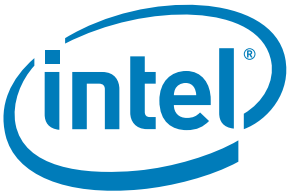
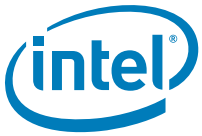


Interrupt Moderation Using Intel® GbE Controllers

April 2007

316922-001
Revision 1.2



Contents

Revision History

Revision	Revision Date	Description
1.2	April 2007	Major edit all sections.
1.1	Sept 2003	The document was modified to be applicable to Intel Gigabit Ethernet Controllers except the 82542, 82543, and 82544. In other words, this document applies to the 82540, 82545, 82546, 82541, and 82547.
1.0	May 2003	Initial release.

Note: This page intentionally left blank.

1.0 Introduction

This application note describes how to use the interrupt moderation features of the Intel® GbE Controllers.

Note: The reader is assumed to have a working knowledge of network device drivers.

1.1 Reference Documents

- *PCIe* Family of Gigabit Ethernet Controllers Software Developer's Manual*
- *PCI/PCI-X Family of Gigabit Ethernet Controllers Software Developer's Manual*

1.2 Glossary of Terms

Windows* Adaptive Mode - dynamically adjusts interrupt rates (3,000 to 20,000) to the I/O patterns received over the wire.

Linux* Mode 3 - dynamically adjusts interrupt rates (4,000 to 20,000) to the I/O patterns received over the wire.

Linux* Mode 1 – dynamically adjusts interrupt rates (4,000 to 70,000) to the I/O patterns received over the wire.

Latency – The amount of time it takes for a packet of data to get from one designated point to another.

Throughput - The amount of digital data per time unit that is delivered over a network. Usually measured in bits per second (bps).

CPU Utilization – Amount of CPU time required to transfer data over a network.

1.3 Supported GbE Devices

Windows*	Linux*	
PCIe* Devices	**PCI-X Devices**	**PCIe* Devices**
82571/82572	82540	82571/82572
82573	82541	82573
631xESB/632xESB	82545	631xESB/632xESB
82575	82546	82575
	82547	

Use these links to help identify an adapter or GbE device:

- http://www.intel.com/design/network/products/ethernet/linecard_ec.htm
- http://www.intel.com/network/connectivity/products/server_adapters.htm
- http://www.intel.com/network/connectivity/products/desktop_adapters.htm
- http://support.intel.com/support/network/adapter/pro100/21397.htm

1.4 Supported Operating Systems and Drivers

Linux*	Windows*
• Red Hat • Novell SUSE Kernel versions 2.4.x and 2.6.x	• 2000 • XP • Server 2003 • Vista

- Linux* driver versions: 7.3.15 and higher

Note: Linux* drivers must be downloaded because they are not currently included in distributions (releases).

- Windows* driver versions: 9.6.31.0 and higher

For the latest Intel network drivers for Linux*, go to:

- http://downloadfinder.intel.com/scripts-df/support_intel.asp

For the latest Intel network drivers for Windows*, go to:

- http://support.intel.com/support/network/sb/CS-006120.htm

2.0 Background

Interrupt moderation reduces host processor interrupts, thereby enabling technologies such as Gigabit EtherChannel* to deliver more of their 16-Gb/s bandwidth potential (8 Gb/s x full duplex). As a result, performance gains from load-balancing used in adapter teaming becomes a smaller part of overall performance as compared to the performance gains achieved through the increased server CPU headroom provided by interrupt moderation.

Host processor interrupts are generated by the GbE controller in order to request cycles for packet processing. These interrupts need to be controlled to achieve optimum throughput. Too few interrupts can lead to latencies and too many can unduly burden the server's processor. By bundling an appropriate number of packets before issuing an interrupt to the host (Figure 1), the GbE controller tunes interrupt frequency to match traffic conditions while maintaining packet flow.

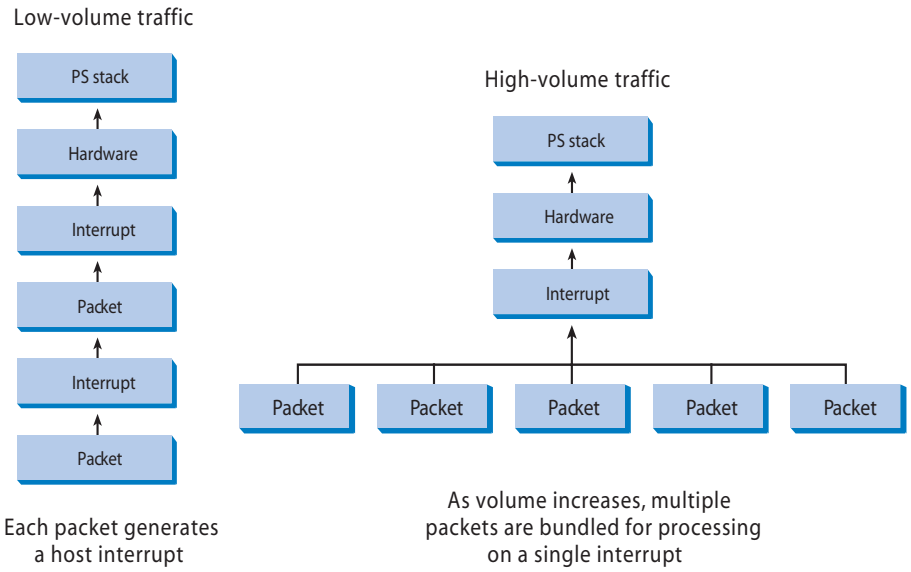


Figure 1. Host Interrupt Modulation

2.1 Basic Interrupt Processing

Without any interrupt moderation, the GbE controller interrupts the CPU after every packet event (both packet transmission and reception). This involves:

1. The GbE controller successfully transmits or receives a packet.
2. The GbE controller generates an interrupt in response to this event.
3. The CPU suspends its current activity in order to handle the interrupt. This involves saving state information and executing an interrupt handler for the GbE controller.
4. Software (device drivers) examine the GbE controller to determine the cause of the interrupt. Software might also take additional action(s) based on the exact nature of the interrupt.
5. The CPU resumes its previous activity.

At low traffic rates, this behavior is acceptable since this process occurs relatively infrequently. However, as traffic rates increase, the system spends more and more time servicing these interrupts. The overhead of processing these interrupts begins to degrade overall system performance as the CPU spends the majority of its time scheduling and executing the interrupt handler. If the traffic rate continues to increase, the traffic might overrun the GbE controller causing it to drop packets, or the system itself might become temporarily unusable.

In addition to the packet events described, the GbE controller might also interrupt after certain external events, such as a change in link status. Since these other events occur relatively infrequently, they do not usually degrade system performance in the same manner as the packet events.

2.2 Interrupt Handling with Interrupt Moderation

To avoid potential problems of interrupts from packet events, the GbE controller employs a series of timers for moderating or limiting the number of interrupts it generates.

Rather than immediately interrupting the transmission or reception of a new packet, software can configure the GbE controller to delay the generation of this interrupt. By postponing this interrupt, the GbE controller can collect any additional interrupt events that arrive within this delay period. When it finally does interrupt, the GbE controller is then able to deliver several events to software all at once.

Software can typically process these events in a single iteration of its interrupt handler. By bundling and processing interrupts in bulk, software and hardware can operate at higher rates of traffic without incurring excessive scheduling and execution overhead. As a result, software and hardware are able to operate more efficiently without compromising performance.

In effect, interrupt moderation adds an additional step to the procedure previously outlined:

1. The GbE controller successfully transmits or receives a packet.
2. The GbE controller delays delivery of the interrupt in order to transmit or receive additional packets.
3. The GbE controller generates an interrupt in response to the original event.
4. The CPU suspends its current activity in order to handle the interrupt.
5. Software examines the GbE controller to determine the cause of the interrupt. Software processes the packet event(s).
6. The CPU resumes its previous activity.

Transmit tests with an 82540EM in a dual-processor Intel® Pentium® 4 Xeon™ system running Microsoft* Windows 2000* have shown an 11% reduction in CPU utilization at wire-speed using interrupt moderation. Receive tests have demonstrated a 30% reduction in CPU utilization in this same configuration, also at wire-speed.

2.3 Trade-Offs Inherent with Interrupt Moderation

Although interrupt moderation increases the overall interrupt-processing efficiency, it also increases the average latency incurred by each packet. By delaying the delivery of an interrupt, the GbE controller is also delaying the delivery of packet information. As a result, determining optimal interrupt moderation settings usually involves a trade-off between latency and efficiency.

When network use is low, delayed interrupts are unlikely to improve performance since the rate of packet transmission or reception is relatively infrequent. In these cases, shorter interrupt delays are desirable to minimize the latency on each packet.

When network use is high, the system must operate as efficiently as possible. Every extra CPU or bus cycle spent on interrupt-processing overhead is one less cycle available for processing the actual packet data. Larger interrupt delays are desirable in these situations, to minimize interrupt-processing overhead and improve efficiency. At the same time, excessive interrupt delays might lead to resource starvation and overrun conditions. Software must negotiate between these two extremes to determine the optimal configuration for the expected workload.

Ultimately, the goal of interrupt moderation is to reduce the CPU overhead inherent in handling interrupts without adding significant packet latency or causing undue packet loss.

3.0 GbE Controller Interrupt Moderation Features

All GbE controllers contain five different mechanisms for managing the interrupt rate:

- Two absolute timers (one for transmit interrupts and the other for receive interrupts).
- Two packet timers (one for transmit interrupts and the other for receive interrupts).
- One master timer for throttling all interrupt sources.

The different mechanisms each have different strengths and weaknesses; when used together, the different timers complement each other, providing a flexible and powerful scheme for managing interrupts.

3.1 Absolute Timers

Absolute timers delay the assertion of an interrupt to enable the GbE controller to collect additional interrupt events before delivering them to software. The absolute timers are particularly useful in high traffic environments.

The receive absolute timer starts to count down upon receipt of the first packet (after software has enabled interrupts). Subsequent packets, if any, do not alter the countdown. Once the timer reaches zero, the controller generates a new interrupt. In the 82540EM/EP, 82545EM/GM and 82546EB/GB controllers, software controls the receive absolute timer using the Receive Interrupt Absolute Delay Timer (RADV) register (offset 282Ch).

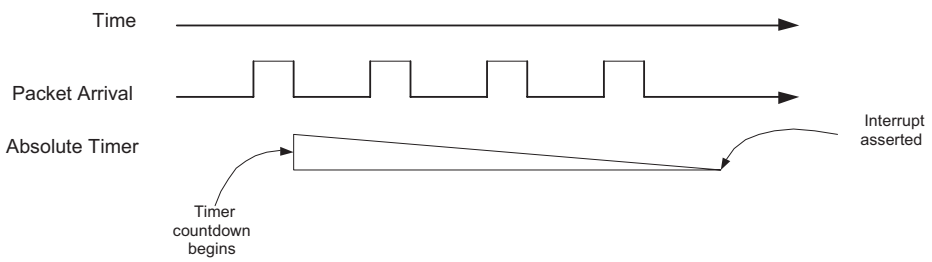


Figure 2. Receive Absolute Timer

Note: The arrival of new packets after the timer has started does not affect the countdown.

The transmit absolute timer starts to count down upon transmission of the first packet (after software has enabled interrupts). Subsequent packets, if any, do not alter the countdown. Once the timer reaches zero, the GbE controller generates a new interrupt. Software controls the transmit absolute timer using the Transmit Absolute Interrupt Delay Value (TADV) register (offset 382Ch).

The delay values are intended to be relatively large (several packet times in duration). This enables the GbE controller to transmit or receive multiple packets in succession prior to generating an interrupt.

The drawback of absolute timers is that a single packet incurs the full countdown latency even when traffic rates are low. Therefore, the absolute timers do not perform well in low traffic situations.

3.2 Packet Timers

Packet timers are inactivity timers, triggering interrupts when the link has been idle for a long interval. Software can use these timers to minimize packet latency in low traffic environments.

The receive packet timer starts to count down upon receipt of a new packet. If the GbE controller receives another packet before the timer expires, it resets the timer to its original value and restarts the countdown. If the timer reaches zero, the GbE controller generates a new interrupt. Software controls the receive packet timer using the Receive Interrupt Packet Delay Timer (RDTR) register (offset 108h).

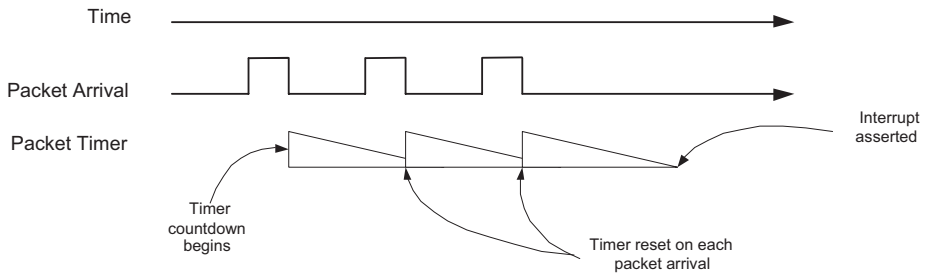


Figure 3. Receive Packet Timer

Note: The arrival of new packets after the timer has started causes the timer to restart.

The transmit timer begins to count down upon transmission of a new packet. If the GbE controller transmits another packet before the timer expires, it resets the timer to its original value and restarts the countdown. If the timer reaches zero, the GbE controller generates an interrupt. Software controls the receive packet timer using the Transmit Interrupt Delay Value (TIDV) register (offset 440h).

Unlike absolute timer delays, packet timer delays are intended to be short (possibly two or three packet times in duration). This minimizes the latency suffered by each packet.

The drawback of the packet timers is that they might be chained indefinitely. Under a sustained load, the packet timer never expires until the GbE controller has completely exhausted all of its resources. Therefore, the packet timers do not perform well in high traffic situations.

3.3 Combining the Timers

When an absolute timer and packet timer are used together, they are complementary. The absolute timers ensure that interrupts occur even when packets arrive fast enough to prevent the packet timer from ever expiring. The packet timers ensure that even in low traffic conditions, the GbE controller interrupts with relatively low latency when traffic subsides. Software can use both timers simultaneously to optimize for both types of traffic loads.

Under sustained loads, the absolute timers are the primary source of device interrupts. Figure 4 shows this process. In these situations, each packet incurs a latency of one-half of the absolute timer delay.

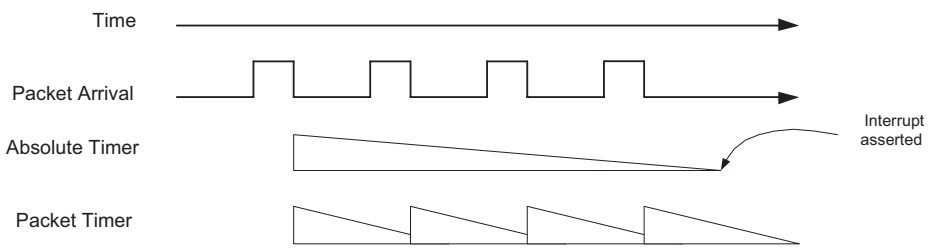


Figure 4. Absolute and Packet Timers at High Traffic Rate

Note: In these situations, the absolute timer is the source of most device interrupts.

Under light loads or brief bursts of traffic, the packet timers are the primary source of interrupts. Figure 5 shows this process. In these situations, the packet timers determine the latency suffered by most packets. The packet timers also determine the minimum traffic rate required to trigger the absolute timer interrupts. For example, if the traffic rate is high enough to prevent the packet timer from ever expiring, then the GbE controller does not interrupt until the absolute timer has expired.

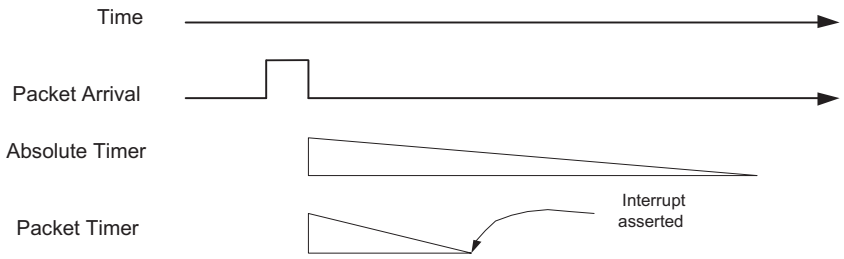


Figure 5. Absolute and Packet Timers at Low Traffic Rate

Note: In these situations, the packet timer is the source of most device interrupts.

3.4 Interrupt Throttling

A few additional factors make the process of determining optimal timer settings more challenging. First, while software can configure each pair of timers for the expected workload, the transmit and receive timers operate independently. Therefore, transmit interrupts can disrupt the idealized behavior of the receive interrupt processing and vice-versa. Second, Ethernet traffic is inherently unpredictable, exhibiting both large surges in traffic and other periods of relative inactivity. These fluctuations trigger corresponding storms and lulls in the interrupt rate. Lastly, the use of advanced GbE controller features such as TCP segmentation can introduce additional considerations by altering the usual timing of transmit events.

To limit these effects, the GbE controller also provides an interrupt throttling mechanism for placing an upper bound on the GbE controller's interrupt rate. The throttling mechanism operates independently of any interrupt source and no network events affect the throttle mechanism. Software can use the throttle timer to limit the GbE controller to a maximum interrupt rate, regardless of the traffic rate or other external factors. Software controls the throttle timer using the Interrupt Throttling Rate (ITR) register (offset C4h).

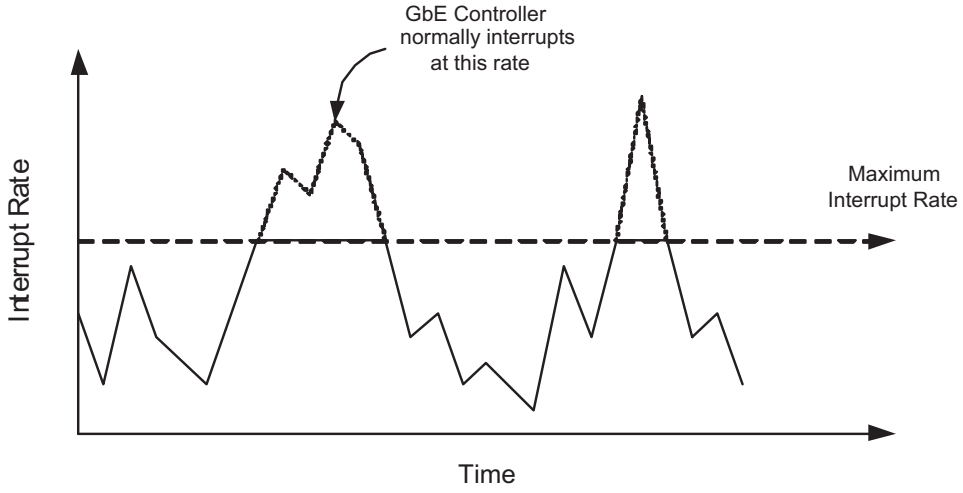


Figure 6. Effect of the Interrupt Throttle on a GbE Controller's Interrupt Rate

Note: The interrupt throttle places a ceiling on the number of interrupts per second that the GbE controller might generate.

Similar to the absolute and packet timers, the throttle timer is a simple countdown timer. The GbE controller blocks all interrupt sources until the throttle timer expires. If interrupt events are pending when the throttle timer expires, the GbE controller then generates an interrupt. When the countdown reaches zero, the throttle timer resets and restarts its countdown.

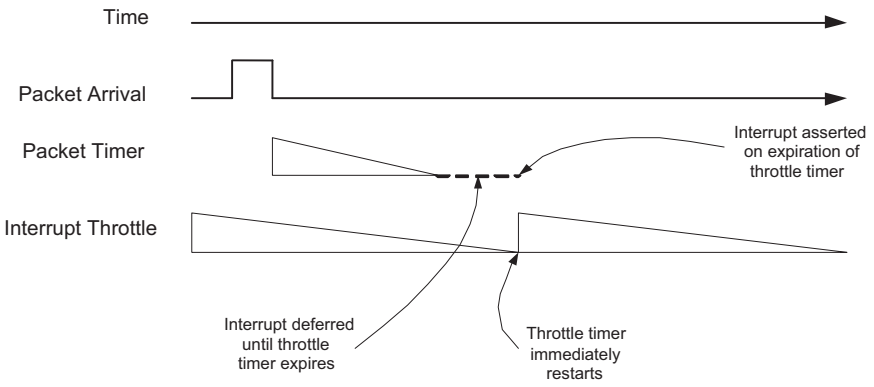


Figure 7. Interrupt Throttling Used with Packet Timer

Note: The throttle timer gates all other device interrupt sources. The throttle timer runs continuously, regardless of packet events.

On average and under light loads, the throttle timer adds half of its countdown delay to the latency of each packet. Under heavy loads, it is functionally equivalent to the absolute timers. Therefore, software typically uses the interrupt throttle as a means of restricting the GbE controller's overall interrupt rate rather than as a third interrupt-generation mechanism. For optimal performance, software might configure the throttle mechanism for a slightly higher-than-desired interrupt rate to reduce the per-packet latency previously described.

4.0 Sample Configuration

This section provides GbE controller example settings for illustration purposes only. For optimal performance, the exact GbE controller configuration is best determined through actual experimentation.

Note: It is assumed that software is optimizing for full-size frames of 1538 bytes.

The following calculations make use of the following facts:

- GbE Ethernet operates at 1.0 Gb/s or 1,000,000,000 bits per second. At this speed, the time required to transmit or receive a single bit (bit-time) is 1.0 ns.
- A full-size Ethernet frame requires 1538 bytes (12,304 bits) of bandwidth:
  - 8-byte preamble and start-of-frame delimiter
  - 14-byte Ethernet header
  - 1500-byte payload
  - 4-byte FCS
  - 12-byte inter-packet gap
- The GbE controller can transmit or receive a full-size frame every 12.3 µs or approximately 81,000 packets per second.

4.1 Absolute Timers

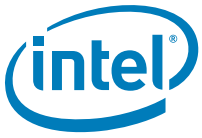
Configuring the absolute timers is typically a matter of determining the desired interrupt rate or the desired number of packets per interrupt. To receive approximately 3000 interrupts per second, software would configure the absolute timers to interrupt every 333 µs.

Alternately, to receive approximately 50 packets per interrupt, the GbE controller must interrupt approximately 1620 times per second (81,000 packets-per-second at 50 packets-per-interrupt). Software would then configure the absolute timers to interrupt every 617 µs.

4.2 Packet Timers

Testing has shown that values between 20 and 40 µs work well for the packet timers.

Software might set the packet timers to expire after two full-length packet-times, or approximately 25 µs. The packet timers would then expire when throughput falls below about 333 Mb/s (two unused packet-times follow every packet arrival; approximately one-third of the total bandwidth is in use). At greater levels of use, the packet timers would likely chain repeatedly until the one of the absolute timers expired.



4.3 Interrupt Throttle Timer

Configuring the throttle timer is determined by the desired maximum interrupt rate. As described earlier, software might realize better results by setting the throttle timer to interrupt slightly more often than desired to reduce unnecessary latencies.

For typical applications, software might configure the GbE controller to interrupt no more than 5000 times per second.

Different operating systems and environments are capable of sustaining different maximum interrupt rates. Testing has shown that Microsoft* Windows*-based operating systems perform best when the GbE controller interrupts between 4,000 and 12,000 times per second. Linux*-based operating systems perform best with an interrupt rate between 1,000 and 8,000 interrupts per second. Other operating systems might perform differently.

4.4 Additional Tuning Considerations

The previous example configuration requires modifications to suit the intended environment. The following factors might influence the tuning of the interrupt moderation parameters:

- The latency associated with scheduling an interrupt handler. Larger scheduling latencies imply larger packet latencies. Lower interrupt delays might be required in these situations to avoid overrun conditions and excessive per-packet delays.
- The cost associated with handling an interrupt. In operating systems with low-cost interrupts, higher interrupt rates might be acceptable. In operating systems with high-cost interrupts, lower interrupt rates might be required.
- The expected mixture of packet sizes. The previous sections assumed that software is optimizing for full-size frames. Optimizing for small packets or for a variety of packet sizes requires recalculating the expected packet rate.
- The expected network use. High use implies high traffic rates, which makes the GbE controller more susceptible to overrun conditions if it delays interrupts too long.

5.0 Interrupt Moderation Algorithm

Current Windows* and Linux* device drivers include an interrupt moderation algorithm that dynamically adjusts the Interrupt Throttle Rate (ITR) value based on the I/O patterns it receives.

The algorithm moderates ITR values for these I/O patterns:

- Bulk/Intermediate – large amounts of normal sized packets
- Low Latency – small amounts of traffic and/or significant percent of small packets
- Lowest Latency – minimal traffic and predominately small packets

Previous device drivers defaulted to a fixed interrupt rate of 8000 for Linux* and 4000 for Windows*, which worked for most I/O patterns but was lacking in small packet performance and latency.

This algorithm can now dynamically adjust the interrupt throttle rate for all packet sizes, which provides the flexibility to achieve lower latency, higher throughput, and lower CPU utilization.

5.1 I/O Patterns

- **Bulk** - This category is for large I/O and high throughput patterns. Throughput essentially reaches wire speed with the average packet size being close to full-size frames. This category allows for lower interrupt rates and reduced CPU utilization.
- **Bulk Intermediate** (Windows* Only) – This category is for 2 KB and 4 KB I/O sizes along with moderate throughput. This category balances CPU utilization, latency, and throughput for medium size packets.
- **Low Latency** – This category is for small packet performance and is targeted for small I/O sizes of 64 bytes to 1024 bytes. For this category, acceptable throughput can be achieved (about the same throughput with ITR turned off).
- **Lowest Latency** – This category is for small ping-pong type messages. Other I/O patterns can transition to this mode when the number of packets are small and the number of total bytes processed are small.

5.2 Default Settings/Interrupt Rates

These settings are suitable for most applications especially when variable I/O patterns are likely to occur.

I/O Patterns	Windows* (Adaptive Mode)	Linux* (Mode 3)
Bulk	3,000	4,000
Intermediate	10,000	Ignored/Not Supported
Low Latency	20,000	20,000
Lowest Latency	No Limit (ITR 0)	Use Mode 1 or Fixed Value*

While in either of these two modes, the current device drivers provide dynamic transitions from bulk/intermediate to low latency (when throughput starts to drop) and to lowest latency when the number of packets are small and the number of total bytes processed are small.

5.3 Manual Settings (Windows*)

Following are fixed Interrupt rates for all I/O patterns.

Select one of these settings from the Advanced tab in the Server Adapter dialog box. For example, if using a single application with a constant I/O pattern.

Note: To achieve the optimum interrupt rate for a specific I/O pattern, experimentation is required.

Value	Interrupt Rates	Latency	Throughput	CPU Utilization
Extreme	1,000	Highest	High	Lowest
High	1,900	Higher	High	Lower
Medium	4,000	High	High	Low

Value	Interrupt Rates	Latency	Throughput	CPU Utilization
Low	10,000	Low	High	High
Minimal	19,000	Lower	High	Higher
Off	No Limit	Lowest	Varies	Highest

Note: Adaptive is the default setting and should be used when variable I/O patterns are likely to occur.

5.4 Manual Settings (Linux*)

Loaded from the command line as:

- `insmod e1000.ko InterruptThrottleRate=1` (Mode 1)
- `insmod e1000.ko InterruptThrottleRate=0` (ITR off)
- `insmod e1000.ko InterruptThrottleRate=8000` (Fixed value for all I/O patterns)

Mode/Value	Interrupt Rates	Latency	Throughput	CPU Utilization
3	4,000 to 20,000	High to Low	High	Low to High
1[1]	4,000 to 70,000	High to Low	High	Low to High
0	No Limit	Lowest	Varies	Highest
Fixed Value	100 to 100,000	High to Low	Varies	Low to High

1. Use Mode 1 when low latency is a must and when variable I/O patterns are likely to occur.

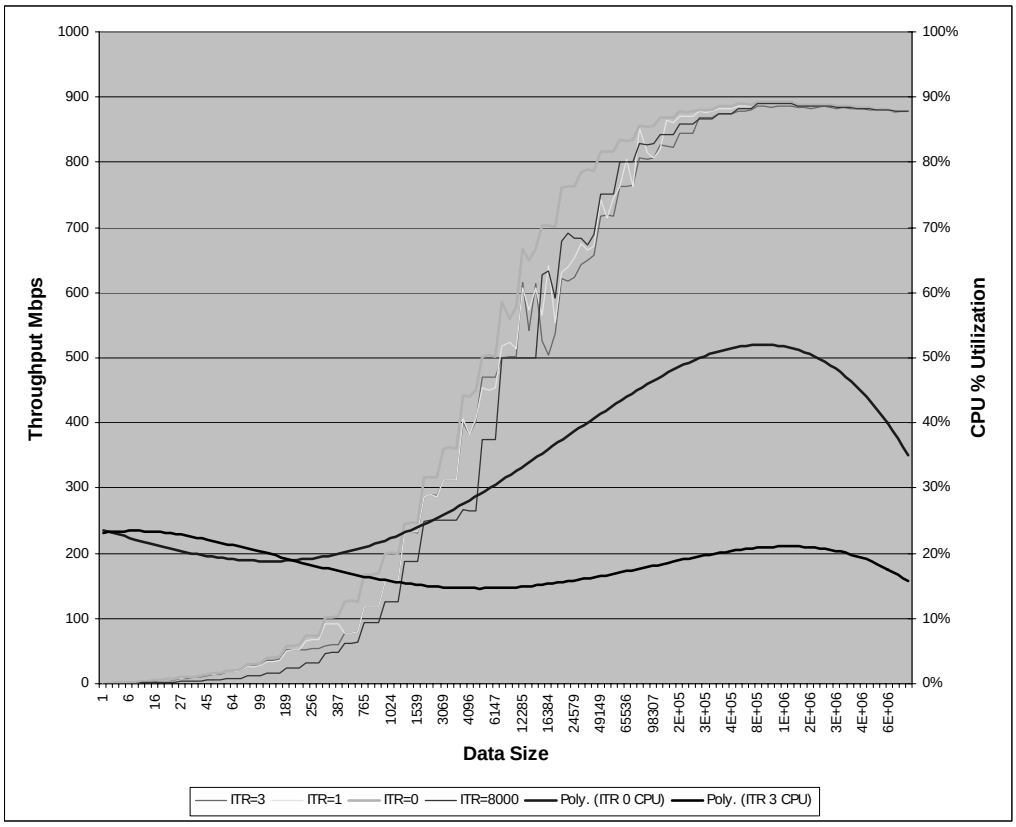
- For dual-port GbE controllers, each port can be set independently. For example, one port can be set to adaptive (Mode 3 or Mode 1) and the other to a fixed interrupt rate.
- If using multiple NICs with multiple ports, each NIC or NIC port can also be set independently. For example, a single-port NIC can be set to adaptive (Mode 3 or Mode 1) and other NICs/NIC ports can be set to fixed interrupt rates.

Note: For multiple network ports, ITR must be set for each port (with comma separated values):

```
insmod e1000.ko InterruptThrottleRate=1,1,1,1
```

5.5 Linux* ITR Throughput Comparisons

- ITR-3 (New Default)
- ITR-8000 (Old Default)



Note: Smoothing algorithm applied to ITR-0/ITR-3 CPU values.

5.6 Linux* ITR Latency Comparisons

- ITR-3 (New Default)
- ITR-8000 (Old Default)