the event building system are:

- comparison of alternative hardware technologies and software protocols at the scale of the final implementation for the experiment;
- verification of the ability of a proposed final setup to sustain the nominal event data rate, with some margin for fluctuations;
- verification of the simulation code itself through comparisons with results from test stands with various configurations.

2 The EVBSIM package

EVBSIM is a CMS event building simulation package written in C++ and based on the public domain CNCL class library developed by Aachen University, Germany, for the simulation of computers and networks. The EVBSIM package consists of modules implementing the basic functionalities of the GTP (L1), EVM, RU, BU, and of a perfect NxN crossbar switch that allows any two transfers to proceed in parallel when they have different sources and different destinations. The switch is connected to the RUs, BUs and EVM through NIC objects that take care of sequencing packets. Each NIC can do three things simultaneously:

- sending a packet to the switch
- receiving another packet from the switch

- copying yet another packet either to or from host memory via DMA over the PCI bus

The NIC code is generic, i.e. it does not precisely model e.g. a Myrinet NIC. The path to send a packet from one host through the switch to another host has three legs, each with its own latency pedestal (overhead) and bandwidth:

$$\Delta t(\mathrm{host} \rightarrow \mathrm{NIC}) = \Delta t_0(\mathrm{host} \rightarrow \mathrm{NIC}) + \mathrm{bytes}/\mathrm{bandwidth}(\mathrm{PCI}) \tag{1}$$

$$\Delta t(\mathrm{switch}) = \Delta t_0(\mathrm{switch}) + \mathrm{bytes}/\mathrm{bandwidth}(\mathrm{switch}) \tag{2}$$

$$\Delta t(\mathrm{NIC} \rightarrow \mathrm{host}) = \Delta t_0(\mathrm{NIC} \rightarrow \mathrm{host}) + \mathrm{bytes}/\mathrm{bandwidth}(\mathrm{PCI}) + \Delta t(\mathrm{delay}) \tag{3}$$

A bandwidth is equal to the maximum burst speed of the medium, e.g. 132 (528) MB/s for "min." ("max.") PCI and 160 (250) MB/s for Myrinet 1999 (2000). The three latency pedestals give the minimum durations for which the respective mediums are busy on any transaction (but note that the switch can handle transactions in parallel for different sources and different destinations). The latency pedestals are of the order of microseconds. The $\Delta t(\mathrm{delay})$ term represents an overlappable delay before the receiver process gets woken up to process the incoming message; the delay does not count in the reservation time of the PCI bus resource.

The EVBSIM switch is used to implement the physically distinct BCN and BDN. When these logical networks are to be implemented on a single physical network medium, the BCN also takes on the role of the BDN. The RCN is emulated by the EVM module. Each RU represents a single DDU. A picture of the module interconnections is shown in Fig. 1.

2.1 Comparison with the Fermilab test stand

The EVBSIM framework has been verified against the Fermilab XDAQ test stand, which uses a single Myrinet switch for BCN and BDN. Furthermore, the XDAQ code does not have the notion of an RCN, i.e. an RU generates its event fragments on the fly whenever it receives an event data request from a BU. This situation has been emulated in EVBSIM by using the BCN also as BDN and by giving the RCN a very small latency and very high bandwidth. Furthermore, the dependence on the GTP is effectively removed by operating it at a frequency much higher than the BUs can cope with, such that the EVM always has event IDs available for BU requests. The resulting configuration is depicted in Fig. 2.

The principal result of a test stand measurement is the asymptotic event data bandwidth observed by a BU (it should be the same for each of them). A large number of such measurements has been performed:

- the number of RUs can be 1, 2 or 4
- the number of BUs can be 1, 2 or 4, independent of the number of RUs
- the number of event ID requests *per* message from BU to EVM can be 1, 4, 16 or 64
- the number of active event ID request messages per BU can be 1, 4, 16 or 64, such that the total number of active event IDs per BU does not exceed 256

The resulting matrices of comparisons between EVBSIM and XDAQ are given in Table 1. The notation "NxM" indicates a setup with "N" RUs and "M" BUs. Each column pertains to a fixed number of event ID requests per message ("N"), while each row pertains to a fixed number of active event ID messages per BU ("R"). Each matrix entry shows the ratio of the BU event data bandwidths in MB/s obtained by EVBSIM and in the test stand. The given results are in

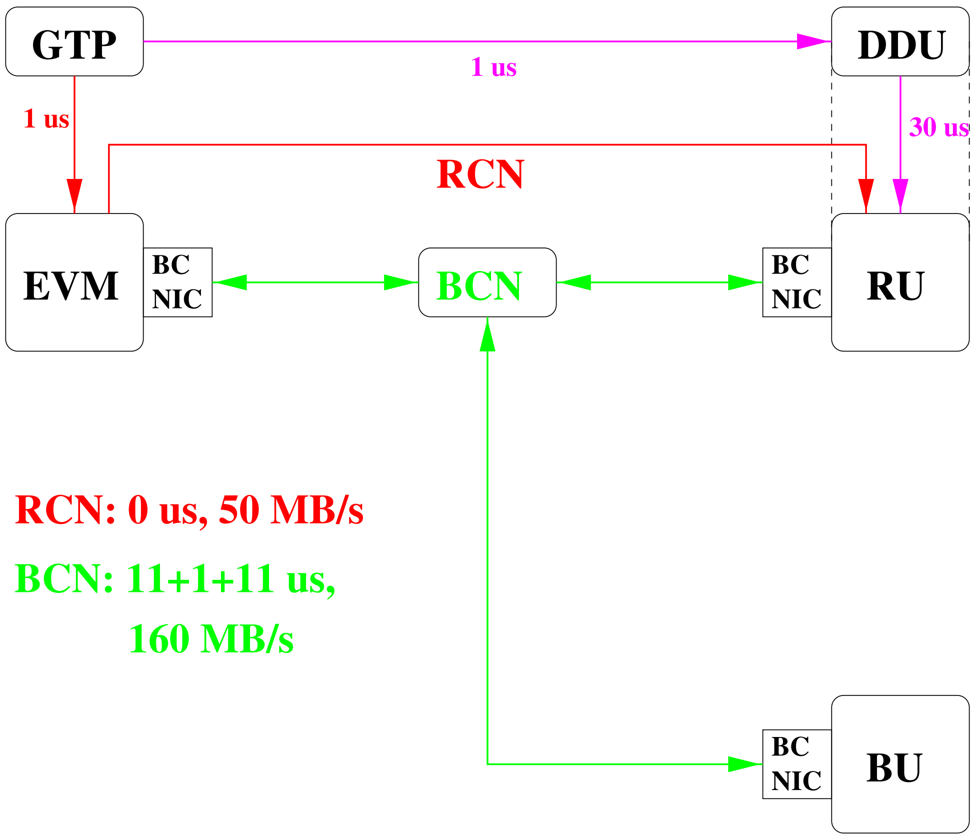

Figure 2: The EVBSIM modules and their interconnections with typical bandwidths and latencies for the Fermilab XDAQ test stand.

reasonable agreement, given that only the 3 latency pedestals were tuned for a *single* case: 1x1, N=1, R=64. The resulting values were $\Delta t_0(\mathrm{host} \rightarrow \mathrm{NIC}) = \Delta t_0(\mathrm{NIC} \rightarrow \mathrm{host}) = 11\mu$s and $\Delta t_0(\mathrm{switch}) = 1\mu$s.

There are a few matrix entries that show deviations from unity that are unexpected, given the values for the surrounding entries. These deviations appear to be due to anomalous behavior of the XDAQ measurements rather than the EVBSIM code: it is the XDAQ values that are inconsistent with trends that are present in most of the tables. It has been found both at CERN and at Fermilab that the XDAQ asymptotic bandwidths show occasional instabilities that are not understood.

The tests so-far have avoided internal collisions in the switch. For larger tests the simulation of the simple switch would overestimate the efficiency of the true (Myrinet) switch, but hopefully displaying a trend that can at least qualitatively be understood. One could then have some confidence in the simulation of the complete event building system, once the trend has been corrected for.

2.2 Traffic shaping through barrel-shifting

The throughput of the switch and hence of the event building system is improved by shaping the message traffic such that contention for any particular destination is reduced. The switch can be operated at close to 100% efficiency when the traffic is barrel-shifted: each sender on one side of the switch sends fixed-sized packets in round-robin mode to every receiver on the other side of the switch. It is crucial that each destination is sent a packet in every cycle, even if the host does not have anything for a particular destination: in that case a dummy (filler)

	R \ N	1	4	16	64
1x1:	1	8 / 10.7 = 0.7	23 / 27.2 = 0.8	43 / 43.8 = 1.0	56 / 52.0 = 1.1
	4	23 / 23.8 = 1.0	48 / 49.3 = 1.0	62 / 68.0 = 0.9	66 / 68.1 = 1.0
	16	25 / 25.2 = 1.0	48 / 49.3 = 1.0	61 / 64.8 = 0.9	
	64	25 / 25.4 = 1.0			
1x2:	1	8 / 10.6 = 0.8	22 / 26.4 = 0.8	35 / 33.7 = 1.0	35 / 35.0 = 1.0
	4	22 / 22.3 = 1.0	32 / 33.9 = 0.9	35 / 35.2 = 1.0	35 / 34.8 = 1.0
	16	25 / 24.9 = 1.0	32 / 34.2 = 0.9	35 / 35.2 = 1.0	
	64	25 / 25.1 = 1.0			
1x4:	1	8 / 9.3 = 0.9	16 / 16.6 = 1.0	17 / 17.4 = 1.0	18 / 17.1 = 1.1
	4	12 / 12.7 = 0.9	16 / 17.0 = 0.9	17 / 17.5 = 1.0	18 / 17.1 = 1.1
	16	12 / 12.6 = 1.0	16 / 17.0 = 0.9	17 / 17.5 = 1.0	
	64	12 / 12.6 = 1.0			
2x1:	1	14 / 18.8 = 0.7	34 / 41.5 = 0.8	54 / 60.0 = 0.9	63 / 67.1 = 0.9
	4	34 / 32.6 = 1.0	54 / 56.5 = 1.0	65 / 71.6 = 0.9	68 / 78.0 = 0.9
	16	34 / 33.4 = 1.0	55 / 56.5 = 1.0	65 / 71.9 = 0.9	
	64	34 / 33.6 = 1.0			
2x2:	1	14 / 18.7 = 0.7	34 / 40.5 = 0.8	54 / 53.3 = 1.0	63 / 55.0 = 1.1
	4	34 / 31.9 = 1.1	54 / 55.6 = 1.0	65 / 66.0 = 1.0	68 / 57.5 = 1.2
	16	34 / 33.0 = 1.0	55 / 55.7 = 1.0	65 / 66.3 = 1.0	
	64	34 / 33.1 = 1.0			
2x4:	1	14 / 17.4 = 0.8	32 / 32.2 = 1.0	35 / 34.9 = 1.0	34 / 34.2 = 1.0
	4	25 / 26.1 = 1.0	32 / 33.8 = 0.9	34 / 35.3 = 1.0	35 / 29.8 = 1.2
	16	25 / 26.0 = 1.0	32 / 33.9 = 0.9	34 / 35.2 = 1.0	
	64	25 / 25.9 = 1.0			
4x1:	1	24 / 28.9 = 0.8	47 / 53.1 = 0.9	62 / 67.6 = 0.9	68 / 71.9 = 0.9
	4	40 / 39.7 = 1.0	59 / 61.3 = 1.0	67 / 73.2 = 0.9	70 / 78.4 = 0.9
	16	40 / 40.0 = 1.0	59 / 61.3 = 1.0	67 / 73.1 = 0.9	
	64	40 / 40.0 = 1.0			
4x2:	1	23 / 28.7 = 0.8	47 / 45.1 = 1.0	62 / 57.2 = 1.1	68 / 49.9 = 1.4
	4	40 / 39.2 = 1.0	59 / 52.1 = 1.1	67 / 56.0 = 1.2	69 / 69.7 = 1.0
	16	40 / 39.3 = 1.0	59 / 52.1 = 1.1	67 / 50.2 = 1.3	
	64	40 / 39.6 = 1.0			
4x4:	1	23 / 27.9 = 0.8	45 / 42.4 = 1.1	61 / 57.1 = 1.1	67 / 57.2 = 1.2
	4	40 / 37.5 = 1.1	59 / 56.1 = 1.1	67 / 62.5 = 1.1	69 / 54.8 = 1.3
	16	40 / 39.1 = 1.0	59 / 56.4 = 1.0	66 / 62.4 = 1.1	
	64	40 / 39.1 = 1.0			

Table 1: Comparisons of BU event data bandwidths obtained by EVBSIM and in the Fermilab XDAQ test stand.

packet is generated. For a Myrinet switch such a scenario works very well, because a transfer will remain blocked until the path to the destination becomes available. When any two senders are contending for the same destination, the one will wait until the other has finished. From that moment on the one sender will always remain one step behind the other, as both move on to the next destination after finishing their current transfers. After a short startup period all N senders keep sending in parallel to N destinations. The barrel shifter operation in EVBSIM is demonstrated in Fig. 3 for a 64x64 system.

For switches that have internal buffering (e.g. Gbit Ethernet) the situation is more complicated, because packets can get dropped. For such configurations barrel-shifting can still be made to work, but care has to be taken that buffer overflows occur neither in the switch itself nor in the destination.

The vast majority of the traffic through the switch(es) of the event building system consists of event data fragments. It turns out that this traffic has to be shaped, otherwise the switch efficiency would be insufficient to keep up with the nominal 100 kHz trigger rate. Shaping the control message traffic appears to be necessary only for switches larger than those foreseen for the CMS event building system (see next section).

2.3 Scaling studies

Various studies have been performed to get an understanding of the scaling properties of the XDAQ protocol. From the perspective of the number of control messages, the most efficient operation would occur when the "indirect mode" is used: a BU informs the EVM of its number of free buffers ("resources") for event building, and the EVM will broadcast instructions to all RUs to send each event to a particular BU. Though this mode was originally being used in EVBSIM, the relevant code has long been superseded by code that corresponds to the "direct mode" currently favored and in use by XDAQ. It would not be hard to change EVBSIM such that the EVM either uses the RCN or a separate broadcast network for BCN messages, but we note that the use of broadcasts comes at a price: it may become harder to detect transmission errors and recover from them (e.g. a particular RU may have missed a message because its NIC buffers were full).

In the "direct mode" on the other hand, the EVM replies to BU requests by sending back event IDs; the BUs then send requests for event data directly to each of the RUs. In this mode the amount of BCN traffic scales with the product of the number of RUs and the number of BUs. If the BCN traffic is not shaped, its efficiency is expected to drop to unacceptable levels for large configurations. As a result the maximum sustained trigger rate would drop below the 100 kHz nominal value. This effect appears to be demonstrated in Fig. 4, which shows as a function of time the number of busy event IDs in the EVM for a 256x256 configuration in which each event is discarded in zero time once all its fragments have been received by the BU. Fig. 5 shows that the effect can be avoided by letting the BUs send their event ID requests and event data requests in barrel-shifter mode, such that contention among the BUs for the same destination (RU/EVM) is avoided.

The number of busy event IDs in steady state is seen to be about 8000. This number can be roughly understood from the following considerations. We start with the parameters that were used:

- $\Delta t_0(\mathrm{host} \rightarrow \mathrm{NIC}) = \Delta t_0(\mathrm{NIC} \rightarrow \mathrm{host}) = 1\mu\mathrm{s}$ and $\Delta t_0(\mathrm{switch}) = 0\mu\mathrm{s}$
- the bandwidths of PCI and the switch are 528 MB/s and 250 MB/s respectively

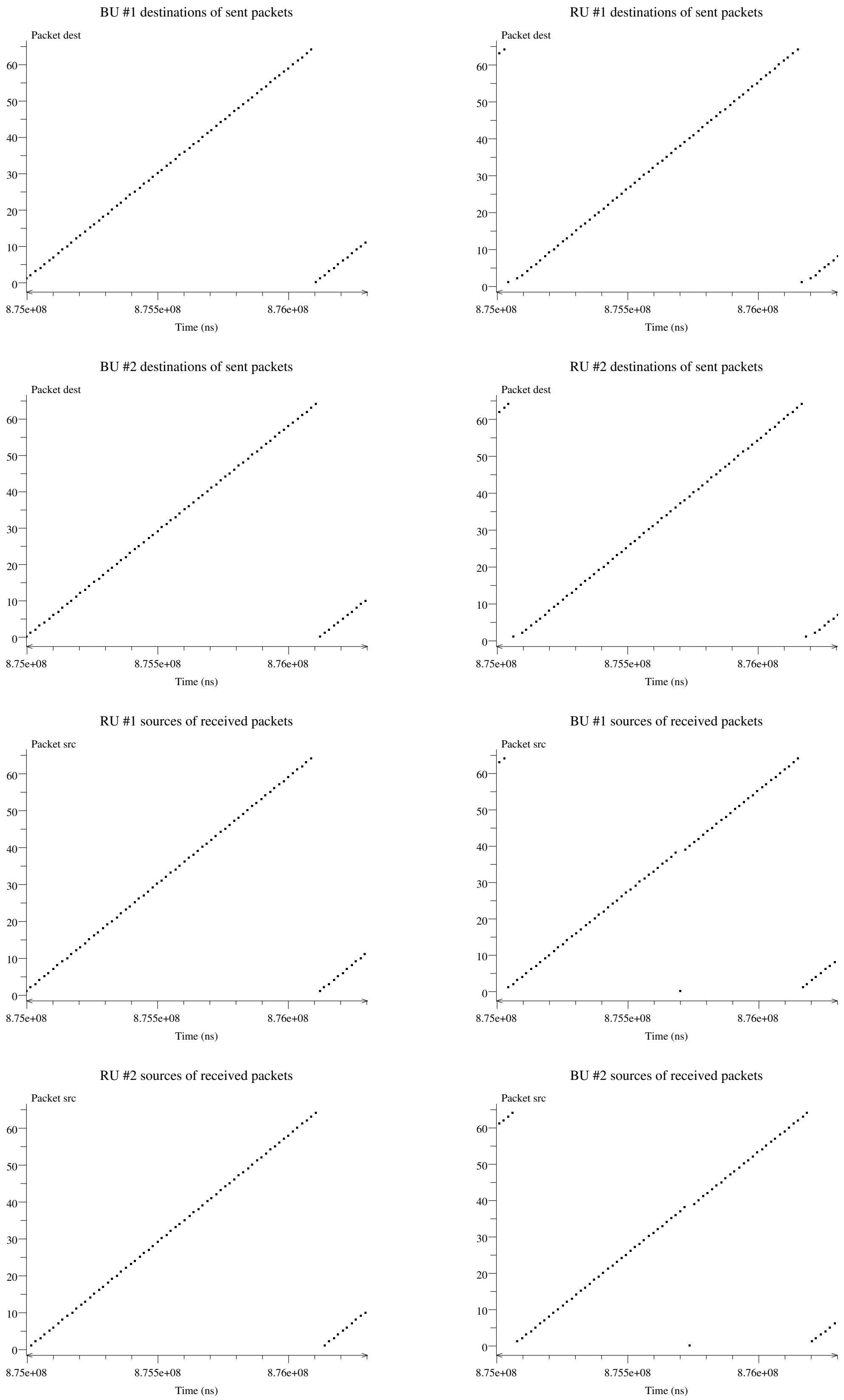

Figure 3: Demonstration of barrel shifting in a 64x64 system. Each player only starts its barrel shifter operation when it sends the first XDAQ message. That is why in each figure showing the sources of packets received in a BU, the EVM (source #0) is found to get its turn somewhere in the middle of the set of RUs.

- each basic XDAQ control message takes 64 bytes
- each XDAQ control packet consists of 224 bytes of I2O header and 2048 bytes of payload, making 2272 bytes per packet; the payload contains 2048/64 = 32 basic control messages (e.g. event IDs)
- each XDAQ event data packet consists of 224 bytes of I2O header and 2048 bytes of payload, making 2272 bytes per packet
- the minimal time to transfer a XDAQ packet from sender to receiver:

  $$\begin{aligned} \text{host} \rightarrow \text{NIC:} &\quad 1 + 2272/528 = 5.3\ \mu\text{s} \\ \text{Myrinet:} &\quad 0 + 2272/250 = 9.1\ \mu\text{s} \\ \text{NIC} \rightarrow \text{host:} &\quad 1 + 2272/528 = 5.3\ \mu\text{s} \end{aligned}$$

  since all packets are buffered both in the hosts and in the NICs, in steady state the transmission time is solely determined by Myrinet
- the trigger rate is 100 kHz

Then we consider the fundamental cycle in all of the BUs together:

releasing 32 ready event IDs to the EVM	1 x 2272/250 =	9 μs
requesting 32 new event IDs from the EVM	1 x 2272/250 =	9 μs
requesting 32 event fragments from the 256 RUs (all sent in parallel, barrel-shifted)	256 x 2272/250 =	2327 μs
receiving 32 event IDs from the EVM	1 x 2272/250 =	9 μs
receiving 32 event fragments from 256 RUs (all received in parallel, barrel-shifted)	32 x 256 x 2272/250 =	74449 μs
total time per cycle		76803 μs
number of new events accumulated in this time	100 kHz x 76803 μs=	7680
number of events handled in this time	32 x 256 =	8192
number of busy events observed in steady state		~8000

From these considerations we first conclude that the number of busy events observed in steady state is smaller than the number of events handled in a fundamental cycle. This can be explained as follows: by the time the last BU receives its share of event IDs, the first few BUs already have returned some of their event IDs. The number of busy events observed in steady state is higher than the amount of new events accumulated while the current cycle is being processed. This is due to the initial, temporary inefficiency of the barrel shifter operation: in the beginning most of the packets sent by any RU are empty, because only a few BUs have received event IDs and requested event fragments to be sent.

We further conclude that the system is able to handle the events faster than they are generated by the trigger. In fact, the trigger rate can roughly be increased to 8192/7680 x 100 = 107 kHz and the system should still keep up. It turns out that the maximum sustained rate in this setup is between 109 and 110 kHz, extremely close to the 110 kHz limit from considering Myrinet alone.

Fig. 6 shows the number of busy event IDs in the RUs as a function of time. An event ID is considered busy in the RUs from the moment it is received via the RCN until the last of its fragments has been copied into the Myrinet NIC. The remarkable peaks in the RU occupancy are taken to be due to the initial temporary inefficiency of the barrel shifter operation. In steady state the number of busy event IDs is about 500.

Fig. 7 shows the total number of queued event packets in the RU NICs as a function of time. In steady state every RU has about 4000 packets waiting to be sent. In reality a NIC would not have sufficient on-board memory to accommodate so many packets: the excess would be waiting in the RU host memory instead.

Fig. 8 shows the number of busy event IDs in the BUs as a function of time. An event ID is considered busy in a BU from the moment the last of its fragments has been received until it is released to the EVM. In steady state the number of busy event IDs is about 3500.

Now we look at a 64x64 setup with more realistic values for processing times. Each BU controls a subfarm of 8 nodes that have to deal with 12.5 kHz being 1/8 of the nominal trigger rate. Each event has a fixed size of 64 x 16384 bytes. Each node then gets about 24 Hz on average and should spend no more than about 40 ms per event on average. With the control message overhead it turns out that 39 ms per event on average is the maximum that can be sustained. Figs. 9, 10 and 11 show the number of busy event IDs in the EVM for various Gaussian processing time distributions with the indicated peak values and widths. Since a processing time cannot be negative, the negative tails of the distributions are cut off, thus shifting the mean values slightly upward.

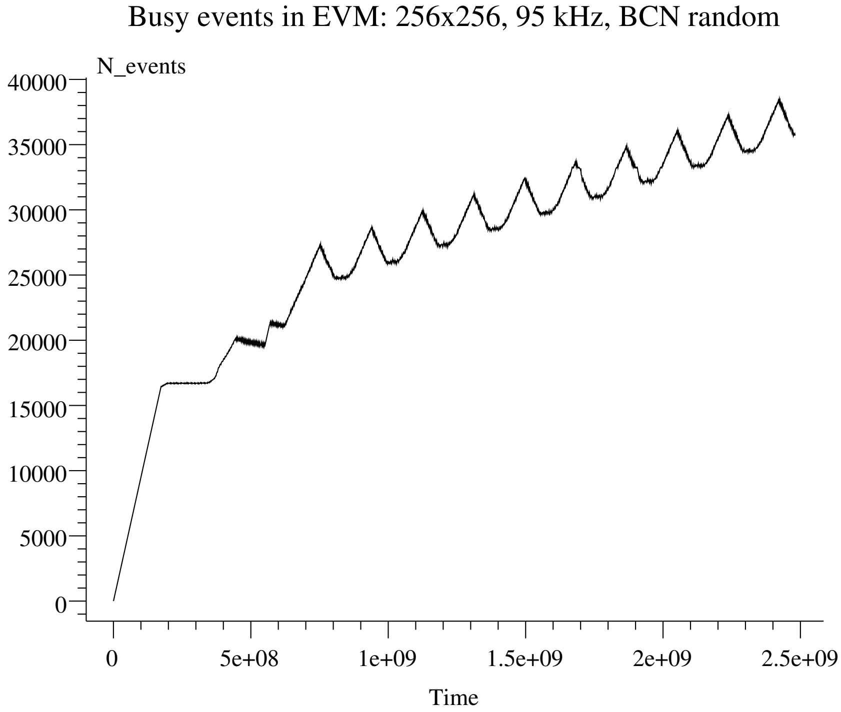

Figure 4: Events in EVM: 256x256 at 95 kHz with "random" BCN traffic.

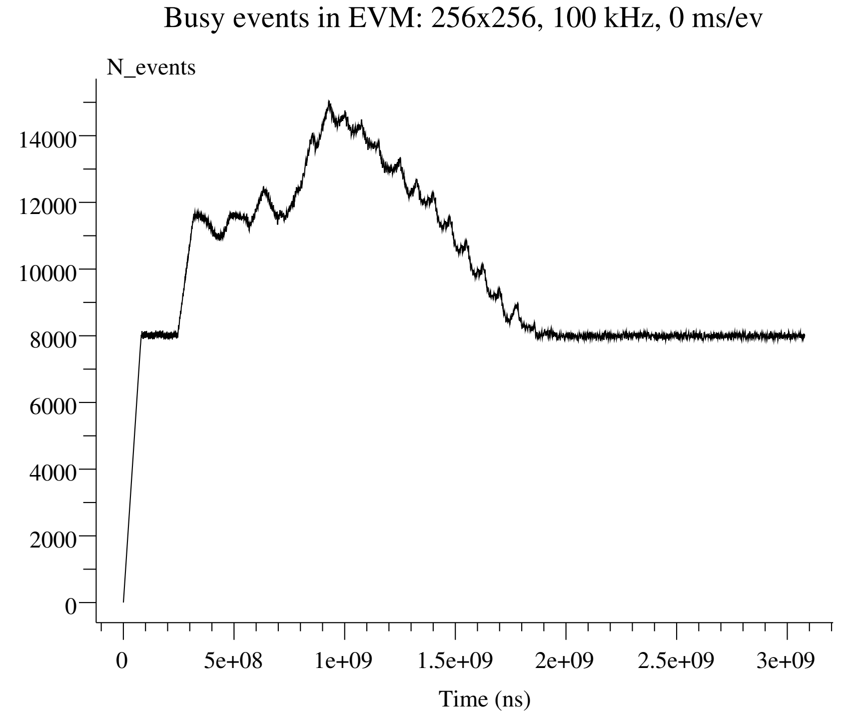

Figure 5: Events in EVM: 256x256 at 100 kHz with barrel-shifted BCN traffic.

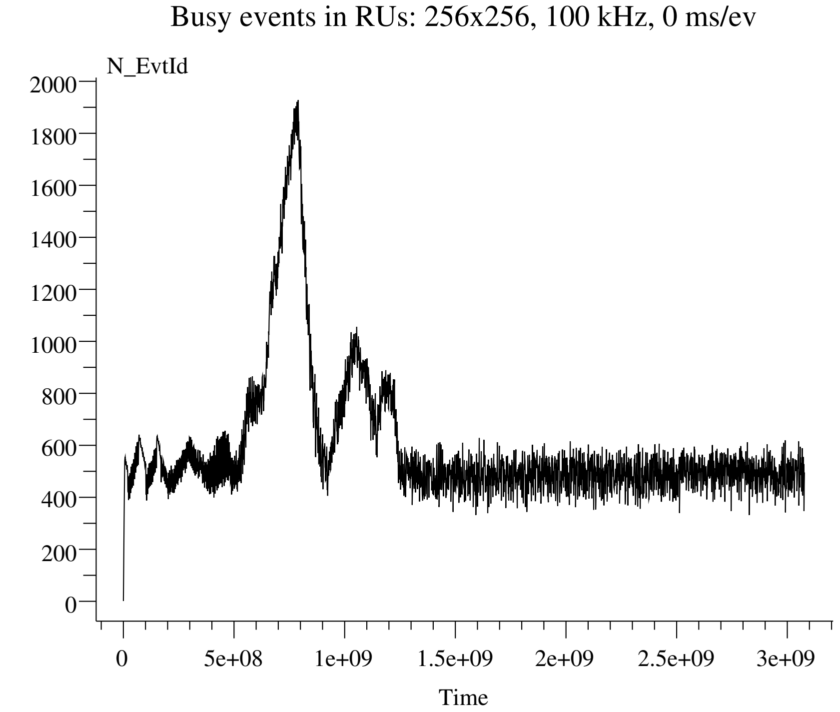

Figure 6: Events in RU: 256x256 at 100 kHz with barrel-shifted BCN traffic.

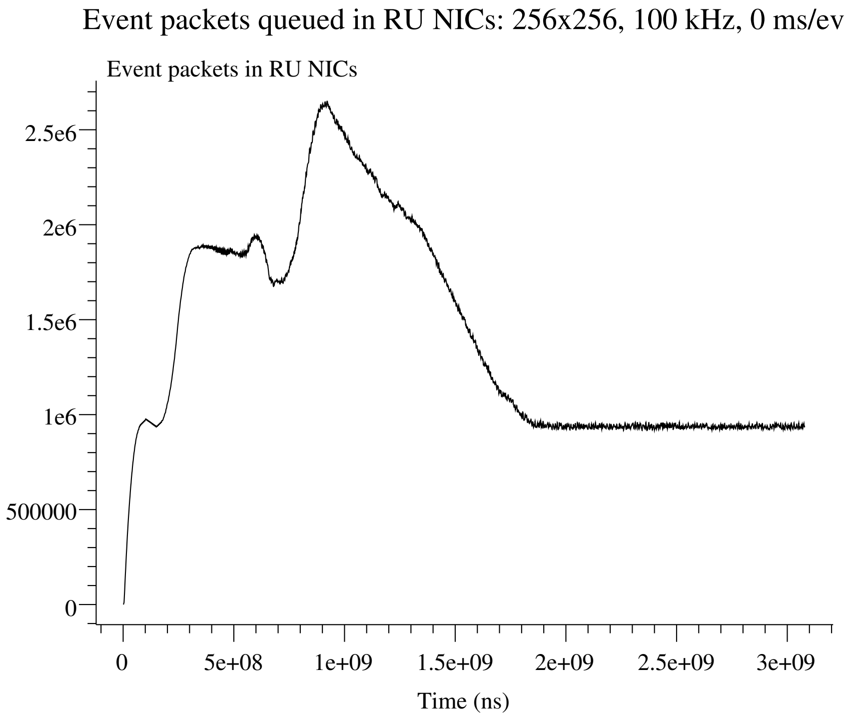

Figure 7: Events in RU NICs: 256x256 at 100 kHz with barrel-shifted BCN traffic.

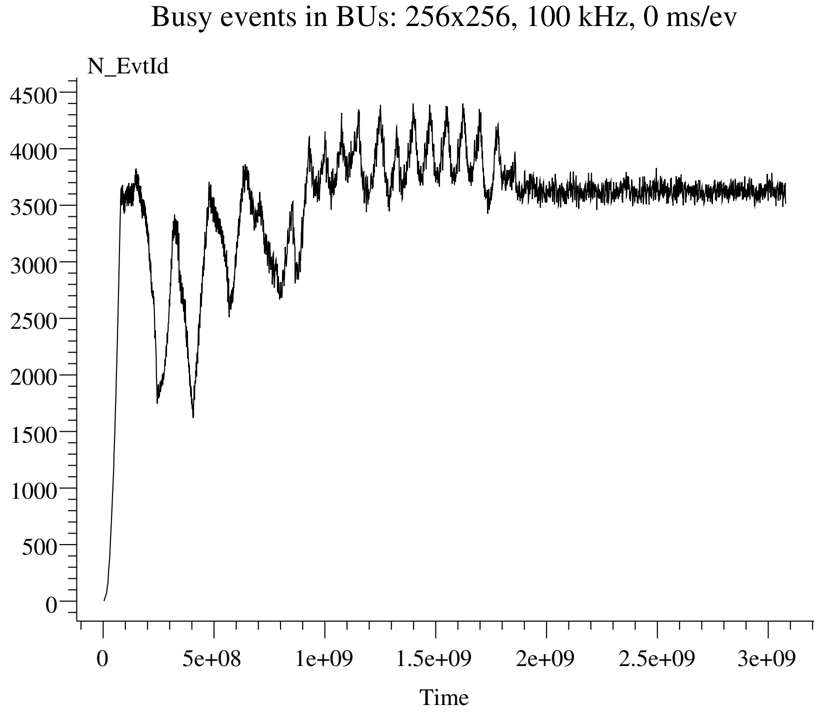

Figure 8: Events in BU: 256x256 at 100 kHz with barrel-shifted BCN traffic.

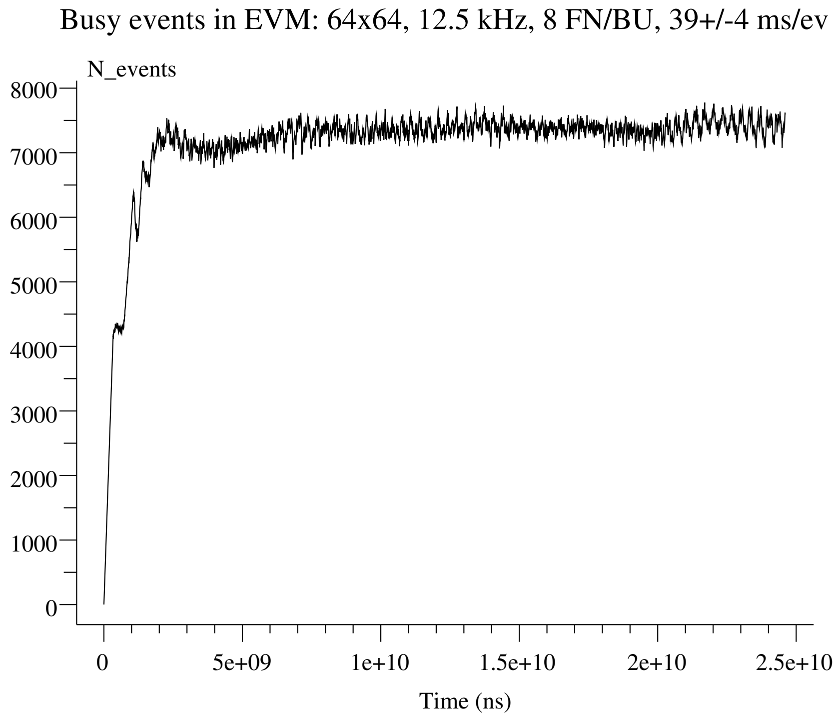

Figure 9: 64x64 system at 12.5 kHz with 8 filter nodes per BU and 39±4 ms per event.

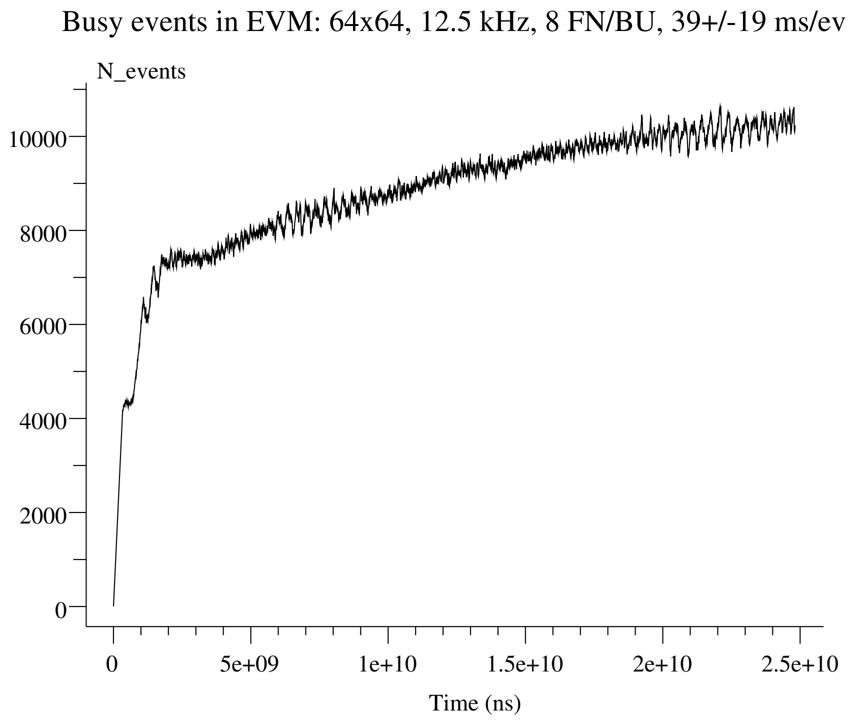

Figure 10: 64x64 system at 12.5 kHz with 8 filter nodes per BU and 39±19 ms per event.

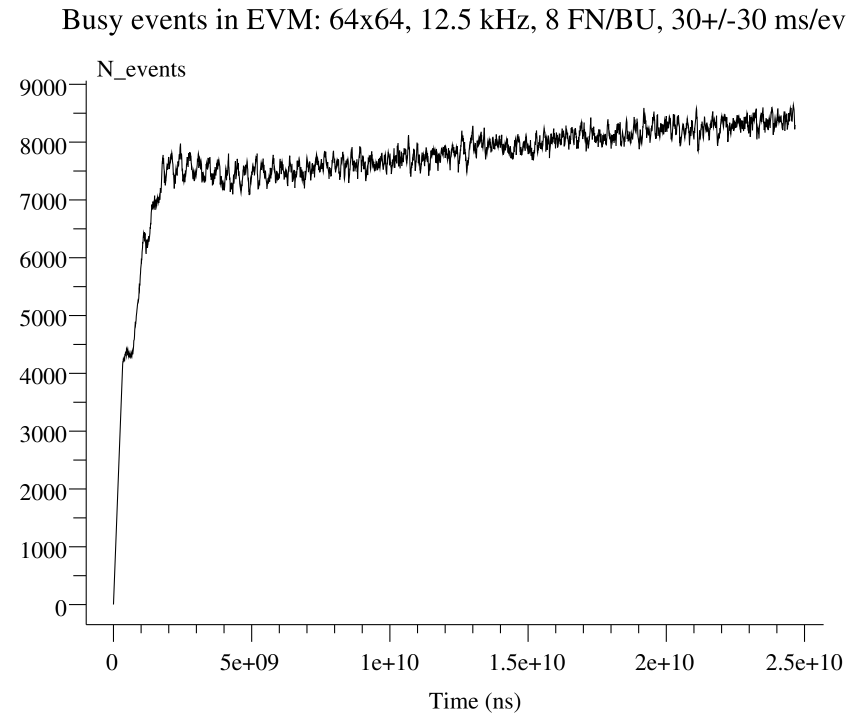

Figure 11: 64x64 system at 12.5 kHz with 8 filter nodes per BU and 30±30 ms per event.