Effects of Interrupt Coalescence on Network Measurements *

Ravi Prasad, Manish Jain, and Constantinos Dovrolis

College of Computing, Georgia Tech., USA
ravi,jain,dovrolis@cc.gatech.edu

Abstract. Several high-bandwidth network interfaces use Interrupt Coalescence (IC), i.e., they generate a single interrupt for multiple packets sent or received in a short time interval. IC decreases the per-packet interrupt processing overhead. However, IC also introduces queueing delays and alters the "dispersion" (i.e., interarrival time spacing) of packet pairs or trains. In this work, we first explain how IC works in two popular Gigabit Ethernet controllers. Then, we identify the negative effects of IC on active and passive network measurements. Specifically, we show that IC can affect active bandwidth estimation techniques, causing erroneous measurements. It can also alter the packet interarrivals in passive monitors that use commodity network interfaces. Based on the "signature" that IC leaves on the dispersion and one-way delays of packet trains, we show how to detect IC and how to filter its effects from raw measurements. Finally, we show that IC can be detrimental to TCP self-clocking, causing bursty delivery of ACKs and subsequent bursty transmission of data segments.

1 Introduction

The arrival and departure of packets at a Network Interface Card (NIC) are two events that the CPU learns about through interrupts. An interrupt-driven kernel, however, can get in a *receive livelock* state, where the CPU is fully consumed with processing network interrupts instead of processing the data contained in received packets [1]. In Gigabit Ethernet (GigE) paths, 1500-byte packets can arrive to a host in every $12\mu s$. If the interrupt processing overhead is longer than $12\mu s$, receive livelock will occur when an interrupt is generated for every arriving packet.

The system overhead for each arriving packet consists of a context switch, followed by the execution of the network interrupt service routine, followed by

* This work was supported by the "Scientific Discovery through Advanced Computing" program of the US Department of Energy (award number: DE-FG02-02ER25517), by the "Strategic Technologies for the Internet" program of the US National Science Foundation (award number: 0230841), and by an equipment donation from Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the previous funding sources.

another context switch. In order to reduce the per-packet CPU overhead, and to avoid receive livelock, most high-bandwidth network interfaces today (in particular GigE cards) use *Interrupt Coalescence* (IC). IC is a technique in which NICs attempt to group multiple packets, received in a short time interval, in a single interrupt. Similarly, at the sending host, the CPU learns about the departure of several packets through a single interrupt. IC is not a new technique. It becomes common whenever a new LAN technology brings the network speed closer to the CPU speed; for instance, it was also used in the early days of Fast Ethernet NICs [1].

In this work, we identify and describe the negative effects of IC on both active and passive network measurements. Specifically, IC can affect active bandwidth estimation techniques, when the latter ignore the possibility of IC at the receiver's NIC [2]. Bandwidth measurements are classified as either capacity estimation or available bandwidth estimation. For a recent survey, we refer the reader to [3]. Capacity estimation is based on the analysis of dispersion measurements from back-to-back packet pairs and packet trains. Available bandwidth estimation, on the other hand, is based on the relation between the departure and arrival rates of periodic packet streams.

We have modified our previous capacity and available bandwidth measurement techniques, presented in [4] and [5] respectively, so that they can provide accurate estimates *in the presence of IC*. For capacity estimation, a sufficiently long packet train can provide a clear IC signature, in which packets are received in successive bursts at a certain time period. Based on this signature, we can detect IC, filter our the erroneous measurements, and estimate the correct path capacity. For available bandwidth estimation, we can similarly detect the IC signature in the one-way delay measurements of periodic packet streams. It is then simple to ignore the one-way delay measurements that have been affected by IC, and to determine whether the remaining measurements show an increasing trend.

IC can also affect passive measurements that use commodity NICs for packet trace capture. The reason is that IC can alter the packet interarrivals in the monitored traffic stream. We finally show that IC can be detrimental to TCP self-clocking, causing bursty delivery of ACKs to the sender and bursty transmission of data segments.

The rest of the paper is structured as follows. §2 explains how IC works in two popular GigE controllers. §3 illustrates the negative effects of IC in both active and passive network measurements, and it describes how to filter dispersion and one-way delay measurements in the presence of IC. We conclude in §4.

2 Description of Interrupt Coalescence

The basic objective behind IC is to reduce the number of network interrupts generated in short time intervals (in the order of a few hundereds of microseconds). This can be achieved by delaying the generation of an interrupt for a few hundreds of microseconds, expecting that more packets will be received in the meanwhile. Most GigE NICs today support IC by buffering packets for a variable time period (*dynamic* mode), or for a fixed time period (*static* mode) [6, 7]. In *dynamic* mode, the NIC controller determines an appropriate interrupt generation rate based on the network/system load. In *static* mode, the interrupt generation rate or latency is set to a fixed value, specified in the driver.

Users (with root privileges) can modify the default controller behavior by changing the parameters provided by the NIC drivers. For instance, the driver for the Intel GigE E1000 controller [8] provides the following parameters for adjusting the IC receive (Rx) and transmit (Tx) operations:

- *InterruptThrottleRate*: the maximum number of interrupts per second. It is implemented by limiting the minimum time interval between consecutive interrupts.
- *(Rx/Tx)AbsIntDelay*: the delay between the arrival of the first packet after the last interrupt and the generation of a new interrupt. It controls the maximum queueing delay of a packet at the NIC buffers.
- *(Rx/Tx)IntDelay*: the delay between the last arrival of a packet and the generation of a new interrupt. It controls the minimum queueing delay of a packet at the NIC buffers.

Note that the previous parameters can be combined to meet constraints on the maximum interrupt rate and on the maximum/minimum IC-induced queueing delays. In case of conflict, *InterruptThrottleRate* has a higher precedence than *(Rx,Tx)AbsIntDelay* and *(Rx,Tx)IntDelay* parameters.

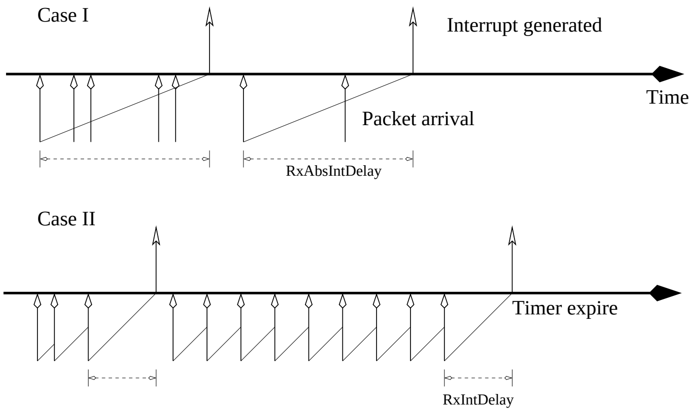

Fig. 1. Illustration of IC with *RxAbsIntDelay* (Case I) and with *RxIntDelay* (Case II).

Figure 1 illustrates the interrupt generation process at the receiving host with only the *RxIntDelay* or *RxAbsIntDelay* constraint. In case I, the *RxAbsIntDelay* timer is set to expire after a certain delay since the arrival of the first packet after the last interrupt. All the subsequent packets are buffered at the NIC buffers, and they are delivered with the next interrupt. If the minimum packet spacing at the given NIC is T, the maximum number of packets that may need to be buffered is $RxAbsIntDelay/T$. In case II, every packet arrival resets the *RxIntDelay* interrupt timer. Consequently, if packets arrive periodically with a lower dispersion than *RxIntDelay*, the interrupt generation can be delayed indefinitely, causing NIC buffer overflows.

The SysKonnect GigE NIC [9] provides two parameters: *moderate* and *intpersec*. *Moderate* determines whether IC is enabled, and whether it is in *static* or *dynamic* mode. *Intpersec* determines the maximum interrupt generation rate in static mode. It is similar to the *InterruptThrottleRate* of the Intel GigE NIC.

3 Effects of Interrupt Coalescence

In the following, we explain how IC can affect active and passive measurements as well as TCP self-clocking. We also describe the IC signature on packet train dispersion and one-way delay measurements, and show how to filter out the effects of IC.

3.1 Capacity Estimation.

Active bandwidth estimation tools use the received dispersion (interarrival time spacing) of back-to-back packet pairs (or trains) to estimate the end-to-end capacity of a network path [3]. The basic idea is that, in the absence of cross traffic, the dispersion of a packet pair is inversely proportional to the path capacity. Note that the received dispersion is typically measured at the application or kernel, and not at the NIC. If the receiver NIC performs IC, however, the packet pair will be delivered to the kernel with a single interrupt, destroying the dispersion that the packet pair had at the network.

Figure 2 shows the cumulative dispersion of an 100-packet Ethernet MTU (1500B) train, with and without IC, at the receiver of a GigE path. The capacity estimate, as obtained without IC, is quite close to the actual capacity of the path (about 1000Mbps). In the case of IC, however, we see that the application receives 10 to 12 packets at a very high rate (unrelated to the network capacity), separated by about 125μs of idle time. Note that, if a bandwidth estimation tool would attempt to measure the path capacity from the dispersion of packet pairs, it would fail miserably because there is not a single packet pair with the correct dispersion.

Detection of IC: In the presence of IC, many packets will be delivered from the NIC to the kernel, and then to the application, with a single interrupt. We refer to the packets that are transferred to the application with a single interrupt as a *burst*. The dispersion of successive packets in a burst, measured at the application layer, will be equal to the latency T of the *recvfrom* system call. A measurement tool can measure T, and so it can determine which packets of a received train form a *burst*.

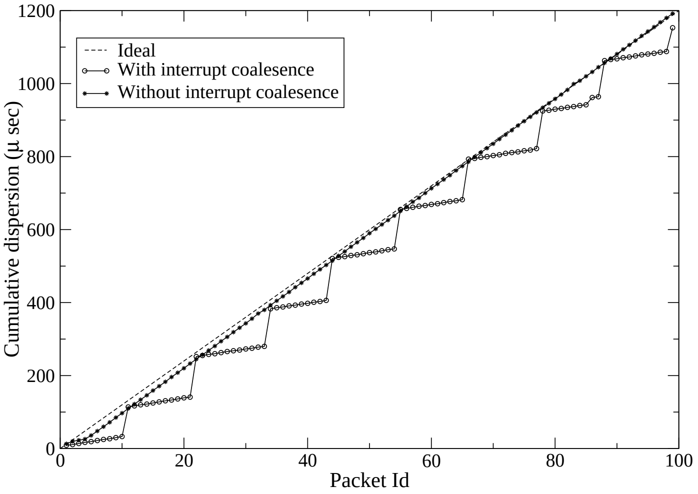

Fig. 2. Cumulative dispersion in an 100-packet train with and without IC.

The presence of IC at the receiver NIC can be detected by sending a long back-to-back packet train from the sender to the receiver. If IC is enabled, the receiving application will observe a sequence of bursts of approximately the same length and duration. For example, Figure 3 shows that bursts of 10-12 packets arrive roughly every 125μs. A packet train of about 50 packets would be sufficient to detect the IC signature reliably.

An important point here is that a context switch at the receiver can cause a somewhat similar signature with IC. Figure 3 illustrates that context switching can also force several consecutive packets to be delivered to the application with the *recvfrom* period. However, as shown in Figure 3, there is a clear difference between the signatures of context switching and IC. The former is a probabilistic event that only affects a subset of successive packets, starting from a random initial location in the packet train. The latter is a deterministic event that affects all the packets of a packet train, starting from the first, and creating a regular pattern of burst lengths and durations.

Estimation with IC: Our capacity estimation tool, pathrate [10], uses the dispersion of consecutive packets in 50-packet trains to detect the presence of IC. To estimate the path capacity in the presence of IC, pathrate relies on the interarrivals between successive bursts. Without significant cross-traffic, the dispersion between the first packet of two consecutive bursts is equal to BL/C, where B is the length of the first burst, L is the packet size, and C is the path capacity. The train length should be sufficiently long so that at least two bursts are observed in each received train. Pathrate sends multiple packet trains and finally reports the median of the resulting capacity estimates.

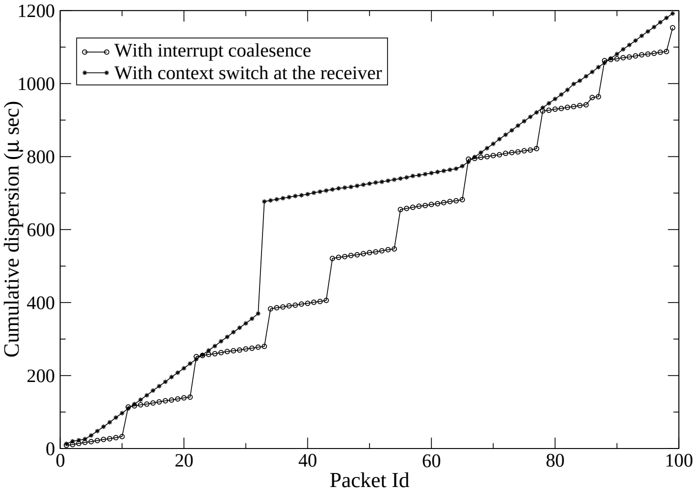

Fig. 3. The signatures of context switching and interrupt coalescence in the cumulative dispersion of a packet train.

3.2 Available Bandwidth Estimation.

Available bandwidth estimation tools examine the relationship between the *send rate* R_s and *receive rate* R_r of periodic packet streams in order to measure the end-to-end available bandwidth A [3, 11–13]. The main idea is that when $R_s > A$, then $R_r < R_s$; otherwise, $R_r = R_s$. Furthermore, when $R_s > A$, the queue of the tight link builds up, and so packet i experiences a larger queuing delay than packet $i - 1$. Consequently, when $R_s > A$, the One-Way Delay (OWD) of successive probing packets is expected to show an increasing trend. On the other hand, if $R_s < A$, the OWDs of probing packets will be roughly equal. Pathload [5] and pathchirp [14] estimate the available bandwidth of a path by analyzing OWD variations.

Figure 4 shows the OWDs of a periodic 100-packet Ethernet MTU stream with and without IC. The sender timestamps and transmits packets at 1.5Gbps. The available bandwidth in the path is about 940 Mbps. The OWDs of successive packets, without IC, are clearly increasing showing that the probing rate is greater than the available bandwidth. However, in the presence of IC, packets are buffered at the NIC until the interrupt timer expires. Then, the packets are delivered back-to-back to the kernel, and then to the application. We refer to all successive packets delivered with a single interrupt as a *burst*. Among the packets of a burst, the first experiences the largest queueing delay at the NIC, while the last observes the minimum (or zero) queueing delay at the NIC. Notice that the increasing OWD trend of the packets within a burst has been destroyed due to IC.

Suppose that s_k is the send time of the k'th packet in a certain burst, while t is when the interrupt is generated. Also, let r be the latency to transfer a packet

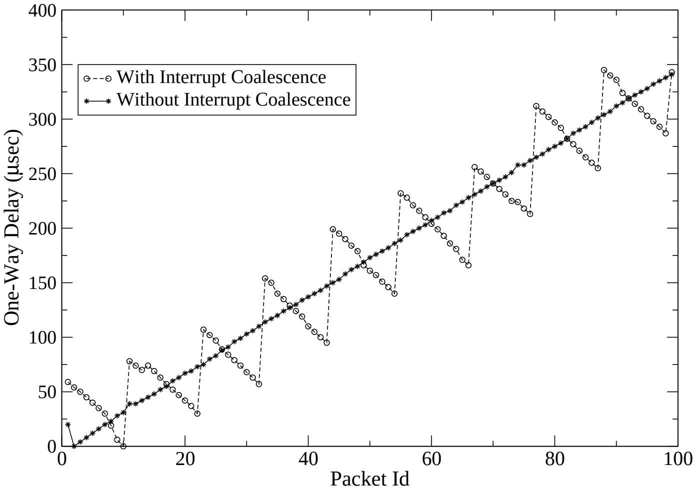

Fig. 4. OWDs in 100-packet train with and without IC.

from the NIC to user space, and g be the period between successive packets at the sender. Then, the OWD of the k'th packet will be given by

$$\begin{aligned} d_k &= t + k * r - s_k \\ &= t + k * r - (s_1 + k * g) \end{aligned}$$

and so the relative OWD of packet $k + 1$ will be

$$d_{k+1} - d_k = r - g \tag{1}$$

From Equation (1), we see that if $g > r$, the successive OWDs in the same burst are decreasing. In the experiment of Figure 4, we have that $g = 8\mu$s, $r \approx 3\mu$s, and so there is a clear decreasing trend in the OWDs of each burst.

Note that tools that only use packet pairs, such as [12, 14], or short packet streams (less than about 10 packets in our GigE cards) will fail to detect IC, since none of the packet pairs, or short streams, shows a correct OWD trend. On the other hand, notice that, although the OWD trend is destroyed in short timescales, the overall increasing trend in the OWDs is still preserved. Based on this observation, we have included an algorithm in pathload to detect IC and measure available bandwidth even in the presence of IC.

Estimation with IC: Pathload uses the same technique as pathrate (see §3.1), to detect whether the receiver NIC performs IC. When that is the case, pathload filters out the OWD measurements that have been affected the most from IC. As explained in the previous paragraph, the queueing delay at the NIC is minimum (or zero) for the last packet of a burst. So, pathload rejects all OWD measurements *except the last of each burst*. From the remaining measurements, we can then examine whether an increasing OWD trend exists or not.

3.3 Passive measurements

Passive monitors are often used to collect traces of packet headers and interarrivals. Several studies have focused on the traffic statistical characteristics (burstiness) in short timescales, by analyzing such packet traces. However, if the traces have been collected with commodity NICs that perform IC, then the packet interarrivals can be significantly altered. First, IC can make several packets appear as if they form a burst, even though that may not be the case. Second, IC can affect the correlation structure of packet interarrivals in short timescales.

We emphasize that special-purpose passive monitors typically timestamp each packet at the NIC, and so they avoid the negative effects of IC [15].

3.4 TCP self-clocking

Another negative effect of IC is that it can break TCP self-clocking [16]. Specifically, the TCP sender establishes its *self-clock*, i.e. determines how often it should be sending packets based on the dispersion of the received ACKs. To preserve the dispersion of ACKs at the sender, or the dispersion of data segments at the receiver, packets must be delivered to TCP as soon as they arrive at the NIC. IC, however, results in bursty delivery of data segments to the receiver, destroying the dispersion that those packets had in the network. The bursty arrival of segments at the receiver causes bursty transmission of ACKs, and subsequently bursty transmission of more data segments from the sender. The problem with such bursts is that TCP can experience significant losses in under-buffered network paths, even if the corresponding links are not saturated [17].

Figure 5 shows the CDF of ACK interarrivals in a 10-second TCP connection over a GigE path. Without IC, most ACKs arrive approximately every 24μs, corresponding to the transmission time of two MTU packets at a GigE interface. This is because TCP uses delayed-ACKs, acknowledging every second packet. With IC, however, approximately 65% of the ACKs arrive with an erratic dispersion of less than 1μs, as they are delivered to the kernel with a single interrupt. These “batched” ACKs trigger the transmission of long bursts of data packets from the sender. Note that the rest of the ACKs, even though non-batched, they still have a dispersion that does not correspond to the true capacity of the path, misleading the TCP sender about the actual rate that this path can sustain.

4 Conclusions

IC is used to reduce the interrupt processing overhead and becomes necessary when the minimum packet interarrival latency becomes comparable to the per-packet interrupt processing overhead. Unfortunately, IC can affect active network measurement tools that use closely spaced packet pairs, trains, or streams. In particular, capacity and available bandwidth estimation techniques can provide incorrect results if they ignore IC. On the positive side, we developed a simple algorithm to detect the presence of IC, and to filter out the erroneous dispersion or one-way delay measurements caused by IC.

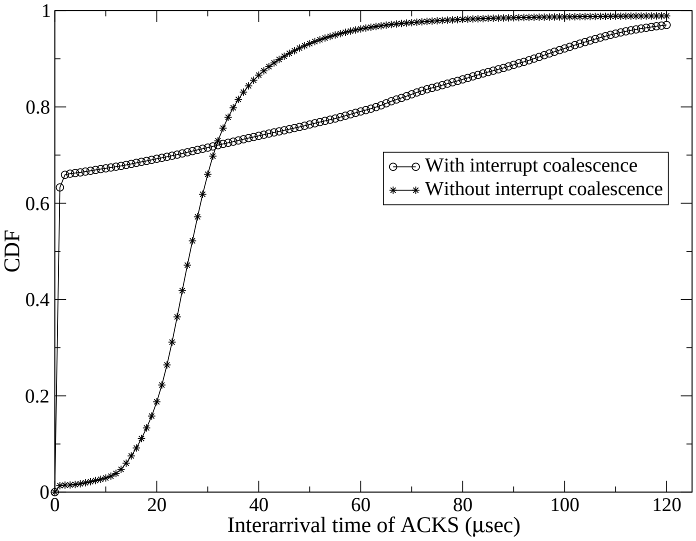

Fig. 5. CDF of ACK interarrivals in a 10-second TCP connection over a GigE path.

References

1. Mogul, J.C., Ramakrishnan, K.K.: Eliminating receive livelock in an interrupt-driven kernel. ACM Transactions on Computer Systems **15** (1997) 217–252
2. Jin, G., Tierney, B.L.: System Capability Effects on Algorithms for Network Bandwidth Measurement. In: Proceedings of ACM Internet Measurement Conference. (2003)
3. Prasad, R.S., Murray, M., Dovrolis, C., Claffy, K.: Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. IEEE Network (2003)
4. Dovrolis, C., Ramanathan, P., Moore, D.: Packet Dispersion Techniques and Capacity Estimation. IEEE/ACM Transactions on Networking (2004)
5. Jain, M., Dovrolis, C.: End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. IEEE/ACM Transactions on Networking **11** (2003) 537–549
6. Intel: Interrupt Moderation Using Intel Gigabit Ethernet Controllers. http://www.intel.com/design/network/applnots/ap450.pdf (2003)
7. Syskonnect: SK-NET GE Gigabit Ethernet Server Adapter. http://www.syskonnect.com/syskonnect/technology/SK-NET_GE.PDF (2003)
8. Intel: Intel Gigabit Ethernet Driver. http://sourceforge.net/projects/e1000 (2004)
9. Syskonnect: SysKonnect Gigabit Ethernet Driver. http://www.syskonnect.com/syskonnect/support/driver/ge.htm (2003)
10. : Pathrate and Pathload. http://www.pathrate.org/ (2004)
11. Melander, B., Bjorkman, M., Gunningberg, P.: A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks. In: IEEE Global Internet Symposium. (2000)
12. Pasztor, A.: Accurate Active Measurement in the Internet and its Applications. PhD thesis, The University of Melbourne (2003)

13. Hu, N., Steenkiste, P.: Evaluation and Characterization of Available Bandwidth Probing Techniques. IEEE Journal on Selected Areas in Communications **21** (2003) 879–894
14. Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., Cottrell, L.: pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In: Proceedings of Passive and Active Measurements (PAM) workshop. (2003)
15. Endace: Endace Measurement Systems. http://dag.cs.waikato.ac.nz/ (2004)
16. Jacobson, V.: Congestion Avoidance and Control. In: Proceedings of ACM SIGCOMM. (1988) 314–329
17. Zec, M., Mikuc, M., Zagar, M.: Estimating the Impact of Interrupt Coalescing Delays on Steady State TCP Throughput. (2002)