D0Note #3221

Results from a Deadtime Analysis of the Current Design of the Level 2 Trigger for the DØ Upgrade

Jay A. Wightman

Iowa State University

Marvin Johnson

Fermilab

James Linnemann

Michigan State University

October 4, 2000

1 Background

The original proposal for the Level 2 trigger for the DØ upgrade involved a farm of processors; a deadtime analysis of this proposal was described in an earlier DØ Note[1]. The conclusions from that study showed that this design of the Level 2 trigger could not meet our demands on deadtime. The new design and a deadtime analysis of it are described in this note.

The basic design criteria we have followed in our models are:

1. 10KHz Level 1 accept rate

2. the front–end digitization crates have 16 buffers for events awaiting a Level 2 trigger decision and 8 buffers awaiting readout to Level 3 following a Level 2 accept – this is based on the design of the tracker readout boards
3. Level 2 rejection factor of 10 – this gives 1KHz Level 2 accept rate
4. Level 2 decisions must be made in the same order as the Level 1 accepts These same requirements are discussed in Ref.[1], and the reader is referred to this document for more details.

2 RESQ Model

To study the latest design we implement our model in RESQ, IBM's Research Queueing Package.[2] This is a Monte Carlo modeling package based on queueing theory and was developed by IBM to study networking issues. Several earlier modeling efforts also used RESQ[1][3], and we have confidence in using the package and understanding its results. One nice feature of RESQ is that it maintains statistics on a number of variables, like queue lengths and times. However, the number of greatest interest to our modeling is the deadtime, and this is an unknown concept to RESQ. Therefore, we are forced to calculate the deadtime ourselves, and to gain some insight into the uncertainty of the measured deadtime we calculate the standard deviation for 10 runs of 10K events each. These 10 runs are done consecutively without restarting the job due to the way in which RESQ hardwires the random number seeds. One concern with modeling systems of queues is doing measurements only after the system reaches steady–state. Our method of consecutive runs without restarting the job addresses this concern as we can study the deadtime with and without including the first run to check these startup effects.

This model deals only with the Level 2 trigger and its decision process so that it must be understood that deadtimes measured here will be lower bounds on the total system deadtime. Since we are concerned only with measuring deadtime resulting from the Level 2 trigger, we do not use the rejection of this trigger. The major difference between this design and that described in Ref. [1] is that we now separate the functionality of the trigger into a parallel set of pre–processors that are responsible for finding the trigger primitives from each sub–detector and a global processor which will combine these primitives and render a trigger decision. The model is shown in Fig. 1 using the RESQ notation for the various elements.

Figure 1: RESQ model of the Level 2 trigger.

This is an open chain system with events entering at the source, **s**, and exiting at the **sink**. The Level 2 trigger model uses an input rate (the Level 1 accept rate) which is exponentially distributed. The number of buffers in the front–end assigned to wait for a Level 2 decision is 16, consistent with the design of the VRB[4], the readout board for the tracking sub–detectors.

3 Results

The first run of the model used 16 Level 2 nodes, an input rate of 10KHz, a mean processing time, μ, in Level 2 of 1 ms, and 16 buffers assigned to Level 2 to await its decision. Each of these 16 buffers is used to hold the full event in the front–end crates while each of the Level 2 nodes is processing a subset of the appropriate event data; there is no additional buffering of the input to Level 2. Based on our experience with processing times measured in DØ we can expect to find long tails in the processing time distribution for Level 2 as well. Therefore, we use a distribution with a moderately long tail, the hyperexponential distribution with variance, $\sigma = 2 \times \mu$. The two processing time distributions that are employed in our study are shown in Fig. 2. They are the exponential distribution with $\sigma = \mu$ and the hyperexponential described above. It is quite a shock to find that the deadtime is $69.3 \pm 1.0\%$. To check that the model is correct we can remove the serialization requirement; we find a deadtime of $2.1 \pm 0.4\%$ which is in nice agreement with the analytical solution of 2.2%. Without the serialization requirement, the model permits an analytical solution, although this requires an exponential time distribution. The analytical calculation is computed using Erlang's loss formula. This is a special case of a multiple server queue where no line is allowed to form, like in the telephone system where a busy signal prevents the caller from queueing to wait for a free circuit. A unique property of the Erlang formula is that it does not depend on the processing time distribution, only its mean; thus, the good agreement between the RESQ model and the analytical solution.

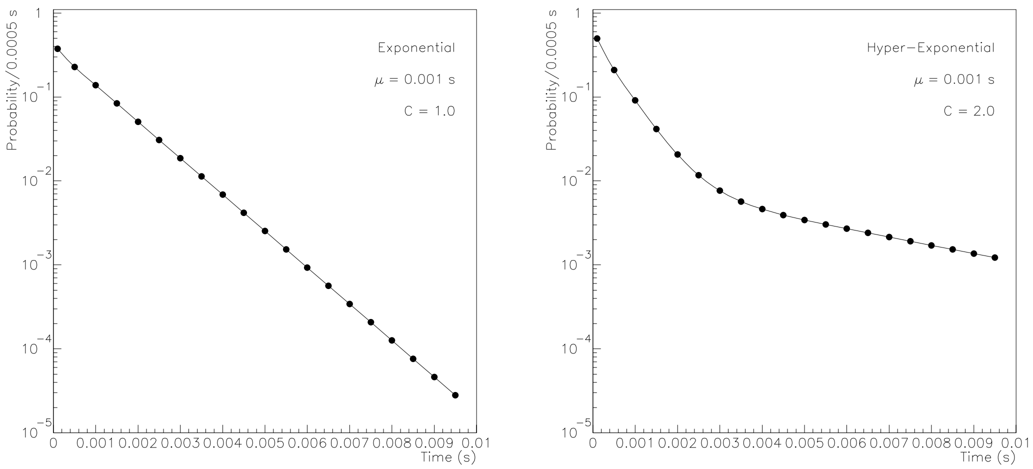

Figure 2: Input timing distributions, where μ is the mean of the distribution and the coefficient of variation, C, is the ratio of σ/μ.

The question is "Where do we go from here?" The most obvious suggestion is to add some buffering to the system. In some sense, this is the "brute–force" approach. We add 8 buffers to the input side of the Level 2 farm and find the deadtime has dropped to $67.3 \pm 1.9\%$. The next suggestion is to move the buffers to the output side of the Level 2 farm. In this way we can take advantage of the node that has just completed processing, a means for getting a jump on the rather long mean processing time. The deadtime improves a little over the buffering on the input side to $63.8 \pm 1.9\%$. Even increasing the buffers on the output side to the highly unrealistic number of 32 does not achieve the desired effect of reducing the deadtime to an acceptable level; it is $48.2 \pm 1.8\%$.

So, why is this result so bad? The answer lies in the use of a distribution with a relatively long tail on it. When we impose the serialization requirement, Erlang's formula is no longer valid, and we become very sensitive to the details of the timing distribution that we use. To show this we repeat the study using the exponential distribution in place of the hyperexponential distribution. For the case of no extra buffering the deadtime is $48.9 \pm 1.4\%$. This result is verified with the simple, independent Monte Carlo calculation which finds a deadtime of 48.1% for the case of serialization and 2.2% without. The unrealistic case of 32 extra buffers on the output side of the Level 2 farm gives a deadtime of $8.1 \pm 1.2\%$. The trend is in the correct direction, but the gradient is too shallow.

To get a better understanding of the effects of the tails of these distributions we generate 8 exponentially or hyperexponentially distributed random numbers and select the largest of the 8 to histogram separately. This should give us some idea of what the effective μ and σ are for the case of serialization. In Fig. 3 we show for our two distributions one of the generated samples along with the "worst of 8", where "worst of 8" is the distribution where we kept the largest random number of the trial. We find for our generated exponential distribution $\mu = 1.02$ ms and $\sigma = 0.99$ ms; for the hyperexponential distribution we find $\mu = 1.03$ ms and $\sigma = 1.99$ ms. The effect of the long tail in the hyperexponential distribution was further reinforced in calculating μ and σ where we generated times from 0–10 ms for the exponential distribution but needed to extend the upper limit to 50 ms for the hyperexponential distribution. For the "worst of 8" case we find $\mu = 2.71$ ms and $\sigma = 1.23$ ms for the exponential distribution and $\mu = 4.02$ ms and $\sigma = 4.19$ ms for the hyperexponential distribution. If the "worst of 8" scenario correctly describes our situation with its 16 Level 2 nodes, then the net result is that for the hyperexponential distribution we have an effective mean processing time of about 4 ms and not the 1 ms for which we had originally designed the system. It is no wonder that this has such a large effect on the system performance.

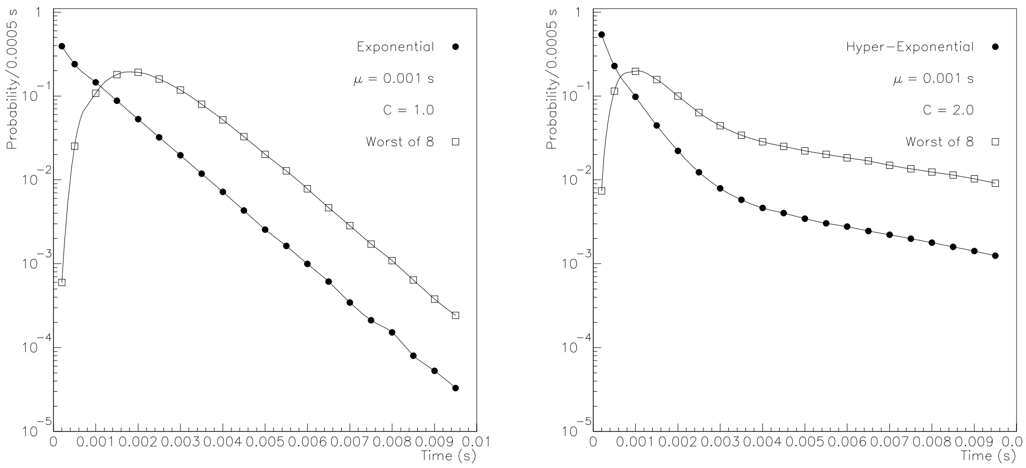

Figure 3: Exponential and hyperexponential distributions and the "worst of 8" for each.

4 Dealing with the long tails

There are two obvious avenues to pursue here, truncating the tail of the distribution or reducing the mean processing time without truncation. RESQ does not supply a distribution that can be truncated; it must be supplied by the user. However, all attempts to date to include a user–specified distribution have met with failure. Not wanting to let this impede progress we use a simple variation on the theme and fix the processing time in the Level 2 farm. This gives us an upper limit on the deadtime since we do not generate any smaller times. It also gives us serialization free of charge. We include in this set of runs the opportunity of adding a modest number of buffers to the output of the Level 2 farm to see if we could reach the desired 1% deadtime goal. The results are shown in Fig. 4 where we have included a dashed line at a deadtime of 1%. The independent Monte Carlo is used to study the effect of truncating the tail of the exponential distribution. The assumption made for this run is that we cannot afford to cut too strongly since it will adversely affect the rejection power of Level 2; our criteria for truncating the distribution is 1/3R, where R is the untruncated rejection factor. For R =

10, we find we can only truncate $\sim 3.3\%$ of the tail; this results in a deadtime of 34.7% for the case of serialization. The conclusion here is the same as we have found in the past, small perturbations tend to produce small effects on the outcome.

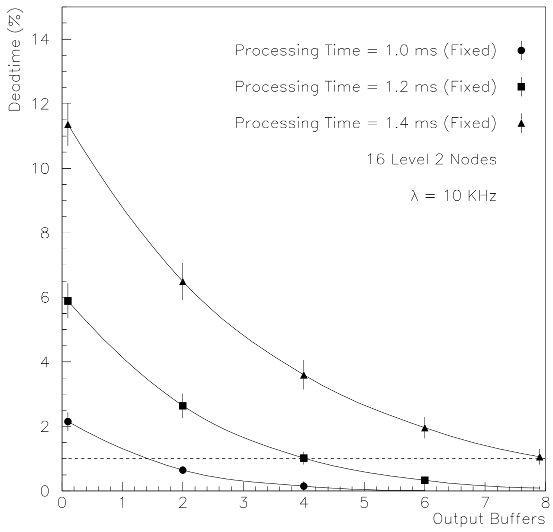

Figure 4: Effect of fixing the Level 2 farm processing time and adding a modest number of buffers to the output of the farm; λ is the input rate.

The second study deals with reducing the mean processing time without truncating the tail of the distribution. For this study we use the exponential distribution so we do not include the effect of long tails. In addition, we do not include any extra buffers in the system. The processing time is reduced from 1 ms to 0.1 ms in steps of 0.1 ms, and the results are presented in Fig. 5. Once again, the independent Monte Carlo model verifies these results as well. We find from the Monte Carlo that with the reduction of about a factor of 4 in the mean processing time, as shown in Fig. 5, we cannot quite recover by increasing the number of Level 2 nodes by a factor of 4.

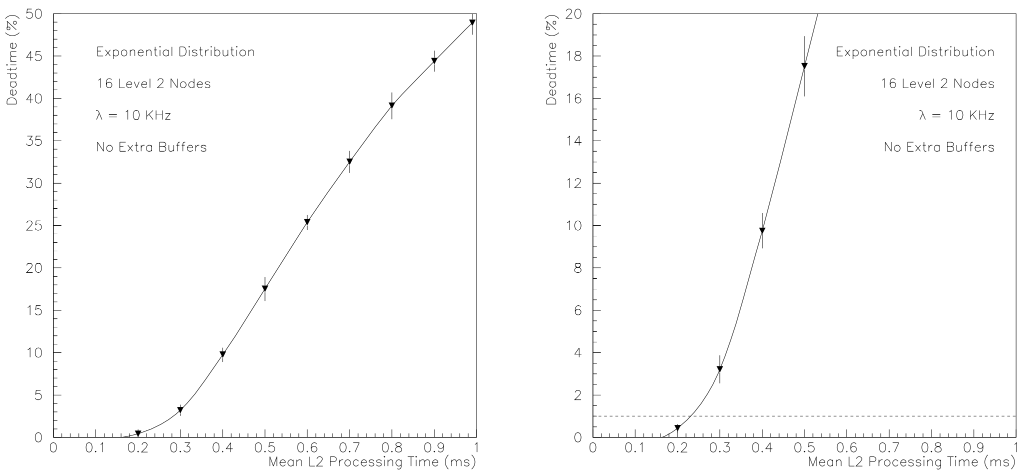

Figure 5: Effect of reducing the mean processing time for the exponential distribution. The right–hand figure shifts the focus to the low deadtime region.

5 Conclusions

We find that the serialization requirement is very tough and has profound consequences. Brute force attempts to solve the problem either fail outright, like throwing more buffers at it, or they generate their own set of side–effects. While truncating the processing time looks promising it has its own pitfall. If we truncate the processing time before a decision is made, we need to decide what to do with these events. If we pass them, we effectively reduce our rejection in the Level 2 trigger. This also will affect the readout to the Level 3 processor farm since very long decision times generally result from very busy events. However, if we fail them, we potentially add a bias to the trigger that may be very difficult to understand. Therefore, we are faced with probably moving the cutoff on the processing time well away from the mean, thus limiting its usefulness. Reducing the mean processing time is reflected directly in reducing the time budget for the Level 2 trigger. It has been estimated that the time to get the data to the Level 2 farm is on the order of 50–100 μs, so reduction of the mean time in our model must necessarily reduce the complexity of the trigger resulting in a lower rejection once again. Even combining these does not give us sufficient manuvering room to guarantee success. The obvious alternative that we see at the moment is to remove the serialization requirement, but this has consequences for the hardware designers.

References

[1] "Results from a Deadtime Analysis of the Current Proposal for a Level 2 Trigger for the DØ Upgrade," Jay A. Wightman, Marvin Johnson, and James Linnemann, DØ Note #2783 (unpublished).

[2] "The Research Queueing Package Version 3," R.F. Gordon, P.G. Loewner, and E.A. MacNair, IBM Report RA 210 (#77143).

[3] "Modeling of the DZero Data Acquisition System," R. Angstadt, M. Johnson, I.L. Manning, and J.A. Wightman, *IEEE Transactions on Nuclear Science* 39, 904 (1992); and "The DZero Data Acquisition System," J.A. Wightman *submitted to the Proceedings of the 5^{th} International School–Seminar on Automation and Computing in Nuclear Physics and Astrophysics.*

[4] "VME Readout Buffer," H. Gonzalez, D. Mendoza, M. Bowden, T. Zmuda, M. Johnson, and E. Barsotti, Document #ESE–SVX–950719 (October 25, 1995 – unpublished).

[5] Mike Fortner, private communication.