NOVEMBER 1993

NOVELL® RESEARCH

Packet Burst Update: BNETX vs. VLM Implementations

Myron Mosbarger
Senior Consultant
Systems Research Department

Drex Dixon
Systems Software Engineer
NetWare Systems Group

Novell's new implementation of packet burst in the DOS Requester (VLMs) offers several advantages over traditional packet windowing implementations and over Novell's previous packet burst shell (BNETX). This AppNote compares the old and new implementations, focusing on the architectural changes that have been made. By understanding the use of PB buffers and how each version uses window size and interpacket gap for dynamic self-adjustment, network administrators will have the conceptual foundation for installing or updating packet burst on their own network.

Contents

Introduction

Two major concerns as local area networks have grown into metropolitan and wide area networks (WANs) are bandwidth utilization and response time. Novell developed packet burst technology as an enhancement to its NetWare Core Protocol (NCP), primarily to increase available bandwidth and reduce response time on busy network segments and over low-speed links.

The first implementation of packet burst consisted of an add-on module (PBURST.NLM) for the NetWare 3.11 operating system and a burst mode version of the NETX shell (BNETX.EXE). With the release of NetWare 4.x and 3.12, packet burst has been redesigned and integrated as a feature in the core products. The workstation packet burst capabilities are integrated into the new VLMs (Virtual Loadable Modules) and are enabled in the default workstation configuration.

The differences between BNETX and the VLM implementations of packet burst are significant. This AppNote focuses on the changes in Novell's packet burst implementation and identifies the effects they have on network throughput.

Brief Review of Packet Burst

NetWare's original service protocol, NCP, was designed and optimized for use over high-speed LAN links (2.5Mbps and faster). The one-to-one request and response ratio imposed by NCP did not present a problem for local area communication. As NetWare users began communicating over slower wide area links, some of the limitations of this request/response dialogue were accentuated.

The Old Implementation (BNETX)

To improve throughput during large file transfers, Novell's initial implementation of packet burst modified the number of packets sent in response to a request. This allowed a many-to-one ratio rather than a one-to-one ratio when a client was reading or writing large files (over 64KB). The process was dynamic and improved throughput during file transfer by increasing the number of packets that could be sent before receiving an acknowledgement.

This required modifications to the workstation shell (NETX), which culminated in the creation of a new shell called BNETX and a new NetWare Loadable Module (NLM) called PBURST.NLM. Generally, users on wide area networks (typically slower links) stood to gain the most from using BNETX. Implementing an optimal configuration with BNETX required manual modification of the workstation's NET.CFG file.

For in depth information regarding Novell's packet burst shell, see An Introduction to Burst Mode Technology in the March 1992 NetWare Application Notes.

The New Implementation (VLM)

While the first iteration of packet burst was successful in many installations, its ability to dynamically configure and adjust to network conditions was limited.

In the current implementation of Novell's DOS Requester, packet burst is automatically enabled at the workstation in the FIO Virtual Loadable Module (VLM). At the server, packet burst is an integral component of the NetWare 4.x operating system. No special code is required. NetWare 3.12 also has packet burst capabilities integrated into the operating system.

Note: If workstations will be running the new VLMs and connecting to file servers running NetWare 3.11, the PBURST.NLM must be installed on those servers to take advantage of the new VLM packet burst features.

Key Concepts

Before proceeding with our comparison of the BNETX and VLM packet burst implementations, it is helpful to have some basic understanding of buffering, windowing, and interpacket gap. These are key to understanding the major architectural differences between traditional windowing protocols, BNETX, and the VLMs.

Buffering

To understand how Novell's packet burst VLM works, we must first understand where and how buffering is used in NetWare communications. There are two kinds of buffers which affect throughput:

- The network interface card hardware buffers, which are responsible for taking packets off the network and passing them to the receiving device.
- Workstation (shell) and router buffers, which are responsible for copying or transferring packets, or their contents, to memory.

As we will see, hardware and software buffers can significantly limit the amount of data a workstation or router can handle.

Windowing

Traditional windowing implementations focus on sending groups of packets back-to-back in what is called a window. Once all of the packets in the window have arrived at the destination, a single acknowledgment packet is returned (see Figure 1).

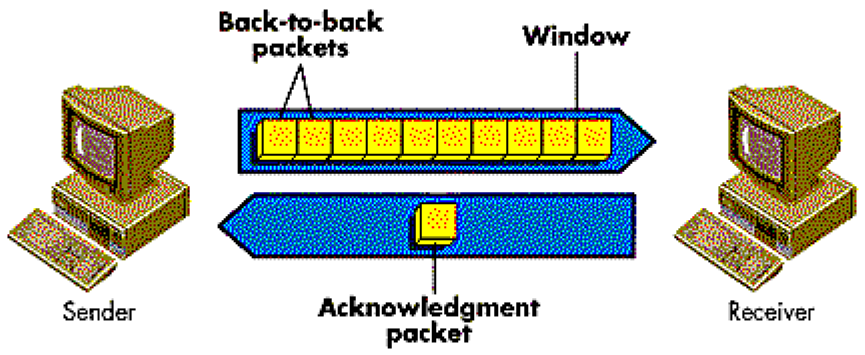

Figure 1: In traditional windowing protocols, a group of packets is sent with a single acknowledgement packet.

Provisions are made in case not all of the packets in the window arrive intact. For example, if 10 packets were requested, the receiving station tracks the packets it receives by sequence number. If any packets are not received within the timeout period, the receiving station requests a re-send. The re-send begins with the lost packet (packet #5 in the example shown in Figure 2) and includes all packets in sequence to the end.

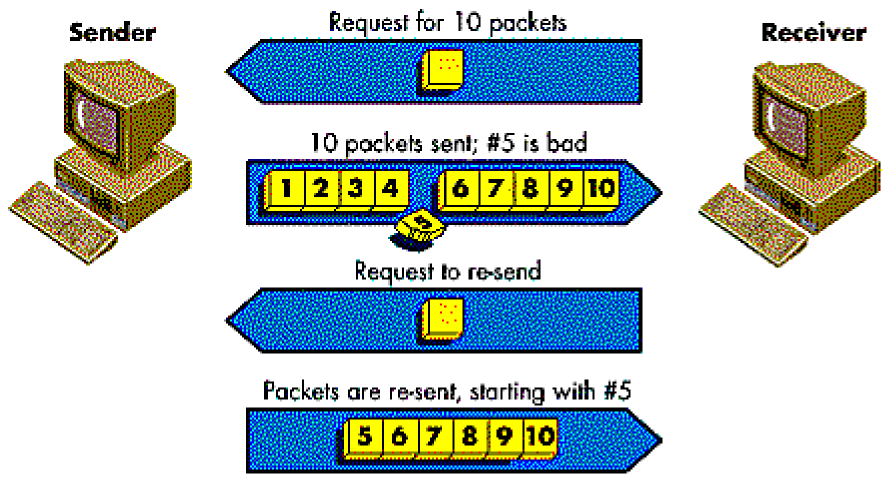

Figure 2: When packets are dropped, the packets are re-sent starting with the one that was bad.

In traditional windowing implementations, adjusting the window size is the primary means of tuning throughput. This tuning process is illustrated in Figure 3.

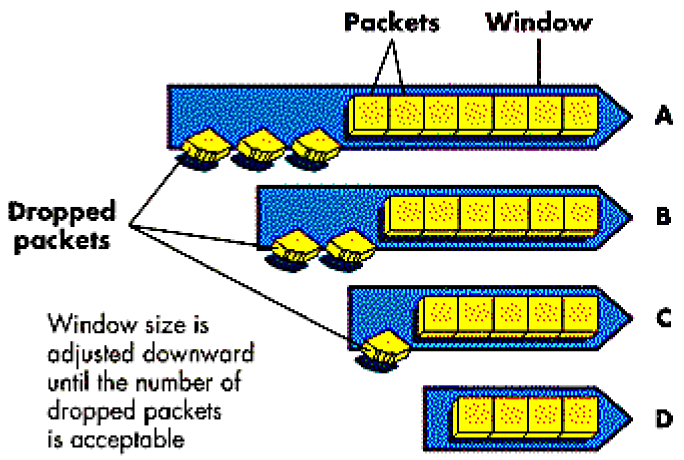
Packets
Window
A
Dropped packets
B
C
Window size is adjusted downward until the number of dropped packets is acceptable
D

Figure 3: When packets are dropped, the window size is reduced in traditional windowing implementations.

A small number of dropped or missing packets is to be expected due to varying network traffic conditions. But when a specified number of packets is dropped in so many windows in a row, the window size is adjusted downward. Decreasing the window size reduces the probability of dropped packets, which generally makes the file transfer process more efficient because there aren't as many re-sends.

However, as the window size becomes smaller and smaller it can also reduce the effectiveness of the windowing protocol. After all, the whole point is to be able to send a group of packets instead of a single packet for each acknowledgement.

Interpacket Gap (IPG)

Interpacket gap is the time that elapses between the end of one packet and the beginning of the next, as illustrated in Figure 4. Adjusting the IPG (or packet metering, as it is sometimes called) is a technique not commonly used in traditional windowing protocols. As explained above, traditional windowing implementations focus on adjusting the window size as the sole means of managing throughput. Novell's latest packet burst implementation adjusts the IPG as the primary variable for increasing throughput, then adjusts the window size as the secondary variable.

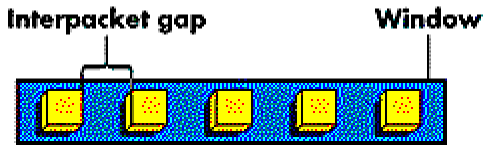

Figure 4: The interpacket gap is the time between the end of one packet in the window and the beginning of the next.

Adjusting the IPG is a technique used in Novell's VLM implementation of packet burst. It provides greater throughput than adjusting the window size because the number of packets per window remains the same. By adjusting the IPG, packets flow through the workstation in larger windows and with no retries.

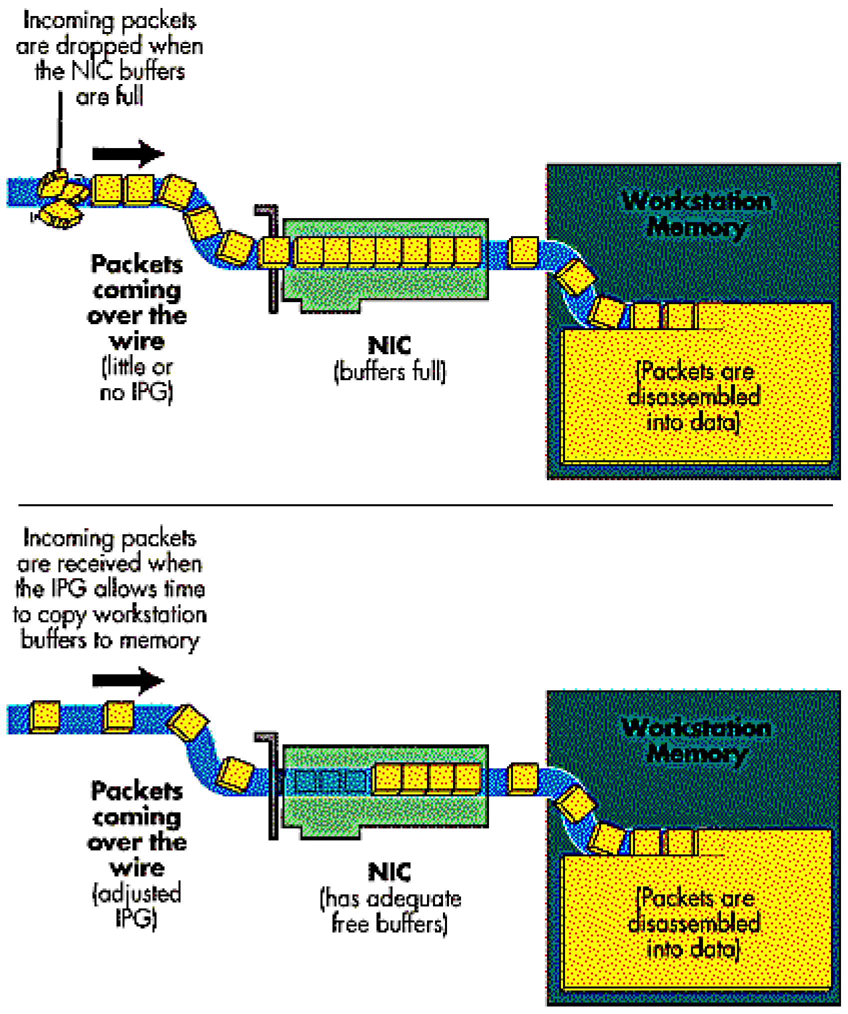
Incoming packets are dropped when the NIC buffers are full
Packets coming over the wire (little or no IPG)
NIC (buffers full)
Workstation Memory
(Packets are disassembled into data)
Incoming packets are received when the IPG allows time to copy workstation buffers to memory
Packets coming over the wire (adjusted IPG)
NIC (has adequate free buffers)
Workstation Memory
(Packets are disassembled into data)

Figure 5: Adjusting the IPG controls the flow of packets to match the ability of the receiving NIC to handle them.

Optimally, the packet receive rate is equal to the workstation's capacity to receive and distribute the data with no dropped packets. The VLMs dynamically adjust and optimize the IPG to overcome the limitations of hardware processing speed and hardware buffering, as shown in Figure 5.

Comparison of Packet Burst in BNETX and VLMs

There are three major differences between packet burst in BNETX and in the VLMs. They are:

- Their use of workstation buffers (PB Buffers)
- Their use of window size and interpacket gap as tuning variables
- Their ability to self-configure and handle congestion

PB Buffers

PB Buffers in BNETX. To configure the number of buffers the shell can use for packet burst, workstations running BNETX require a modification to the NET.CFG file, similar to the following:

PB Buffers = 4 (or some other decimal value)

This value determines the amount of memory that will be claimed by the shell for packet burst buffers. The formula for calculating the memory requirement associated with BNETX is:

Memory required by shell = PB Buffers X Packet Size

For example, if the packet size is 1500 bytes (as it is for Ethernet), the amount of additional memory allocated to the workstation shell to run BNETX with four buffers is 6000 bytes (4 x 1500 = 6000). For Token Ring (with 4096-byte packets), the same configuration would require 16,384 bytes of memory.

Note that this is non-returnable memory. Once it has been pre-allocated to the shell for packet burst buffers, the memory is no longer available for any other use.

A number of other factors come into play here. For one, the NetWare shell has a 64KB segment limit, which makes it possible to specify a PB Buffers value greater than the shell can accommodate. For example, if the current shell size is 43KB, the maximum available memory left for buffers would be 21KB (64KB - 43KB). As shown in Figure 6, this much memory could accommodate fourteen 1500-byte Ethernet buffers or five 4096-byte Token Ring packets.

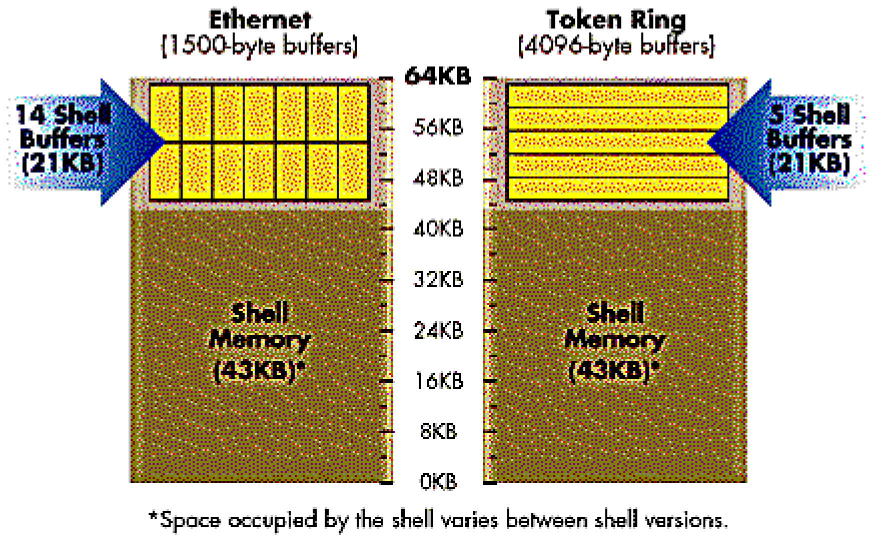

Figure 6: The 64KB segment limit of the BNETX shell limits the amount of buffer space available.

With BNETX, performance generally improves as the number of buffers increases. However, there is a point after which adding more buffers yields only nominal performance benefits. For example, experience has shown that a PB Buffer value of 10 for 4096-byte Token Ring packets provides no more performance benefit than a value of 5 would.

Another weakness with BNETX is its inability to dynamically configure the number of buffers needed. Arriving at an optimized configuration requires user intervention on each workstation using packet burst. To discover the optimal number of buffers, a user must experiment. Without the aid of network analyzers, this is mostly a trial-and-error process. Once the proper number of buffers is determined, you must examine the workstation to see if it has enough memory to support the packet burst configuration along with the other applications running at the workstation.

In addition to its stringent memory requirements, the way BNETX uses buffering requires two moves or copies to transfer data into workstation memory (see Figure 7).

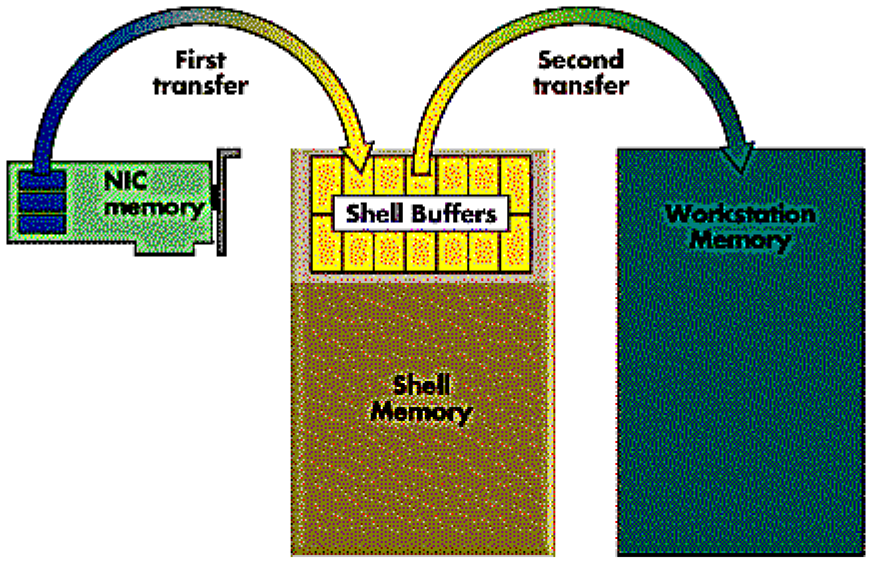

Figure 7: BNETX requires two separate transfers or copies to get data from the NIC memory into workstation memory.

The latency factor as packets are first transferred into buffer memory and then into workstation memory limits the overall speed at which packets could be received. The net result is a decrease in actual throughput.

PB Buffers in VLMs. In Novell's VLM implementation, the workstation packet burst feature is contained in the FIO.VLM module. (FIO stands for File Input/Output.) Like BNETX, FIO.VLM uses the PB Buffers = statement in NET.CFG, which is set by default to a value of 3. Unlike BNETX, the VLM does not use this value to set aside buffer space. Rather, the value either enables (non-zero integer) or disables (zero) the packet burst feature at the workstation.

For example, if packet burst feature will not be used, a small amount of memory can be reclaimed by adding the following command to the NET.CFG file:

PB Buffers = 0

This command disables packet burst at that workstation and reduces the DOS Requester's memory requirement slightly.

Another difference is that the FIO.VLM does not buffer read packets into an intermediate buffer pool as BNETX did. It requires only one copy to transfer memory from the NIC to the workstation (see Figure 8).

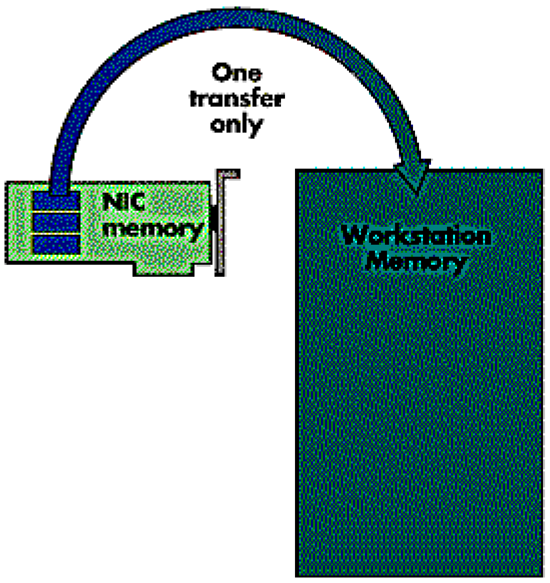

Figure 8: The VLM implementation requires only one transfer to get data from NIC memory into workstation memory.

This major architectural change provides several benefits over BNETX. The most significant advantage is the VLM's ability to dynamically configure buffering at each workstation--no user intervention or NET.CFG modification is required. In addition, the VLM dynamically tunes itself for optimal performance and improves throughput by eliminating the latency effects associated with the addition of intermediate buffers.

Window Size and Interpacket Gap

Window Size in BNETX. BNETX is basically a traditional implementation of a windowing protocol over

IPX. It adjusts the window size as its primary means of flow control. As shown in Figure 9, packets are sent in windows of a predetermined size. If packets are dropped, the window size is adjusted downward to accommodate either network traffic or the destination device's capacity to receive packets. With BNETX, the IPG remains relatively constant and does not generally improve performance.

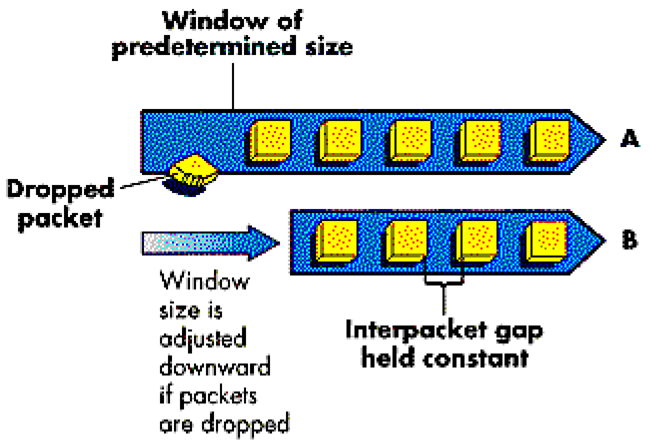
Window of predetermined size
A
Dropped packet
B
Window size is adjusted downward if packets are dropped
Interpacket gap held constant

Figure 9: The BNETX implementation adjusts the window size as its primary means of flow control.

Determining the appropriate window size is crucial to the tuning process. In Novell's BNETX implementation, the initial window size is determined by the shell, the PB Buffers parameter, and the packet size--with a maximum window size of 64KB.

BNETX requires the number of workstation buffers to be set by editing the NET.CFG file. The workstation then configures its shell memory buffers when NET.CFG is run, as described above. For that reason, BNETX will never request a window size larger than its buffering capacity.

It is important to understand that the shell or memory buffering capacity of the receiving device is independent from the hardware buffering capabilities of the network interface card. Therefore, setting the buffer size higher than the NIC can handle will in effect waste memory. In BNETX, the maximum buffering capabilities of a workstation is the weakest buffering component.

Window Size in the VLMs. Version 1.02 and earlier versions of the VLMs do not adjust the window size, only the IPG. VLMs version 1.03 and later do adjust the window size, but only after adjusting the IPG to its maximum value first.

For VLM versions 1.02 and later, window sizes are determined as follows.

By design, the VLM implementation has no window size limitation as BNETX does (this has been successfully tested and confirmed). However, to accommodate the capabilities of the majority of network devices, the values of 16 packets for a read and 10 packets for a write have been set as defaults. The maximum window size is calculated as follows:

Maximum number of reads (16) x the frame size
Maximum number of writes (10) x the frame size

To illustrate how these imposed defaults work, suppose a workstation attached locally on a network segment wants to read a 56KB file. On Ethernet, the workstation will request two windows (16 packets) of 24KB and one window (6 packets) of 8KB. On Token Ring with 4KB packets, the workstation would have one window (14 packets) of 56KB (see Figure 10).

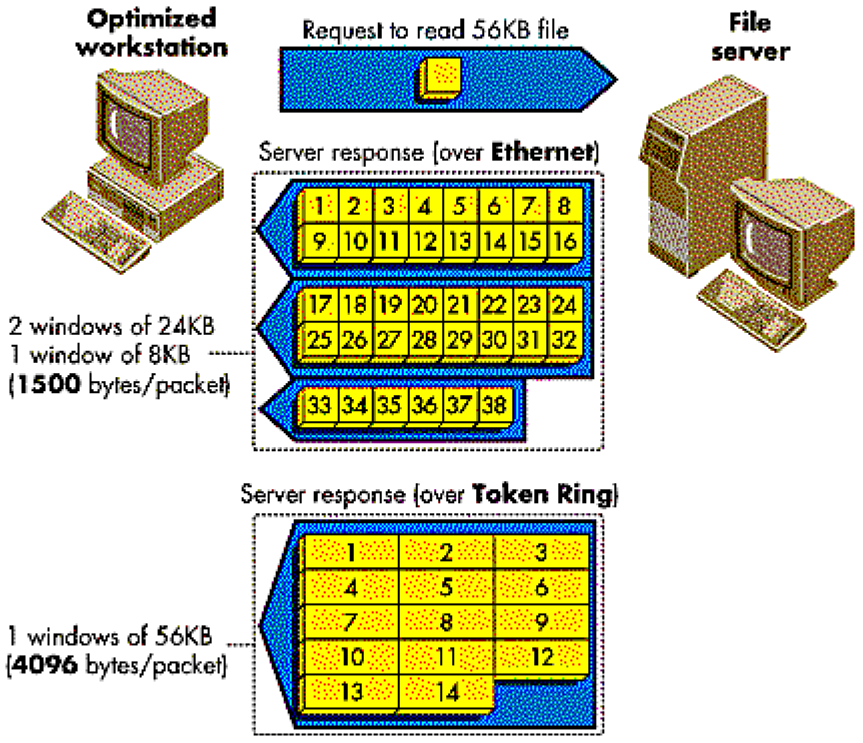

Figure 10: Imposed defaults for maximum window size determine how many packets can be sent in a burst.

Window sizes of 16 for reads and 10 for writes were selected for two reasons. First, they provide a greater reliability in a wide variety of installations. On wide area networks or slower links (which are generally more prone to loss than LAN links), larger window sizes tend to be less reliable. Second, as the number of packets in a transfer increases, the percentage of improvement increases substantially within the first 10 to 16 packets. Beyond that, performance improvements are minimal (see Figure 11).

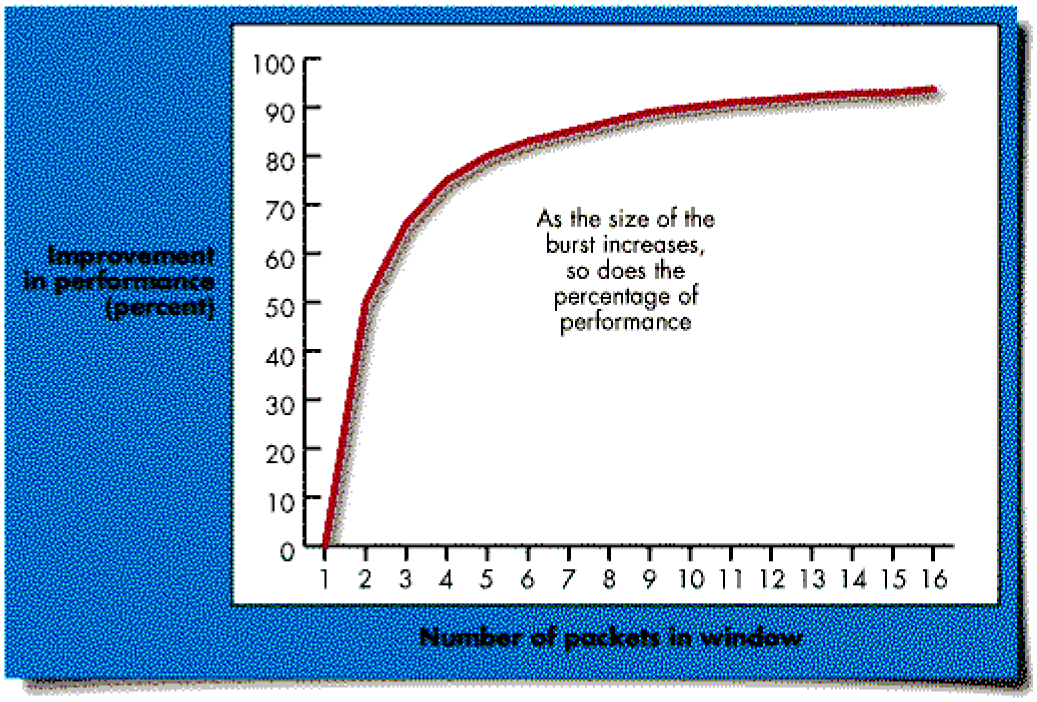

Figure 11: As the size of the burst increases, so does the percentage of performance improvement.

Starting with VLM version 1.02, two new configuration parameters have been added to allow users to modify the packet burst window size:

```
PBURST READ WINDOW SIZE  = n   (3 to 255)
        PBURST WRITE WINDOW SIZE = n   (3 to 255)
```

When entered in the NET.CFG file, these commands allow a user to override the defaults. For users on a locally-attached segment whose network and server have sufficient capacity, increasing the number of packets in a read or write will improve performance even more.

Adjusting the Window Size. The window size is adjusted exponentially. As an example, suppose a window size of 16 packets with a maximum packet size of 1KB reaches it maximum IPG and the percentage of unsuccessful bursts is unacceptable. The window size will be decreased according to the following formula:

NewWinSz = CurWinSz - ((CurWinSz - 3 x MaxPcktSz) / 4)

In our example, the new window size would become 12KB. The calculation is as follows:

16KB - ((16KB - 3 x 1KB) / 4) = 12KB

If a 12KB window size was also found to be unacceptable, the size would be reduced to 10KB, and so on.

Tuning the Interpacket Gap

Unlike traditional windowing implementations, the VLMs substantially improve throughput by tuning the IPG. This technique is unique to Novell's VLM packet burst implementation.

The Optimum Delivery Rate. Conceptually, the optimal window delivery rate sends data at the same rate the slowest device along the packet delivery path can process it, without dropping any packets. This would create a smooth flow of data with the least amount of overhead (timeouts, re-sends, and so on). Novell has found that by adjusting the interpacket gap, very large windows of data can be sent successfully.

Determining the appropriate interpacket gap is the key to this process. Ideally, when the gap is optimally tuned, a workstation can read or write a window of any size.

Determining the IPG in VLM 1.02 and Earlier. Since VLM version 1.02 adjusts only the IPG, not the window size, it is important to determine an appropriate maximum IPG or ceiling for a window. To do so, the workstation sends a series of pings (messages that go to the destination and back again). Half the time value of fastest round trip becomes the maximum interpacket gap (see Figure 12).

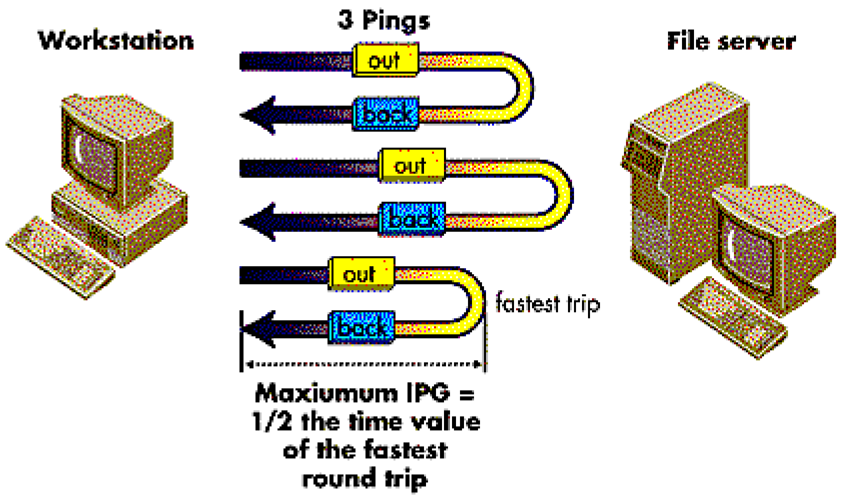

Figure 12: To arrive at the maximum IPG, the workstation takes half of the fastest round-trip ping time.

With this maximum value established, the requesting device begins the file transfer by using an IPG of 0 (zero). When a predetermined number of failures occur (for example, two packets dropped in six bursts), the interpacket gap is automatically increased until either no packets are dropped or the maximum IPG is reached.

As an example, suppose a workstation determined a maximum IPG value of 5000. The initial IPG is set to zero. If this is unsuccessful, the IPG is increased to 3000. If this is successful, it remains unchanged. This process is illustrated in Figure 13.

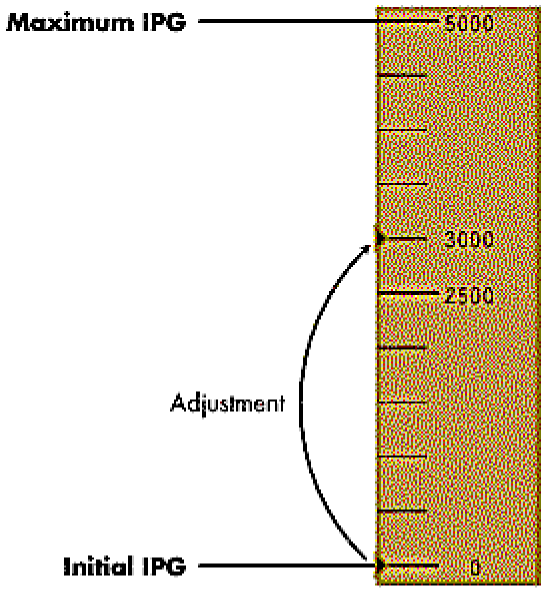

Figure 13: The process of adjusting the IPG in VLM versions 1.02 and earlier.

The ceiling value is large enough to handle nearly all network and workstation bottlenecks. Since the VLMs pass data directly to memory on the workstation, the only buffer limitations are those of the interface card (or in the case of a router, both hardware and receive buffers).

Remember, for VLMs version 1.02 and earlier, only the IPG is changed, not the window size.

Determining the IPG in VLM 1.03 and Later. In the latest version of the VLMs, the initial IPG is set to half of the maximum IPG and uses a binary algorithm to reach the optimum IPG. This represents two changes:

- First, by setting the initial IPG at half the maximum rather than zero, the number of re-sends is reduced for the majority of today's workstations.
- Second, the binary algorithm for tuning the IPG is substantially faster than the method used in VLM 1.02, and allows for fine tuning the gap time to maximize performance.

These two changes can improve login time for packet burst-enabled workstations.

Figure 14 illustrates how the IPG tuning method works for VLM versions 1.03 and later.

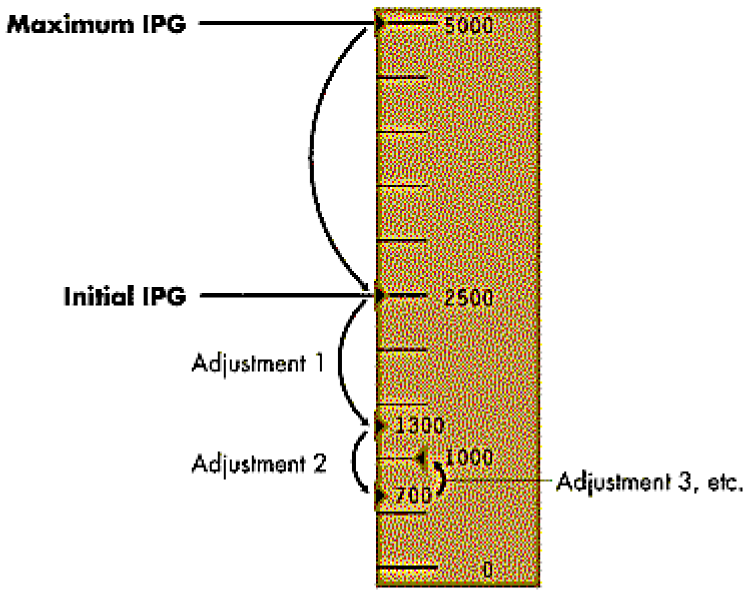

Figure 14: The IPG tuning method for VLM versions 1.03 and later.

In this example, if the maximum IPG is 5000 microseconds, the initial IPG would be set at 2500. If this were successful, the IPG would be decreased to 1300 microseconds. If that were successful, the IPG would be reduced to 700 microseconds. If that were unsuccessful, the IPG would be increased to 1000 microseconds. This process would continue until a baseline IPG is established.

Note: When decreasing the IPG, the algorithm rounds up (1250 becomes 1300). When increasing the IPG, the algorithm rounds down (1250 becomes 1200).

The method used in version 1.03 will be faster for the majority of client workstations and will tune all machines faster than the 1.02 method.

Pings Over Satellite Links. When dealing with satellite or network links with inherent delays, the original ping process produced an unrealistic value. To accommodate this type of link, the ping process was modified to determine an approximate available bandwidth. This is accomplished by sending three pairs of pings across the network link. The time between the end of the first packet and the end of the second packet in each pair is computed and saved. The average of all three pings is computed and becomes the maximum IPG for the workstation. Based on this value, the IPG is set to half the value for the first transfer and is then tuned for optimal throughput (see Figure 15).

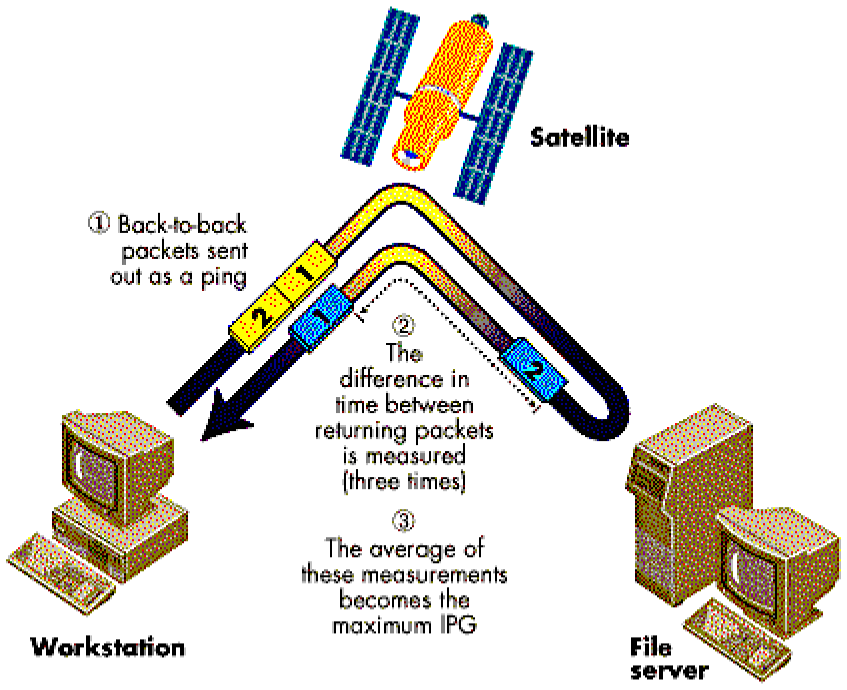
Satellite
① Back-to-back packets sent out as a ping
② The difference in time between returning packets is measured (three times)
③ The average of these measurements becomes the maximum IPG
Workstation
File server
2
1
1
2

Figure 15: Modified process for determining IPG over satellite links with inherent delays.

This method provides a good approximation of the available bandwidth across the route.

Congestion

A potential problem with high volume, bursty traffic is bandwidth availability and congestion. Traditional packet burst implementations can add additional traffic. For example, if a burst of packets begins transferring and packets are dropped (due to increased traffic) during the transfer, retry algorithms tend to increase congestion with retry packets. This creates additional traffic on an already busy network. The effect of this problem as seen by the users could be lost network connections, slow response, or other related timeout problems.

When packets are dropped, it is the function of the client or workstation software to initiate retries. Client software that is unable to dynamically adjust or back off the number of packets and retries in high traffic conditions tends to worsen the problem by increasing the number of packets sent, which was the cause of the problem initially.

How BNETX Handles Congestion. BNETX adjusts to network congestion in the traditional manner by adjusting the window size. The window size is temporarily reduced and will eventually return to the original baseline size. The smaller the window, the more windows are needed to complete a transfer and with each window an acknowledge packet is sent. This means that packet burst becomes less effective in high traffic situations.

How the VLMs Handle Congestion. The VLM deals with congestion by increasing the IPG. The VLM checks for network congestion by monitoring and tracking the number packets lost during a transfer.

For example, if one packet is lost in six windows, the VLM makes no corrections. However, if a packet is dropped in consecutive windows, the VLM would increase the IPG for the next several windows. This process will continue until the number of packets dropped is within an acceptable limit (see Figure 16).

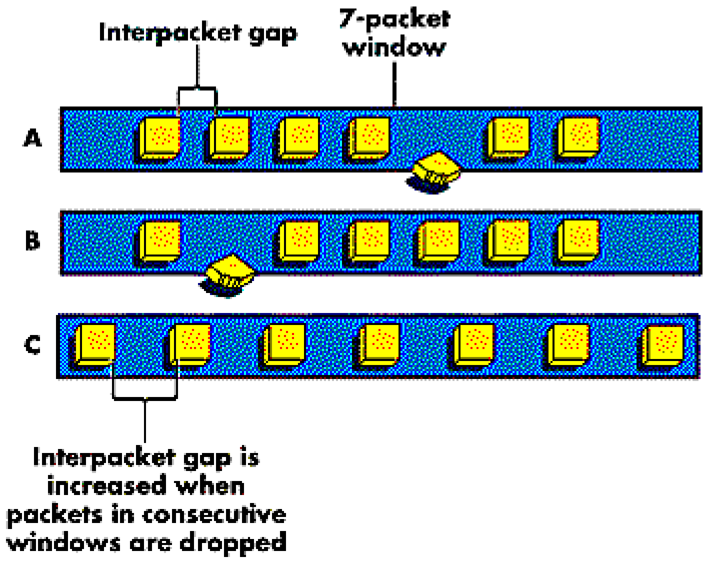
Interpacket gap
7-packet window
A
B
C
Interpacket gap is increased when packets in consecutive windows are dropped

Figure 16: The VLM handles network congestion by increasing the IPG.

Remember, the retry timeout and IPG values are based on information received about the network's capacity prior to the file transfer.

The primary difference between adjusting the window size and adjusting the IPG is that the number of acknowledgements does not increase as traffic increases. This means that when windows of 24KB are being sent, there is no loss in the efficiency of the packet burst protocol even though the packets arrive at a slower rate. In contrast, there is a loss of efficiency when adjusting the window size.

Conclusion

In NetWare 3.12 and 4.x, the FIO.VLM contains Novell's new implementation of packet burst for workstations. This implementation offers several advantages over typical packet burst implementations and over Novell's previous version (BNETX). While VLM versions 1.03 and later have been significantly improved for locally-attached users, they have been modified specifically to provide increased performance for low-speed links and links with delays over VSAT (Very Small Aperture Terminal).

The percentages shown in Figure 17 are representative of the performance benefits that come from using BNETX or the VLMs over NETX. Keep in mind that network interface cards, drivers, workstations, and servers all influence total throughput. Our testing was mainly conducted to show the effect of the tuning algorithms for the VLMs compared to BNETX.

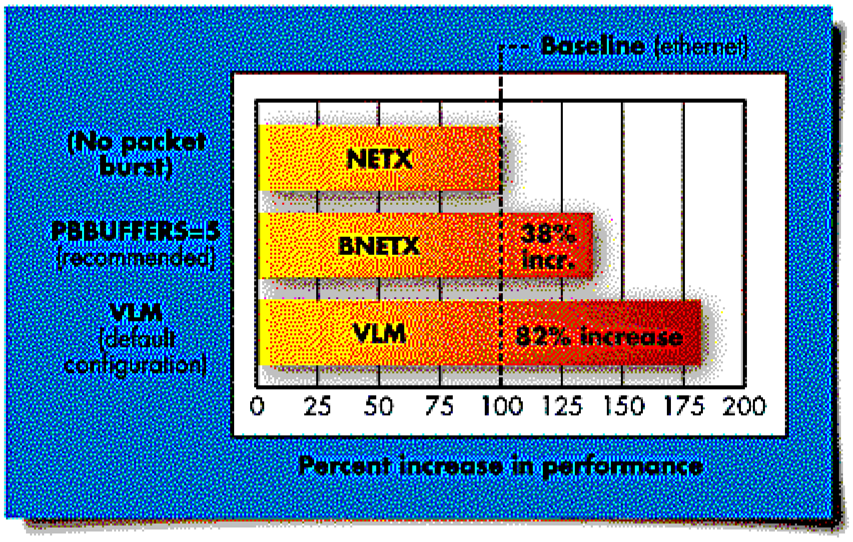

Figure 17: Percentage of performance increase for BNETX and the VLMs as compared to the NETX shell.

Sites which are doing large file transfers or which are currently using BNETX will benefit substantially by installing version 1.03 of the VLMs.