



## *How does it work?*

Applications that use TCP/IP as a data delivery protocol are subject to the performance limitations that are inherent in the protocol. This especially comes into play when there are large amounts of data to be sent (e.g. storage replication or DR/BC applications); over a wide area network (e.g. anything beyond the LAN); and there is at least some level of packet loss that periodically occurs. Over the years, there have been various TCP tuning enhancements made available to help mitigate performance issues due to packet loss and latency, but the fact remains TCP was designed to handle traffic on a LAN, not a WAN. Specifically, TCP interprets all packet loss as congestion, which is the correct interpretation for a LAN, but not necessarily the correct interpretation for a WAN. For example, when running over a WAN, packets are also likely to be discarded due to corruption.

HyperIP *transparently* provides performance benefits to TCP applications, especially in situations that involve packet loss and/or latency. The goal of this paper is to provide the reader with a better understanding of network issues that affect TCP performance, and what HyperIP does to specifically mitigate those issues.

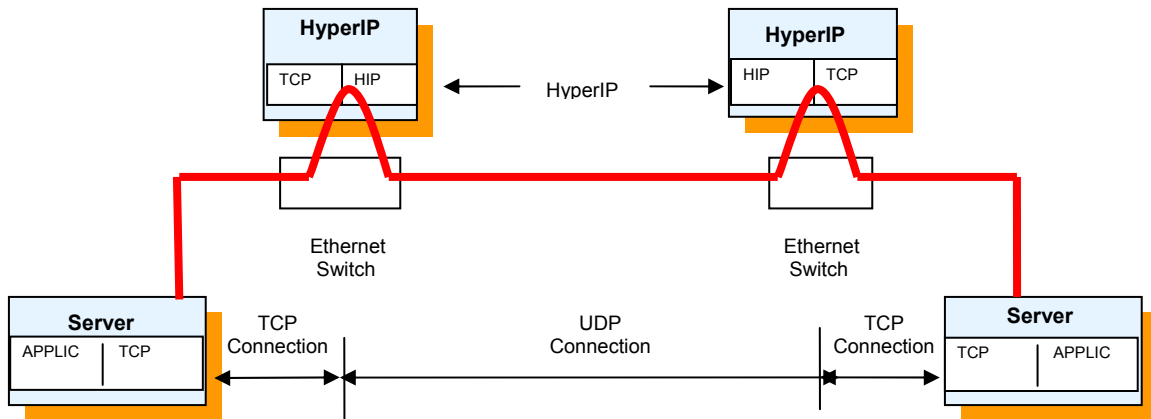
There are four key elements that contribute to the performance enhancing capability of HyperIP:

- 1) Local TCP acknowledgements
- 2) Efficient bandwidth utilization
- 3) Mitigating the effect of packet loss
- 4) Mitigating the effect of latency

Each of these elements is discussed in the following pages.

### **Local TCP acknowledgements**

Each of the HyperIP appliances serves as the endpoint of the TCP connection to the application server on the LAN segment (Figure 1).



**Figure 1. HyperIP network connections**

An independent connection is maintained over the WAN between the HyperIP appliances. The flow of data from the application is governed by the generation of TCP acknowledgements from the local HyperIP to the local application server or storage controller. These acknowledgements keep the TCP windows open, so data can continue to be sent by the application. HyperIP shields the application's TCP connection from performance variations due to packet loss and latency on the WAN, since the performance over the WAN is managed by HyperIP.

## Efficient bandwidth utilization

### *TCP bandwidth sharing*

When individual TCP applications compete for the same bandwidth, congestion is created. As new TCP applications are started, this new workload creates additional demand for the same bandwidth, which creates more congestion. The "bandwidth sharing" technique available in TCP is for each application's TCP connection to attempt to send data faster and faster, until packet loss occurs. When that happens, performance degrades, potentially for each application, due to the TCP slow start and congestion avoidance algorithms. Conversely, as TCP applications are stopped, the workload is decreased, making more bandwidth available. However, the additional bandwidth capacity is not immediately claimed by any of the remaining connections. It is up to each remaining connection to probe the network for additional capacity, which typically happens very slowly.

### *HyperIP bandwidth sharing*

HyperIP serves as the focal point for efficiently managing packet streams over the WAN. When it is the sole driver of the WAN, it has knowledge of the WAN capacity and sustainable link rate.

HyperIP manages multiple LAN packet streams, and aggregates them over the HyperIP network. As new TCP application connections are started, HyperIP is able to

accommodate the additional workload by inserting the new packet stream into the HyperIP connection without creating congestion. As TCP applications are stopped, the additional bandwidth capacity is automatically reclaimed by HyperIP for sharing among the remaining connections.

The HyperIP protocol dynamically adjusts the rate control, latency time, and bandwidth capacity to match the changing conditions of the network. Rate control is established by matching the speed at which the sending HyperIP is sending data, to the speed at which the peer HyperIP is receiving the data. The rate at which the remote server application is receiving the data is returned as part of the HyperIP protocol information, and is periodically used for recalculating the transmission rate.

However, there may be some situations that require specific rate throttle limits to be set. The maximum HyperIP rate limit is controlled by a value contained in the software key. However, HyperIP also provides the ability to set the maximum rate limit to another value, as long as it is less than the value contained in the software key. This may be useful in situations where HyperIP is sharing bandwidth with other non-accelerated applications. Setting the HyperIP rate limit to a value less than the total available bandwidth guarantees that HyperIP will not consume all of the bandwidth, and that some will remain for other applications.

### ***HyperIP compression***

HyperIP also has the ability to compress data before it is sent over the WAN. The benefit of compression is that it may reduce the WAN bandwidth usage, and effectively increase the application data throughput over the network. HyperIP can provide compression benefits at rates of up to 155Mb/s after compression.

HyperIP compresses aggregated blocks rather than individual packets. Compression algorithms are generally more efficient when compressing large amounts of data at a time, rather than working with small individual packets. Compressing large blocks results in a smaller number of packets traversing the WAN. Since the packets are created from the compressed data block, each packet will be full of compressed data. Compressing individual packets does not reduce the number of packets, since each packet will only be partially full of compressed data.

The compression feature is implemented as “adaptive compression”. This means that if defined levels of compression are not being achieved, the compression feature shuts itself off, and data will be sent uncompressed over the HyperIP connection. HyperIP will then periodically re-enable compression to determine if the current data being sent is now compressible. A benefit of “adaptive compression” is that typically higher data throughput rates can be achieved for data that is not very compressible by sending it uncompressed.

## **Mitigating the effect of packet loss**

### ***TCP congestion control***

Packet loss occurs when packets are damaged and discarded, or when the capacity of an intermediate network component is exceeded, which then causes that component to discard packets. The amount of data that can be sent is limited by the TCP congestion window. The congestion window is flow control imposed by the sending TCP that is based on its assessment of perceived network congestion.

TCP always interprets packet loss as a sign of congestion, and dramatically reduces the congestion window and the rate at which the congestion window will be allowed to grow. This limits the number of packets that can be in flight in the network at any one time, which then decreases throughput.

In networks where congestion is not the only cause of packet loss, it can be a costly mistake to reduce the congestion window. For example, when packets are lost due to corruption, reducing the congestion window unnecessarily decreases throughput.

In both cases, the congestion window is allowed to grow during the recovery from packet loss, but if additional packet loss or packet corruption occurs during this recovery, the congestion window is again dramatically reduced. If the congestion window is repeatedly smaller than the available network capacity, it will be impossible for TCP to fully utilize that capacity.

### ***HyperIP congestion control***

Just as with TCP, HyperIP cannot distinguish between packet loss due to congestion and packet loss due to corruption. However, rather than making window adjustments when packet loss occurs, HyperIP uses rate-based congestion controls to limit throughput.

The rate that data is sent depends upon the rate the receiver has taken data in the past. The receive rate is continuously monitored, and used to make adjustments to the send rate. Since the receive rate will decrease when packet loss occurs, that will cause the send rate to decrease to match the receiver. Similarly, as the event causing the packet loss subsides, the send rate will increase, since the receive rate will be able to sustain a higher rate.

The send rate is one of the factors used to periodically recalculate the capacity of the network (i.e. the “pipe size”). Therefore, rate adjustments based on packet loss and recovery are reflected in subsequent pipe size calculations. Regardless of whether packet loss is due to congestion or corruption, HyperIP is able to drive throughput up to the capacity of the WAN.

## **Mitigating the effect of latency**

The latency time represents the additional time incurred for the act of transmitting data over a network. Longer network distances, as well as additional router hops in a network, result in higher latency times.

The bandwidth capacity represents the total amount of data that can exist on the network at any point in time (“the pipe”). The formula used for the bandwidth capacity calculation is:

$$\text{Capacity} = \text{bandwidth (bits/second)} \times \text{round-trip-time (seconds)}$$

### ***TCP Window Scaling***

To utilize the full available bandwidth of a data session, enough data must be sent to “fill the pipe”. The amount of data that can be “in the air” at any point in time is governed by the window sizing capability of the TCP stacks on both sides of the connection, and the applications. Most TCP implementations support a feature called window scaling, which allows large windows to be used. However, each TCP implementation may use a different default for setting the initial size, may implement the feature but have it disabled by default, or in some cases, may not even implement window scaling at all. At best, it then becomes up to the user to tune the TCP stacks on both the sending and receiving side of the connection, and possibly even the TCP applications in order to enable this capability. In many organizations, this may not be a viable option.

Another disadvantage of tuning the TCP stacks is that the changes apply to all applications (both LAN and WAN) running on that particular server or storage controller, when the desire might be to only selectively enable a feature for one particular application connection. Window scaling may still not provide a large enough window, or the server or application may not have enough buffer space available to support the capacity of the network. For example, the total round-trip time for a 3000 mile connection is approximately 60 milliseconds, creating an available data “pipe” at 100 Megabits per second (Mbps) of 750 Kilobytes. A satellite connection (540 milliseconds round-trip time) at 45 Mbps creates a 3 Megabyte pipe. The actual “pipe” will be further affected by latency induced by network hops between application servers. If window scaling is not implemented, or is not tuned to the network requirements, there may still be a large amount of “empty pipe”, i.e. unused, but still available, bandwidth.

### ***HyperIP Window Scaling***

In order to resolve the performance degradation problem caused by latency, HyperIP dynamically calculates round-trip times, bandwidth capacity, and transmission rates, and uses that information to calculate the capacity of the network. In contrast to the window scaling issues of native TCP applications, the HyperIP window size is dynamically calculated based on the capacity of the network. This calculation remains totally independent of any application TCP stack capability. It doesn’t matter whether or not the application servers are tuned to use enhanced TCP options, since those TCP options only

affect the LAN segment between the application server and the local HyperIP appliance. The performance gain over the WAN is achieved by HyperIP.

HyperIP uses a “continuous block” protocol to keep the network bandwidth pipe full. The capacity of the network determines the HyperIP window size. The latency times are determined by the sending side of HyperIP, by calculating the round trip times for the transmission and acknowledgement of messages. HyperIP makes adjustments to its bandwidth capacity value by periodically calculating a new value as the values for rate and latency time change. The protocol allows the aggregated and compressed data blocks to stream over the HyperIP connection, up to the limits of the network capacity. This results in more efficient utilization of the network, and minimizes or eliminates, the “empty pipe” situation.

## **Summary**

HyperIP provides exceptional value by enabling better network performance for TCP applications. It shields the applications from performance degradation due to packet loss, latency, and jitter. HyperIP achieves this by generating local acknowledgements to the application, and taking responsibility for the delivery of data to the remote application server or storage controller; efficiently utilizing the available bandwidth for all applications; and mitigating the effects of packet loss and latency on performance.

Network Executive Software, Inc.  
6420 Sycamore Lane N #300  
Maple Grove, MN 55369  
(763) 694-4300  
[www.netex.com](http://www.netex.com)