

**Achieving
High Performance TCP/IP
on your
Tandem System**



FAILSAFE SYSTEMS

"Excellence in Action"



9950 W. Lawrence Ave., Suite 202, Schiller Park, Illinois 60176
Phone (847) 671-5700 FAX (847) 671-2988
info@failsafe-systems.com <http://www.failsafe-systems.com>

Introduction

During the last several years, the TCP/IP protocols have achieved wide acceptance in the commercial marketplace as the defacto standard to achieve multi-vendor connectivity.

As one might imagine, both the number of commercially available products as well as information regarding TCP/IP has grown at the same time. Literally hundreds of TCP/IP products are now available and countless articles, papers, and books have been published.

However, little has been written regarding the performance considerations of using TCP/IP, especially for Tandem users.

The purpose of this paper is to consider the performance implications of supporting TCP/IP on your Tandem system, particularly how to achieve high levels of performance and the factors that determine the performance profiles of specific products.

What Do We Mean By "Performance"?

Different meanings can be associated with the word "performance" depending upon the context in which it is used.

It's only fair to define the various ways in which we will use this word. In the context of this paper, the word performance will be used to mean:

Speed (or bandwidth) -- This is the most common meaning of the term. In this context, it simply means how fast data can be moved. This can be easily determined via empirical testing.

Resource Consumption -- Certainly, a TCP/IP implementation which consumes excessive resources can be viewed as a relatively poor performer when compared to an implementation which consumes far less resources. For the purposes of this paper, resource consumption is determined by host cpu utilization.

Cost -- The overall and absolute cost of deploying a TCP/IP implementation can be easily calculated over its life cycle. For the purposes of this paper, a five year life cycle is assumed and the cost calculation includes all hardware and software acquisition costs to support TCP/IP plus ongoing software license fees for the life cycle.

Cost Efficiency -- This metric is used to gauge how much work is done by a TCP/IP implementation per unit of cost.

The Underlying Technologies

It is absolutely impossible to survey performance issues without at least a rudimentary understanding of the underlying technologies of Ethernet and the TCP protocol.

The reason for this is really simple. Certain critical functions associated with Ethernet and the TCP protocol must be implemented to support TCP/IP over Ethernet. As we shall see a little later in this paper, it is the support of these functions that determines the performance profile of a specific TCP/IP product.

Application	File Transfer Protocol (FTP)	TELNET Protocol	User Defined Protocol
Presentation			
Session	Transmission Control Protocol (TCP)		
Transport			
Network	Internet Protocol (IP)		
Data Link	Ethernet		
Physical	Coax, Twisted Pair, Etc.		

Figure 1 - Simplified ISO Model Description Of TCP/IP

Figure 1 shows a somewhat simplified ISO model description of the TCP/IP protocol stack over Ethernet. Other layers of the protocol stack certainly implement critical functions but will be disregarded for the purpose of this paper because their support does not have any significant performance ramifications.

Ethernet

Ethernet is essentially a high speed broadcast technology operating at 10 million bits per second (10 Mbits/second).

The quantum of communication over Ethernet is the Ethernet packet, depicted in Figure 2.

Ethernet Header 14 bytes	User Data 46-1500 bytes	Checksum 4 bytes
-----------------------------	----------------------------	---------------------

Figure 2 - The Ethernet Packet

Carries Up To 1,500 Bytes Of Data And Supplies Only Unreliable Datagram Service

The Ethernet packet is limited in size. After deducting fixed areas for the Ethernet header and checksum, no more than 1,500 bytes remain to be used for actual data.

Also note that Ethernet provides only a datagram service. The sender of an Ethernet packet has no idea if the intended recipient did in fact receive the data. Conversely, a recipient cannot detect if data was ever sent but not received.

The function of message assurance solves these problems of delivery uncertainty. In the case of TCP/IP over Ethernet, this message assurance function is implemented in the TCP protocol.

The TCP Protocol

The TCP protocol implements a reliable session oriented type of service over Ethernet.

Since the TCP data is by definition transmitted over Ethernet, it must be fit into an Ethernet packet as shown in Figure 3.

Ethernet Header 14 bytes	IP Header 20 bytes	TCP Header 20 bytes	User Data 6-1460 bytes	Checksum 4 bytes
-----------------------------	-----------------------	------------------------	---------------------------	---------------------

Figure 3 - The TCP Packet

Carries Up To 1,460 Bytes of Data, Although 1,024 Bytes Is Normally Implemented As The Maximum.

Note that after now deducting room for the required IP and TCP headers, no more than 1,460 bytes remain for actual data.

A special note is in order now with respect to data size limitations endemic in the TCP protocol. Although the theoretical maximum is in fact 1,460 and many TCP/IP implementations do in fact support this size, a far more common value is 1,024 bytes per packet due to implementation convenience. As a result, this paper will use 1,024 as the maximum value.

In order to implement a reliable protocol where the sender of a TCP packet can determine if the intended recipient did in fact receive the packet, an acknowledgement mechanism is used in the TCP protocol. Every byte in a TCP session is consecutively numbered. The recipient of a packet acknowledges its receipt by sending back another packet with a bit set in the header indicating an acknowledgement and the sequence number set to the next expected data byte. Normally, the receiver will send one acknowledgement packet for each data packet received.

Note how the data size limitation and the acknowledgement mechanism impute upon both the sender and receiver of data a certain level of i/o as depicted in Figure 4. In this simple example, exchanging only 2,000 bytes of data requires four i/o's to be performed by the sender and four i/o's on the part of the receiver.

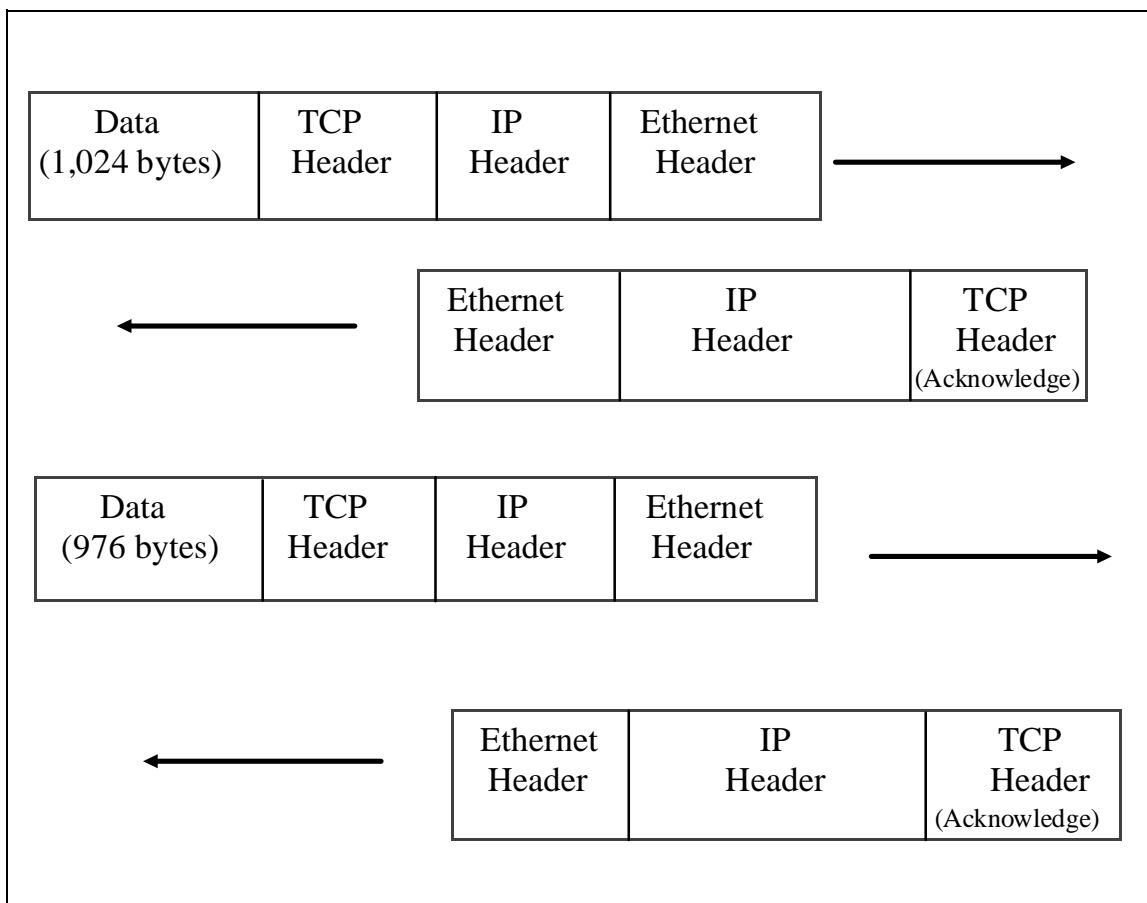


Figure 4 - Sample 2,000 Byte Data Exchange
Even A Single 2,000 Byte Transfer Requires Four I/O's For Both Sender And Receiver

Every TCP/IP implementation must therefore perform the following functions:

Packetize - break the data stream up into individual TCP packets when sending,

De-packetize - assemble individual TCP packets into a usable data stream when receiving,

Acknowledge - send TCP acknowledgement packets when receiving, and

Process Acknowledgements - receive and process TCP acknowledgement packets when sending.

The TCP I/O Pressure Formula

In order to properly evaluate TCP/IP performance, it would be extremely convenient to have a means to predict the amount of i/o that a TCP/IP implementation must do to accomplish a certain amount of work.

Since each TCP must packetize (or de-packetize) the data stream and receive (or send) acknowledgements, we can easily approximate the i/o requirements (i.e., the number of i/o's that must be performed) by the following formula

$$\text{i/o's} = \frac{\text{data size}}{1,024} \times 2$$

The division in this formula is simply a reflection of the packet size limitations while the multiplication is simply a reflection of the acknowledgement mechanism.

Now, let's apply this i/o pressure formula to gain some insight into the actual i/o pressure that must be sustained when supporting TCP/IP over Ethernet.

Most commercially viable TCP/IP implementations are able to achieve a bandwidth in the range of 2 million bits per second (2 Mbits/second). So let's say that our desired bandwidth is exactly that.

$$\text{Desired Bandwidth} = 2 \text{ Mbits/second}$$

The number of i/o's required to send or receive 2 Mbits (or 250 KBytes) can be easily estimated with the i/o pressure formula as

$$\text{i/o's} = \frac{250 \text{ KBytes}}{1,024} \times 2 = 500$$

Now since we desire to transfer 2 million bits in one second, the i/o rate is

$$\text{i/o rate} = 500 / \text{second}$$

and the average time to perform each i/o is

$$\text{average time per i/o} = 2 \text{ milliseconds}$$

If you stop and consider the ramifications of this calculation, it should be obvious that achieving high performance is not a simple matter. Millisecond level i/o must be performed on a sustained basis as a direct result of the TCP acknowledgement and packetization requirements of the protocol.

Figure 5 shows the linear relationship between the i/o rate and bandwidth. Certain benchmark results are conspicuously marked in this figure and will be discussed later.

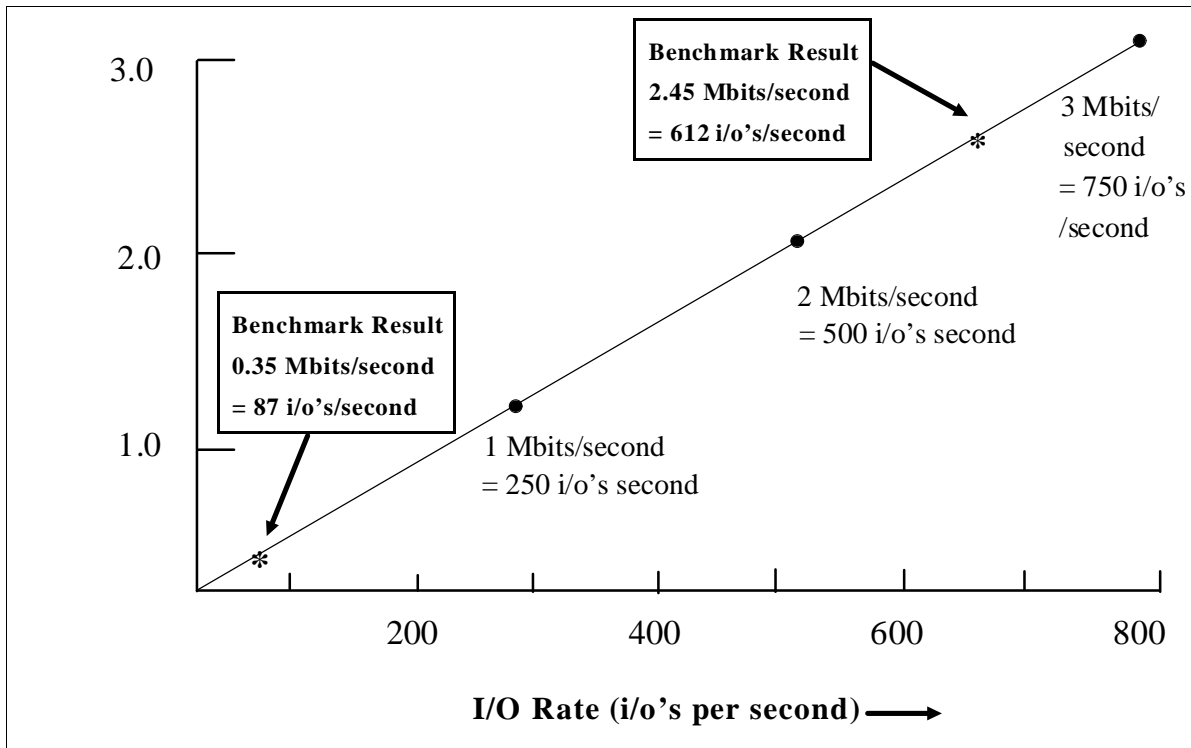


Figure 5 - I/O Rate Versus Bandwidth

The Ability To Achieve High I/O Rates Determines Performance.

Note that as the ability to perform i/o increases, the bandwidth increases. Thus, the ability to achieve high i/o rates determines performance.

This leads us to our first conclusion.

Conclusion #1

**To achieve high performance,
a TCP must be able to achieve very high i/o rates.**

Architecture Versus Implementation

An important point in this paper pertains to the question of architecture versus implementation.

In much the same sense as in building construction, architecture pertains to the overall design while implementation pertains to the actual building of what was designed.

In the context of supporting TCP/IP over Ethernet, the question of architecture is crucially important. This is simply a direct result of the intense i/o pressure that must be endured to achieve high performance.

The key architectural question to be asked prior to implementation is how can a TCP/IP be architected so that it:

- 1) can sustain very high levels of i/o pressure (i.e., so it will provide high bandwidth),
- 2) will have little impact on the application environment,
- 3) will be inexpensive, and
- 4) will provide a cost effective solution?

In other words, how can it be architected so that it meets *all* of our definitions of "high performance"?

Generally speaking, there are two architectural choices:

As its name would imply, the *host architecture* runs the entire TCP/IP protocol stack on the host processor. As a result, all i/o pressure is on the host processor.

On the other hand, the *front end processor (FEP) architecture* runs only a small portion of the protocol stack on the host processor. Low level protocols (including all TCP protocol

and Ethernet functions) are run on an inexpensive processor which is better suited to handle the intense i/o pressure.

Figure 6 compares both of these architectures relative to the ISO model. Note that the additional function of communicating with the FEP is denoted in the FEP architecture diagram. This function's impact will be considered later.

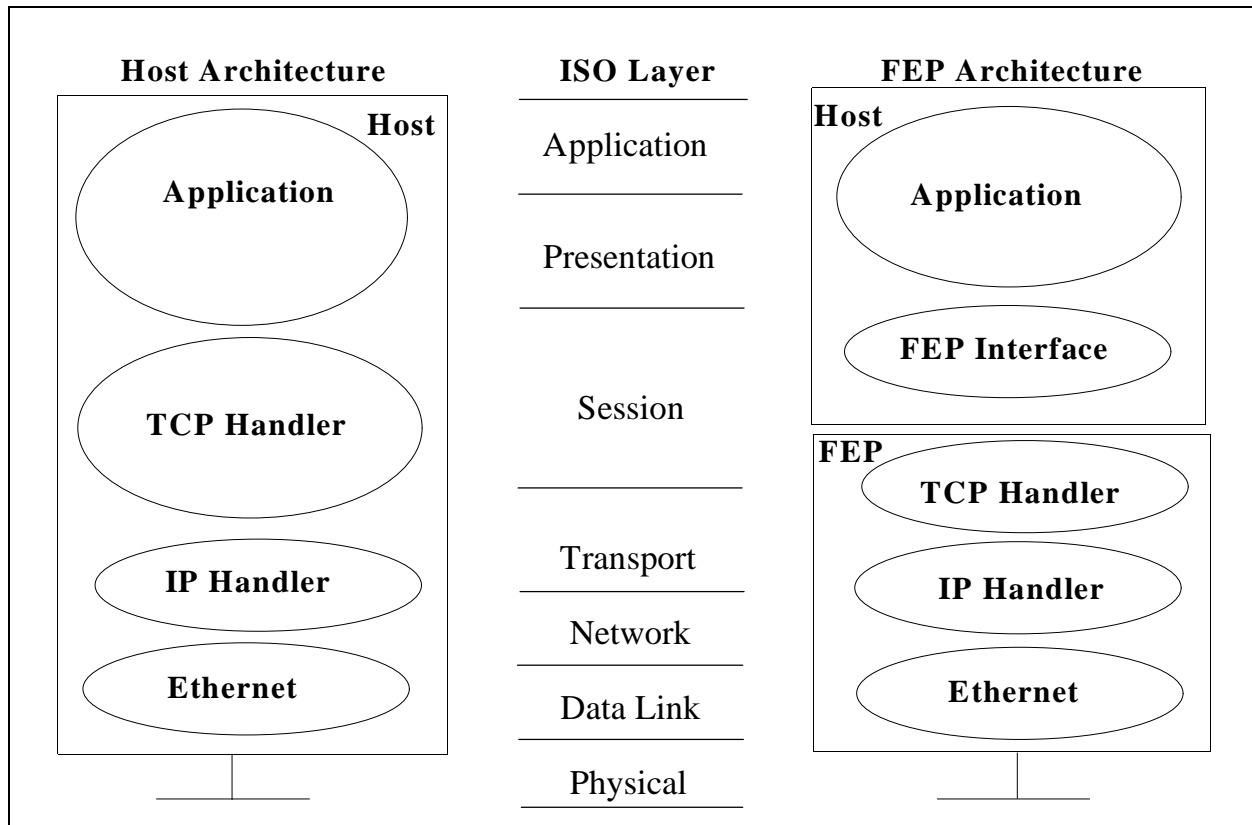


Figure 6 - Host And FEP Architectures
The FEP Architecture Offloads The I/O Pressure

Why would one want to architect a TCP/IP using the FEP architecture?

The answer is simple. As we have already seen, it is the ability to handle the intense i/o pressure that determines performance. The basic idea of the FEP architecture is to offload this i/o pressure on a machine better suited to handling sustained millisecond level i/o in a cost efficient manner.

Empirical results prove that this is not just a theoretical advantage, but *the FEP architecture provides consistently better performance in every respect.*

FEP Versus Host Architecture Benchmark

Benchmarks were conducted using both a FEP and host architecture products on a Tandem CLX 800 system. File Transfer Protocol was used to repeatedly transfer a 6 MByte test file between the Tandem file system and an IBM RS/6000 workstation over Ethernet. Transfers were performed in both directions and results were averaged for simplicity of presentation.

Tandem's Measure product was employed to determine the cpu consumption of various processes.

It is important to note that this benchmark was conducted by an independent third party. No vendor personnel were present during the benchmark, although phone support was occasionally provided to the customer with respect to tuning and configuration details.

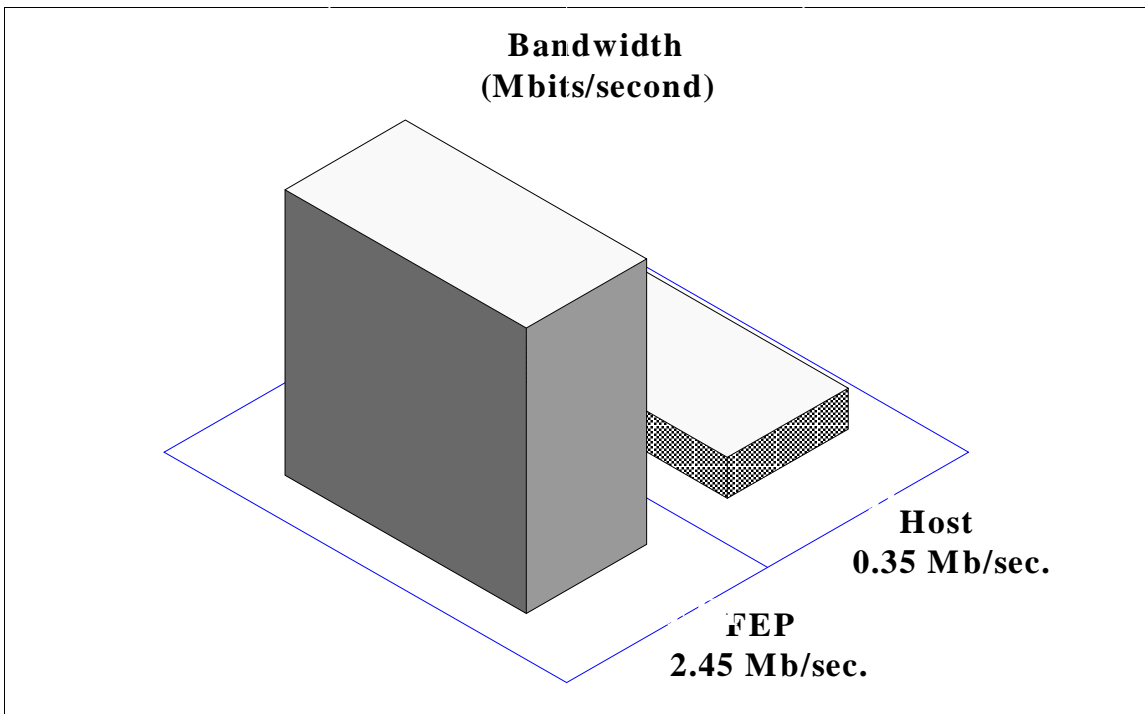


Figure 7 - Bandwidth

The FEP Architecture Is Seven Times Faster Than The Host Architecture

The test results showed that the FEP architecture consistently achieved a bandwidth much higher than the host architecture, averaging 2.45 Mbits/second while the Host architecture averaged only 0.35 Mbits/second. Test results for bandwidth are shown in Figure 7.

Conclusion #2

As a matter of architecture and not implementation, the FEP architecture FAR outperforms the host architecture.

The total amount of Tandem cpu usage was simultaneously measured. Test results for cpu utilization are shown in Figure 8.

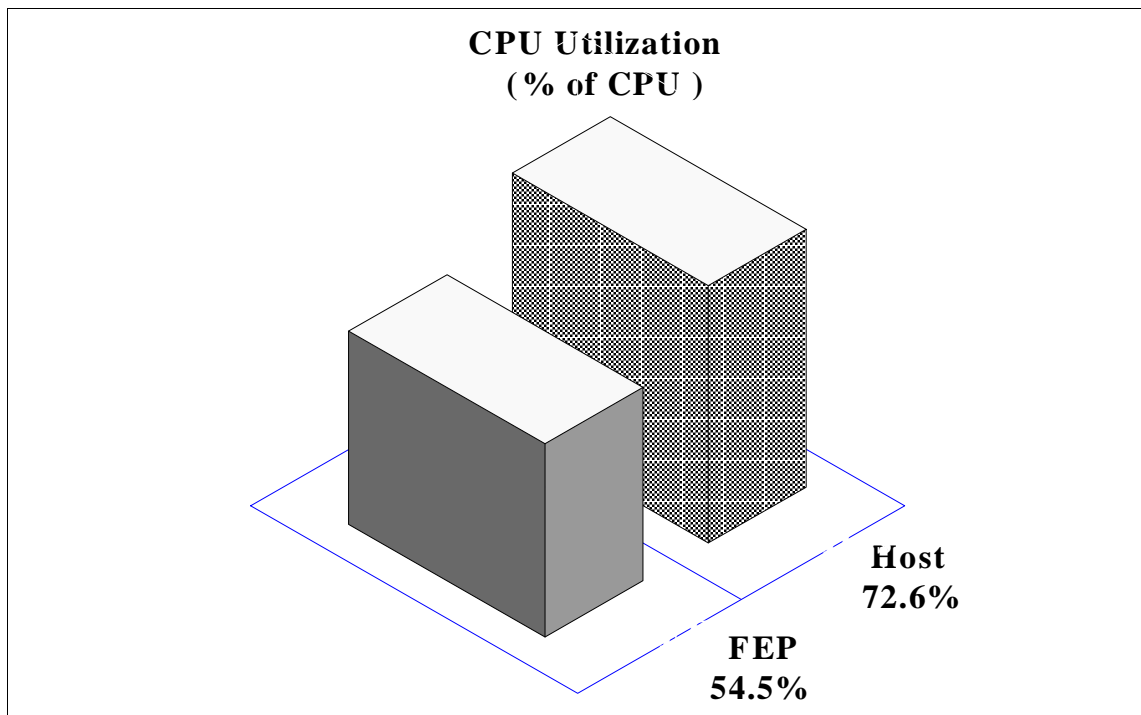


Figure 8 - CPU Utilization

The FEP Architecture Consumed Far Less CPU While Accomplishing More

Note that in spite of the fact that the host architecture performed at roughly one seventh of the speed of the FEP architecture, it still consumed roughly 50% more host cpu resources than the FEP architecture. In order to take both these factors into account, we must look at the Architectural Efficiency depicted in Figure 9.

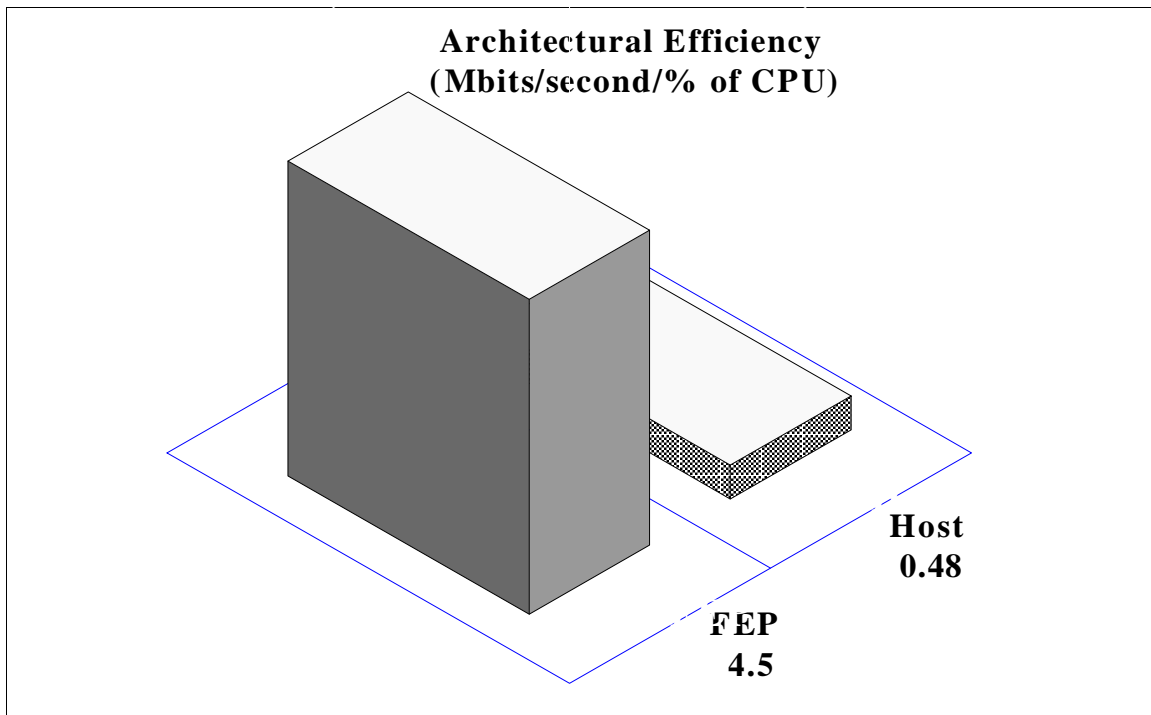


Figure 9 - Architectural Efficiency

The FEP Architecture Is Nearly Ten Times More Efficient Than The Host

Architectural efficiency is calculated simply by dividing bandwidth by cpu utilization. In other words, this efficiency shows how much work is accomplished per unit of host cpu consumption. ***Note that the FEP architecture is nearly ten times more efficient than the host architecture.***

Conclusion #3

As a matter of architecture and not implementation, the FEP architecture consumes less resources while accomplishing more (i.e. is FAR more efficient).

Finally, cost considerations are summarized in the weighted cost shown in Figure 10.

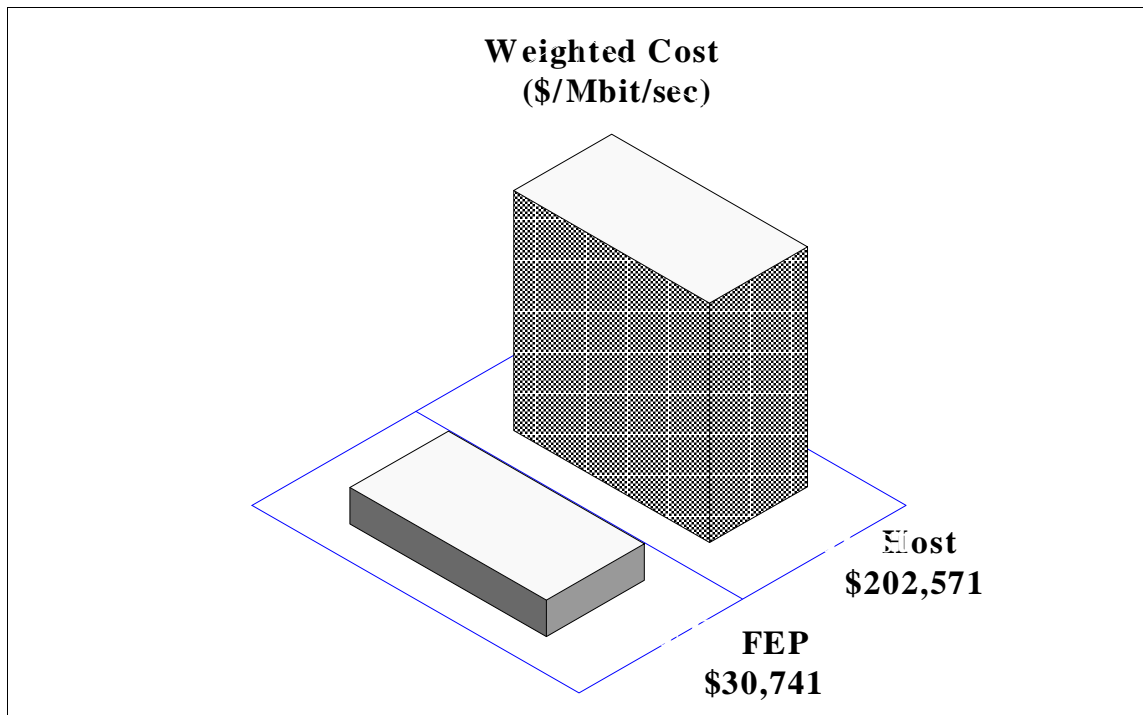


Figure 10 - Weighted Cost

The FEP Architecture Is Seven Times More Cost Efficient Than The Host

Weighted cost is calculated by simply dividing the life cycle cost by the achieved bandwidth. It is effectively a measure of the total real cost to the end user customer per unit of bandwidth performance. Life cycle cost is calculated to be all hardware and software acquisition costs plus ongoing software license fees during a five year period.

Conclusion #4

As a matter of architecture and not implementation, the FEP architecture is FAR more cost efficient than the host architecture .

The Hidden Cost Of Low Performance

Low performance certainly has a hidden yet very real cost associated with it.

Simply put, people are usually waiting for computers to perform their tasks before the people can do theirs. Figure 11 shows the total elapsed time for the FEP versus host architectures to complete the test problem.

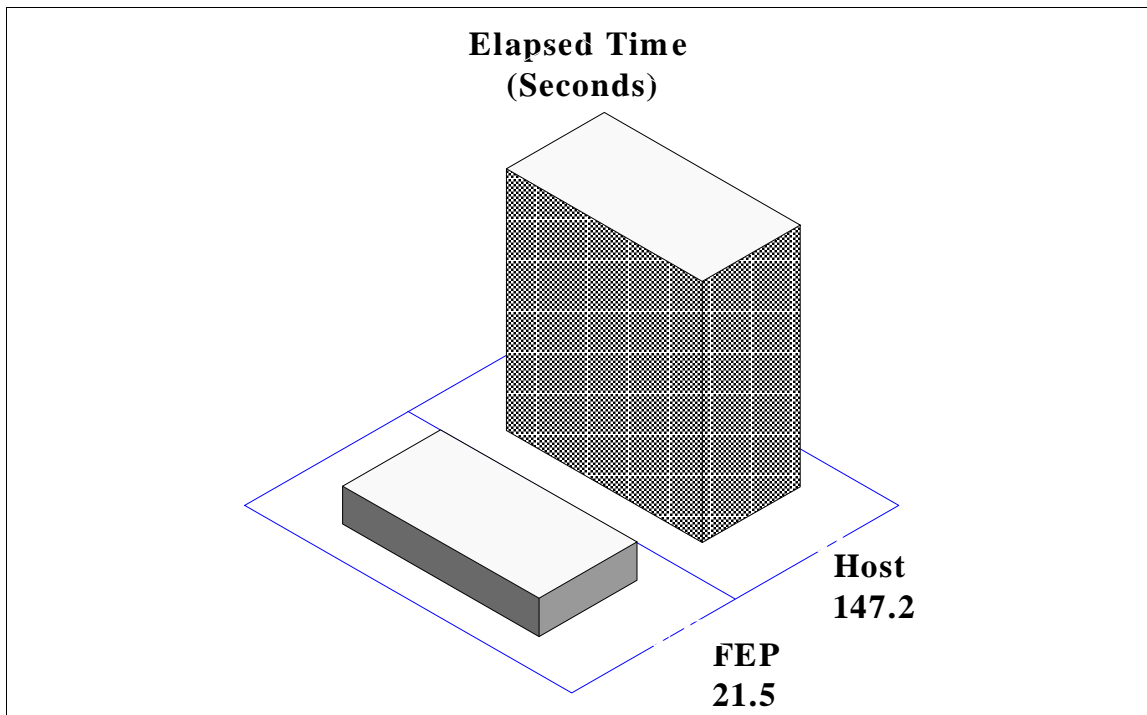


Figure 11 - Elapsed Time

People Wait Much Longer For The Host Architecture To Deliver Data

What must be evaluated on a site by site basis is the effect of the relatively low performance on people productivity. For example, a highly paid engineer may be waiting for file transfers to take place before he can do his job with the data. Occurring frequently enough, the low performance solution can easily impute a not only very real but very significant cost upon the organization.

Since this hidden cost can be evaluated only on a site specific basis, it has not been included in any actual cost computations referred to above.

The "Zero Traffic" Problem

One would naturally assume that if no TCP/IP traffic is actually originating from or destined to your Tandem system, then absolutely no cpu utilization would be expected.

However, this is not necessarily true. In fact, significant cpu utilization for the host architecture can be observed in many cases, even though no traffic is actually destined for the Tandem host.

Basically, there are certain types of TCP/IP traffic known as broadcast datagrams and address resolution protocol (ARP) packets. In both of these cases, the Ethernet packet is not

addressed to any specific node on the network. All nodes receive and must process these packets at least through a relevant protocol layer that can make the determination that the packet is or is not of any interest to this host. Even if they are not of interest, the host still must process them.

How significant can this "zero traffic" cpu utilization be? It obviously depends upon the frequency at which they occur. Unfortunately for the host architecture user, they can occur far more frequently than might be expected.

One Tandem user of the host architecture product has reported that *twenty percent of every VLX system connected to their busy backbone Ethernet was consumed by this type of traffic* even though absolutely no traffic was destined for his Tandem systems.

Another user of the host architecture has reported that *effectively his system was brought down by a malfunctioning PC on the network that began looping sending ARP requests.*

The important point to remember here is that the FEP architecture guards against this zero traffic cpu utilization. All broadcast datagrams and ARP packets that are not destined to the host are discarded by the FEP, insuring zero host cpu utilization for the zero traffic problem.

Conclusion #5

**As a matter of architecture and not implementation,
the FEP architecture protects the host from unwanted network activity
while the host architecture is vulnerable to that activity.**

Is Performance Really Determined By Architecture?

As you may already suspect from the foregoing information, the answer is absolutely yes.

How do we know that? There are really three different answers as to how we know that architecture is the primary determinant.

First, the benchmark results are easily understood (and in fact could also easily be predicted in advance) based upon the i/o pressure discussion earlier in this paper.

Refer again to Figure 5 which marked these benchmark results in advance on the i/o rate versus bandwidth curve. Since the host architecture handles all of the TCP and Ethernet activity on the host processor and since the host processor has a definite limit on the number of i/o's per second that it can perform, *these bandwidth results are simply a reflection of the fact that a FEP can sustain much higher levels of i/o than the host processor.*

Also, refer again to Figure 9 which showed architectural efficiency. The fact that the FEP was almost *ten times* more efficient is again simply a reflection of the offloading effect of the FEP. **Higher bandwidth (i.e., higher i/o rate of the FEP) coupled with lower cpu utilization (i.e., offloading TCP and Ethernet processing to the FEP) make the FEP architecture far superior to the host architecture in terms of efficiency.**

Secondly, we must look again at the performance benchmark data. But instead of comparing one architecture against the other in one respect or another, compare the resources consumed by different portions of each.

Specifically, if we compare network i/o cpu utilization against file i/o cpu utilization, we can attempt to see if a disproportionate amount of resources are consumed by one part of the system or the other. The purpose of this comparison is to determine where the bottleneck is located.

Figure 12 shows the relative cpu consumption of the host architecture. The cpu consumption of the combined Ethernet and TCP processes is nearly seven times higher than the cpu consumption of the combined FTP server and disk process. **This is clearly indicative of severe i/o pressure on the network i/o portion of the host architecture.**

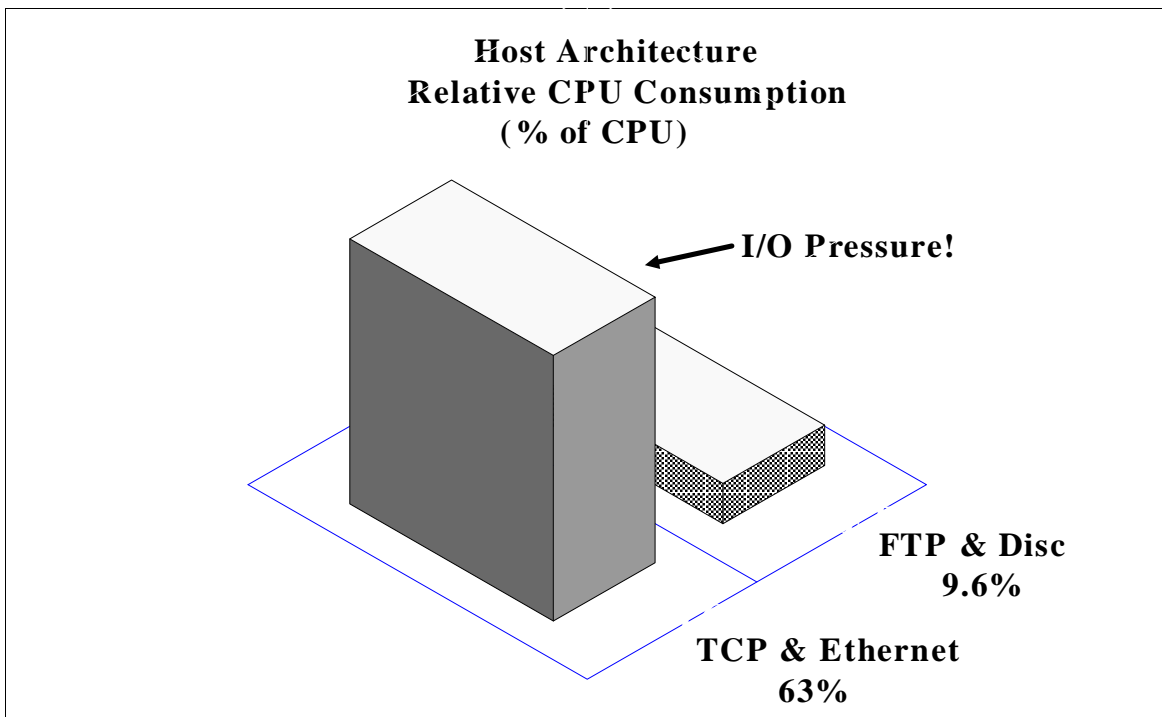
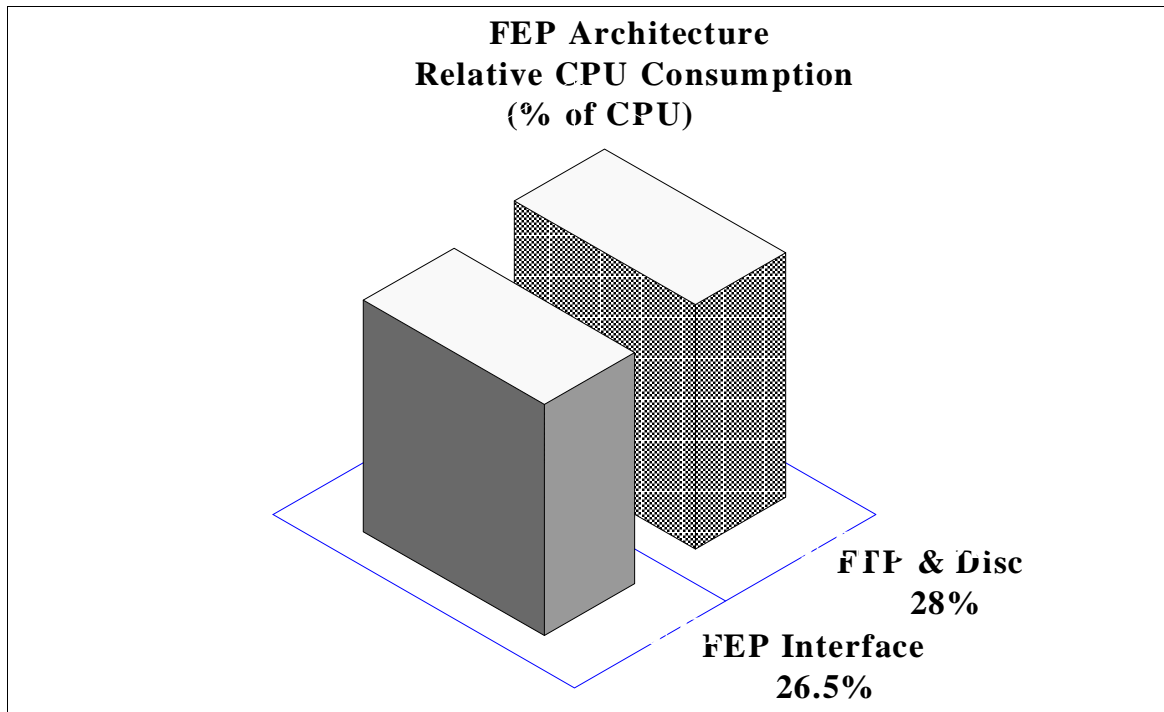


Figure 12 - Host Architecture Relative CPU Consumption
Severe I/O Pressure Is Clearly Observed On The Host CPU

Figure 13 shows the relative cpu consumption of the FEP architecture. The cpu consumption of the FEP interface process is virtually identical to the cpu consumption of the FTP server

and disk process. This is clearly indicative that the i/o pressure and bottleneck are not on the host.



***Figure 13 - FEP Architecture Relative CPU Consumption
I/O Pressure Is Significantly Reduced Using The FEP***

The final answer to the question of architecture determining performance is less technical but even more persuasive.

TCP/IP support is available for many other systems. In fact, it is estimated that more than 300 TCP/IP products are available in the marketplace today. How do they perform on similar machines?

- Both host and FEP architectures have been available on DEC systems for many years. Virtually identical ratios are observed between host and FEP architectures for speed, resource consumption, and cost on DEC systems.
- Stratus has chosen to implement their TCP/IP using a host architecture. When compared to the Tandem based host architecture product, almost identical performance and resource consumption is observed.
- IBM originally architected their TCP/IP on the host. Recognizing the limitations of that decision, it was recently converted to a FEP architecture.

Conclusion #6

Architecture and not implementation is the primary determining factor for high performance.

Cost And Band width Relationships

Another interesting area to explore is the cost versus bandwidth relationships for both host and FEP architectures.

As we have already discussed in detail, i/o pressure is the primary determinant of performance, not only in its meaning of speed or bandwidth, but also cost.

Cost increase is naturally linear with performance increase for the host architecture. This is an obvious and natural consequence of the fact that host processors cost money. As performance increases on a given processor type, so does host cpu utilization and cost.

Equally obvious and natural is that by moving the host architecture product to a faster processor to achieve higher bandwidth increases cost because faster processors cost more money than slower ones.

In other words, *the host architecture is very sensitive to the processor type it is running on.* The good news is that it can be moved to a faster processor type to dramatically increase bandwidth, but the bad news is that that will also dramatically increase cost.

On the other hand, the FEP architecture is relatively insensitive to the processor type on which it is being run. This is a natural consequence of the fact that the intense i/o pressure has been offloaded to the front end processor. As a result, the bandwidth and cost curves for the FEP architecture are relatively flat across all processor types.

Figure 14 depicts these cost and bandwidth relationships for both host and FEP architectures.

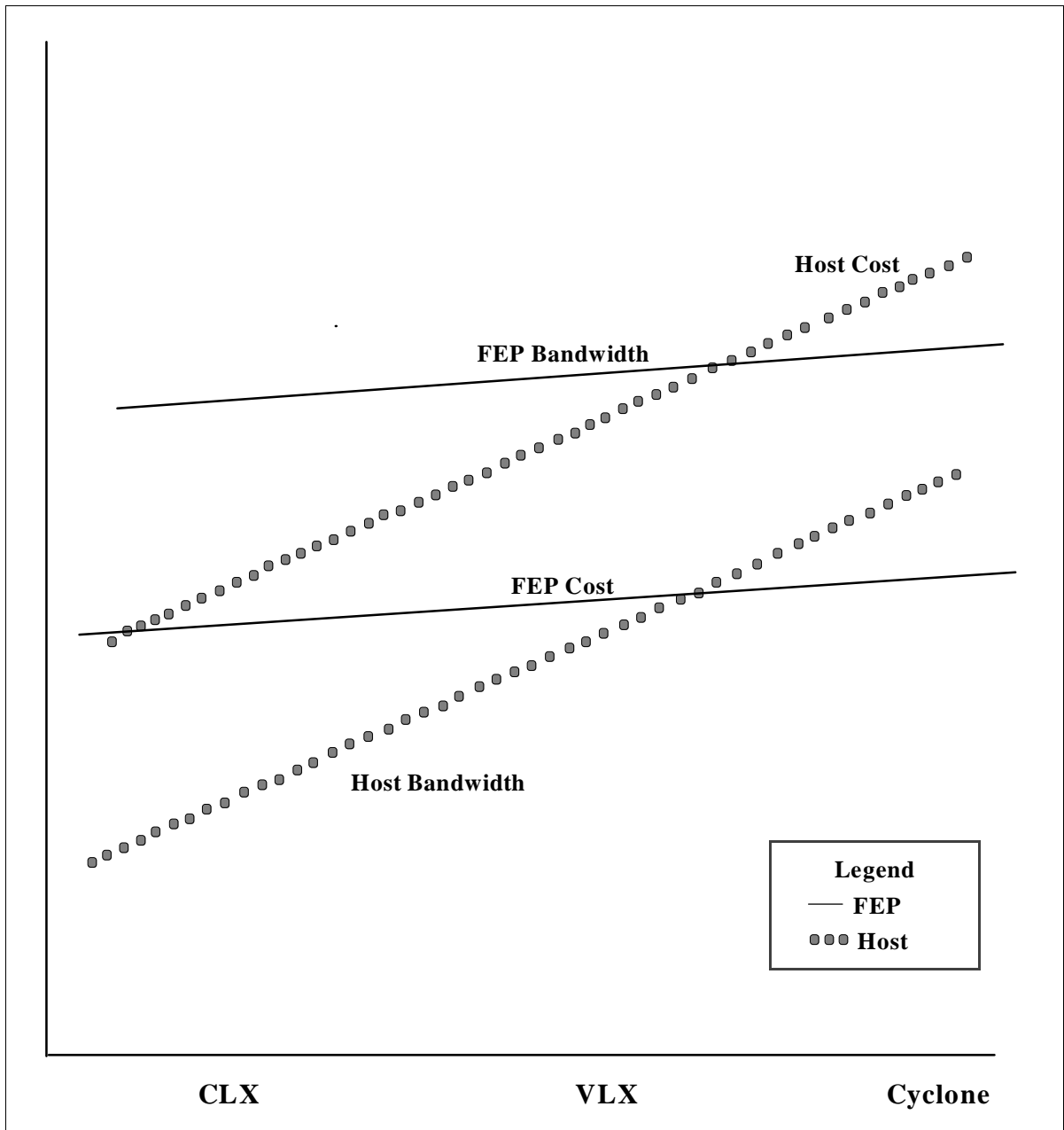


Figure 14 - Cost And Bandwidth Relationships

The FEP bandwidth far exceeds host bandwidth across all processor types including the "best case" bandwidth for the host architecture on a Cyclone.

Note the dramatic increase in cost for the host architecture across processor types, while the FEP architecture cost curve remains relatively flat.

Note also that the FEP architecture is slightly more expensive than the host architecture for the low end CLX system. This seems easily justifiable in view of the seven to one bandwidth advantage.

Future Improvements

The purpose of this section is to survey the most likely areas of product improvement and to project how performance might be impacted.

Note that the following discussion pertains to questions of implementation and not architecture. In other words, given one architecture or another, how can performance be improved by improving the implementation?

"Tweaking The Code"

Code tweaking can certainly provide some benefit for both FEP and host architectures. However, expectations must be realistically tempered by consideration as to where bottlenecks and i/o pressure are found.

For the FEP architecture, some benefit can be anticipated in cost and resource reduction. Since the FEP architecture bottlenecks on the host to FEP interface, only marginal speed improvement (if any) can be anticipated.

Since the host architecture bottlenecks on the i/o pressure on the host processor, code tweaking may yield benefit, *but only up to the limits of the host processor to process Ethernet and TCP packets*. Effectively this option is oriented to dealing a little better with the i/o pressure, but does not remove the i/o pressure from the host. As a result, some speed improvements can be anticipated, but cost and other considerations remain essentially the same.

Faster Controllers

Improving the Ethernet controller to perform i/o faster is another area of possible performance improvement.

For the FEP architecture, this is probably the most promising area for performance improvement. As exhibited in Figure 13, the bottleneck is clearly not on the host, but in the host to FEP interface. Introducing a faster Ethernet controller into the host to FEP interface can be expected to provide a performance improvement proportional to the increase in controller speed.

For the host architecture, a faster controller may provide marginal speed improvement, but i/o pressure on the host processor will remain the bottleneck. *Note that a faster controller will actually **increase** the i/o pressure on the host*, absorbing any cpu utilization reduction made available from "code tweaking". The net effect is that speed may increase, but still only to the limit of the host processor's ability to process Ethernet and TCP packets; but since the i/o pressure increases, cost increases proportionally.

Faster Host Processors

The final area of possible performance improvement is that of faster host processors.

For the FEP architecture, little performance improvement can be realized by moving to a faster host processor. This is a direct result of the fact that the bottleneck is in the host to FEP interface.

For the host architecture, this is the most promising way to increase performance. Since its performance is limited by the host processor's ability to handle the i/o pressure, moving to a faster host processor increases the ability to handle that i/o pressure.

An obvious conclusion is that a host architecture site running on a CLX should upgrade to a Cyclone processor to achieve higher bandwidth.

Conclusion

TCP/IP performance is dictated by the ability of the system to accommodate and endure high levels of i/o pressure (i.e., achieve high i/o rates).

This is an inescapable result of the low level technologies of Ethernet and the TCP protocol.

Achieving high performance TCP/IP on a Tandem system is possible only using a front end processor architecture which offloads support for these low level technologies and accompanying i/o pressure to the FEP.

When compared to the host architecture, the FEP architecture provides clearly higher bandwidth, consumes less host cpu resources, is less costly, and is far more cost efficient.

Although implementations of either architecture may be improved, the general performance profile of an implementation is determined by its *architecture*. Changes in implementation can't and won't change the clearly superior performance profile of the FEP architecture over the host architecture.