

Enabling unique application architectures for TCP/IP processing environments
Greater parallelism, optimized performance



Table of contents

- HP's computing architecture**3
- Traditional solutions to networking bandwidth problems**4
 - SMP solutions and limitations4
 - Clustering solutions and limitations4
 - Distributed computing solutions and limitations4
- NonStop TCP/IPv6 solutions**5
 - Single IP5
 - Reactive distribution5
 - Round-robin distribution5
 - Improved fault tolerance5

Table of contents (continued)

Using parallel library TCP/IP for different networking applications	6
Monolithic server model	6
Listener server model	7
Distributor server model	8
Hybrid server model	10
Broker server model	11
Conclusion	12

The HP NonStop TCP/IP_{v6} software architecture was successfully implemented originally in the Parallel Library TCP/IP products, which have since become obsolete and replaced by NonStop TCP/IP_{v6} software. For convenience, this document refers to the new TCP/IP_{v6} product. However, the parallel library TCP/IP architecture concepts described in this document apply to both products.

The Internet has rapidly become a place to conduct business. It is a source of customers and is a new service-delivery mechanism. It is a facilitator of business-to-business transactions. The Internet is revolutionizing every aspect of customer care, banking, stock trading, merchandizing, and marketing. It also creates problems of performance, scale, interconnectivity, and reliability—problems that are unprecedented in the history of business computing. The democratization of access to computing, business, and information resources, fueled by the growth of the Internet, has led to new demands on computing capacity. Perhaps less obvious are the demands that this rapid growth of network bandwidth—available both at the edges as well as at the interior of the Internet—has placed on the computing resources that provide those services.

The growth trend of computing power, doubling in power every 18 months (as predicted by Moore's law), has been impressive and widely reported in the media. Although many people are familiar with Moore's law, few realize that commonplace networking bandwidth has been growing even faster.

Since the introduction of the PC, processing power has grown from 4.77 megahertz to over 1 gigahertz, and the speed trend seems to be accelerating. As an illustration of the growth rate of networking bandwidth, consider that at the advent of the PC, most network links to servers were 9.6 kilobits per second or slower, and Ethernet networking was largely a campus curiosity, with home dial-up links topping out at 1,200 bits per second.

Then DSL, Fast Ethernet, and OC-3 server links became the standard, providing 1,000 to 10,000 times faster bandwidth. In the same period that average processor clock rates increased by two orders of magnitude, average networking bandwidth increased by three to four orders of magnitude. And now, Gigabit Ethernet provides a 10-fold increase of Fast Ethernet network bandwidth, which is rapidly escalating to the 100-fold increase delivered by the 10Gigabit Ethernet. New Gigabit Ethernet standards are planned for the delivery of even more dramatic bandwidth increases, such as the 30Gigabit Ethernet standard, which may evolve to the 40Gigabit Ethernet standards that are slated for availability in the next few years.

This white paper explores past solutions for network bandwidth problems and illustrates how HP NonStop TCP/IP_{v6} software works in different networking application models.

The parallel library architecture adopted in NonStop TCP/IP_{v6} software provides increased throughput, improved fault tolerance, reduced processor utilization, and better scalability for mission-critical, enterprise-class applications. With its *reactive distribution*¹ capability, this software can distribute traffic across all of the processors within an HP NonStop server, providing greater parallelism without hop costs as well as increased reliability in the event of a processor failure.

HP's computing architecture

To meet the demands of the rapidly developing computing environment, HP has designed a new computing architecture, one that solves the problems of performance, scale, interconnectivity, and reliability: the NonStop server architecture.

This architecture features a high-speed and highly scalable I/O and interconnect technology known as HP ServerNet architecture. ServerNet technology offers a key capability, one heretofore not available with conventional computing architectures. This is the ability for peripheral I/O controllers to be shared by all processors in the NonStop Kernel operating system.

For NonStop servers, which have been "cluster based" for the past 25 years, the ability to share I/O from all processors in the system is not new. The NonStop Kernel operating system has had a cluster file system from its beginning, with the ability to access devices and files transparently, regardless of which processor in the cluster an application is running on. What is different with the ServerNet architecture is that rather than forcing I/O for a given device to go through the processor to which the device is attached, I/O can be performed directly from any processor in the system cluster.

With ServerNet architecture, all processors and peripheral devices (adapters) in the cluster have controlled, direct memory access to one another's memory. In other words, peripheral I/O adapters can be viewed as a direct cluster resource rather than being associated with a particular processor, as is found in traditional clusters.

¹ The term *reactive distribution* describes a technique by which the system is responsive to real, complex data processing needs. This technique is important for the system to be able to react quickly and appropriately, and to be able to spread the workload efficiently.

Traditional solutions to networking bandwidth problems

A single processor today does not have sufficient processing power to handle a full Gigabit Ethernet link, much less do real work simultaneously. Three commonly accepted solutions to overcome this problem are symmetric multiprocessing (SMP), clustering, and distributed computing.

SMP solutions and limitations

The SMP methodology allows higher performance than is available with a single processor but presents numerous problems. First, shared memory itself can be a bottleneck, either because of the bus used to access the shared memory or because of the shared memory itself. Plus, the engineering complexity required to make shared memory fast enough to fulfill the memory demands of n processors rather than one processor usually increases system costs. Although some systems have come up with ingenious solutions to this problem, such as switch-based memory systems and non-uniform memory access (NUMA) mechanisms, these approaches are highly complex and still have scalability limits.

Another issue relating to the scalability of protocol-processing performance in SMP systems is whether the protocol stack is “parallel capable.” In some operating systems, the TCP/IP stack executes “below” the threading level. That is, the TCP/IP stack can run only on a single processor in the SMP complex, thereby limiting the TCP/IP stack’s performance to what can be accomplished in a single processor.

Those operating systems that have implemented TCP/IP to take advantage of the native parallel processing aspects of the SMP system have another issue that affects performance of protocol processing: the use of spin locks for critical sections. Spin locks are used to control access to shared data structures by multiple processors. This controlled access to shared data structures can cause processors to stall. While one processor accesses shared memory, the other processors must wait on the spin lock. Such a wait typically causes the processor to stall—which, in today’s highly pipelined, superscalar processors, seriously affects performance. The more processors there are in an SMP complex, the greater the likelihood of spin-lock contention. This resource contention limits the parallel protocol processing performance and scalability of an SMP.

Clustering solutions and limitations

The NonStop Kernel operating system uses clustering as a scaling mechanism. Clusters have processors that don’t share memory. This methodology is more scalable than SMP because there is no contention and no use of spin lock because the processors operate independently. You can add more processors to handle greater workloads, as opposed to trying to speed up the memory. Also, by eliminating an obvious single point of failure, you achieve better fault tolerance.

Traditionally, in a cluster, data goes into the processor containing the TCP/IP stack, which may not be the processor containing the target application. In that case, data must be forwarded to the target application using an interprocess communications (IPC) mechanism; this forwarding of data is commonly referred to as hopping. The cost (in terms of the processor’s processing time) of this hop is roughly equal to the cost of processing TCP/IP data in the first place. Hence, with traditional clustering, you pay roughly two times the TCP/IP cost to get data to the application. By contrast, with SMP-based systems, because of their shared memory, you can bring in data with one step and no hops to the application-accessible memory, regardless of which processor is running the application. So the cost of data transfer in SMP is $1.x$ times the single processor TCP/IP cost, where x represents additional overhead because of spin locks.

Distributed computing solutions and limitations

Another commonly used technique for scaling is distributed computing, such as Web serving. With the distributed computing approach, typically, multiple servers are configured to perform the same task in parallel. This technique, coupled with some form of external load balancing, enables good scaling, as long as the services being provided lend themselves to replication. It requires that the data be replicated on each of the servers to which the load is being distributed.

There is, of course, the management headache of keeping the services and data replicated on each server. Also, managing a “server farm” when the scale becomes quite large can require significant staff. Nevertheless, for the Web server example, the Web pages being served would be replicated in each server, and a load balancer could simply distribute incoming requests across the servers. The load balancer could even monitor the time the various servers take to respond to the incoming SYN (connect request), determine to some degree the relative loads on each server, and adjust future redirection accordingly.

More sophisticated load balancers also allow some form of “biasing” of servers to be configured by a system administrator. This technique is adequate as long as the data being served is fairly static and the service patterns are consistent. But once you have a more dynamic environment in which the data is not static but rather is generated dynamically, or transactional in nature, or of a scale that can't be served by a single server, this technique begins to break down. If the database involved must be larger than a single server can handle, the database can be partitioned. However, if the load balancer sends the Web request to the server with the wrong partition, the request must somehow be redirected to the server with the correct partition. This forces the equivalent of a hop, which negatively affects performance.

If the load on a server is highly variable, the recent history of response time is not a good predictor of current or future loads. Load balancing transactional systems is complex and should not be underestimated. Certainly a simple load balancer cannot perform the function with little insight into what is going on in the servers. This would result in the need to scale each server to a capacity appropriate for the worst-case load possible as well as to scale the server farm similarly. To avoid service outages, therefore, the service provider must pay for the worst-case computing needs. Unfortunately, many websites have not done this; such omission has resulted in widely publicized website failures.

Although distributed computing solves some of the limitations of shared-memory SMP computing, it introduces management problems in highly scaled applications and can introduce the equivalent of a hop, which negates the advantage of SMP.

NonStop TCP/IP_{v6} solutions

Wouldn't it be nice to be able to preserve the scaling benefits achievable with cluster-based systems, and yet get the zero hop costs associated with SMP-based systems? Wouldn't it be nice to be able to get the simple scalability that distributed systems offer without the management headaches, lack of flexibility, or inability to handle truly massive databases?

With NonStop TCP/IP_{v6} software—a parallel processing, single IP product—HP breaks with the traditional clustering paradigm and routes data to the correct application process in the correct processor in the first place, eliminating the interprocess hop that placed the clustering architecture at a performance disadvantage to SMP. By taking advantage of the ServerNet architecture's capacity to link peripheral I/O controllers to all processors in the system or cluster, we can access LAN adapters (network interface cards, or NICs) directly from any processors in the system or cluster.

For outbound data, we can then transfer data directly to the LAN adapters. Similarly, for inbound frames we replicate a portion of the TCP layer in the LAN adapter and perform inbound “filtering” to route incoming packets directly to the memory of the processor in the system or cluster containing the application process that is the target for the data.

This use of host-controlled filtering is unprecedented, and U.S. patents 6359886 and 6370583 have been granted for it. The filtering method retains the scaling and manageability properties of a clustered system without the traditional hop penalties. Indeed, in the absence of the spin-lock overhead of SMP systems, processing costs drop to one times that of TCP/IP processing.

Single IP

In Conventional TCP/IP, if you run multiple process instances of a listening application in multiple processors (to increase computing power), you need a different TCP/IP process (one per listening application process instance) in each processor. Each of those TCP/IP processes requires a unique physical port and presents a unique IP host to the outside world. By contrast, with NonStop TCP/IP_{v6} software, it is possible to run multiple process instances of a listening application in multiple processors, all sharing the same port and presenting a single IP host to the outside world.

Reactive distribution

Reactive data distribution, which occurs directly from the adapter to the processor, is different from typical load balancers in that it is host-processor directed. With reactive distribution, the load follows the application processes as they create TCP/IP “sockets” and issue requests against them. This form of data distribution prevents the customer from having to pay for worst-case computing and database resources, which are required by any blind, static distribution technique.

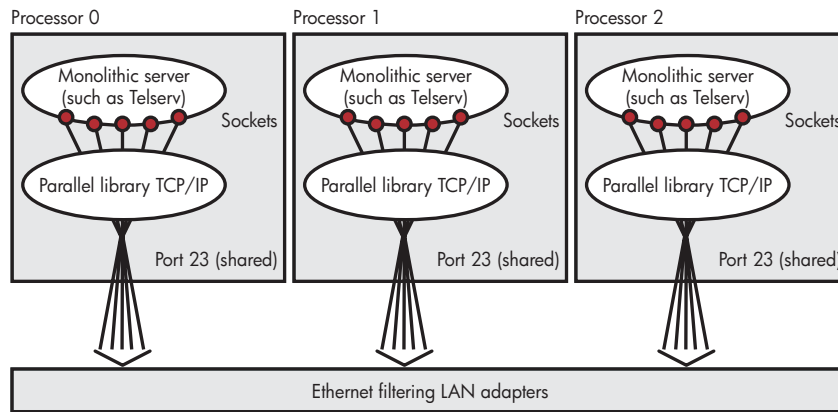
Round-robin distribution

One final concept to understand before discussing the potential applications for NonStop TCP/IP_{v6} software is round-robin distribution. This is a method in which the adapters route incoming connection requests to applications that are replicated across multiple processors and that share the same port. This distribution occurs in reaction to application socket-layer requests, thereby closely coupling the distribution to application activity.

Improved fault tolerance

Another aspect of the NonStop TCP/IP_{v6} product is improved fault tolerance. Through its parallel architecture, with TCP/IP potentially running in every processor in the system or cluster, failure of a processor does not affect the availability of TCP/IP access in the surviving processors.

Figure 1. Monolithic server model.



Using parallel library TCP/IP for different networking applications

Applying NonStop TCP/IP_{v6} and ServerNet architectures to common networking applications can provide solutions to problems in the Internet marketplace such as scale, interconnectivity, and reliability. For the purposes of this discussion, we will group networking applications running on the NonStop Kernel operating system into five architectural models:

- Monolithic server model
- Listener server model
- Distributor server model
- Hybrid server model
- Broker server model

Many other models are possible. However, for the sake of simplicity and because the design principles described here can be extended to apply to other models, this paper discusses only five architectural models.

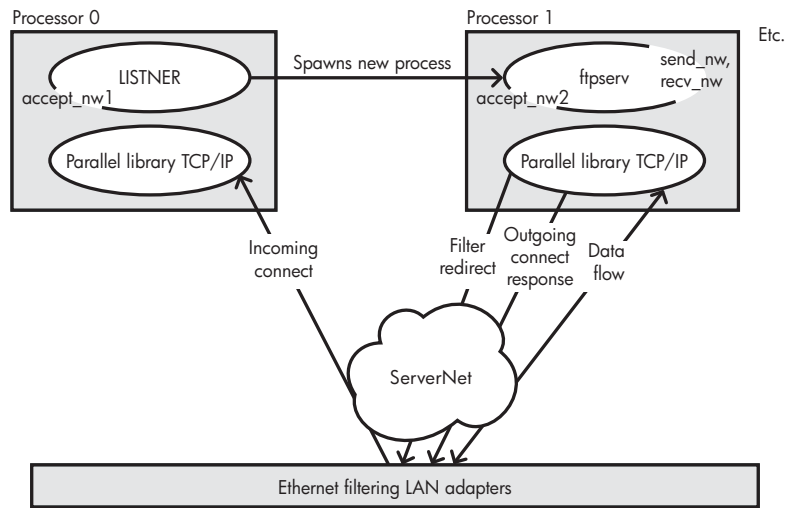
The networking models discussed represent approaches to handling large network application problems. The scales of these problems have the characteristics of many thousands to millions of users, enterprise databases (terabytes), millions of transactions a day, and tens of thousands of simultaneous TCP/IP connections. Also, these are “server” models—that is, ones for which the dominant form of communication involves the server “listening” passively for incoming connections, rather than initiating connections of its own. In the server model, in most cases the clients initiate connections to a “well-known” port number in the TCP protocol.

Monolithic server model

The monolithic server model places all network application logic in a single process (see figure 1). This single process internally manages the multiple contexts associated with many connections by using some form of context switching.

Typically, a single process implementation may have many constraints on scalability, ranging from operating system kernel resource limits to memory limits. In addition, in a cluster-type system without SMP capability (like NonStop servers), there is a single processor limit to how much work the monolithic process can do. Even on an SMP-type system, if the monolithic application does not use threading to manage the multiple contexts and use appropriate locking mechanisms for an SMP environment (that is, SMP-safe spin locks), the application cannot scale. Furthermore, given that the monolithic application generally listens on a single TCP port number, you could not run multiple copies of the monolithic application on the NonStop Kernel operating system because port sharing is not allowed with the Conventional TCP/IP product.

Figure 2. Listener server model.



A good example of the monolithic model running on the NonStop Kernel operating system is the Telserv product. Telserv handles many incoming TCP connections on Port 23. However, there can be only one instance of Telserv running in the system for a single TCP/IP host when using the Conventional TCP/IP product for the NonStop Kernel operating system. You could run multiple TCP/IP stacks in the Conventional TCP/IP environment, but that would require multiple IP hosts, presenting the image of separate systems of the server to clients and resulting in extra configuration effort, among other things.

Telserv, combined with a single instance of Conventional TCP/IP, is a potential bottleneck and doesn't scale. By contrast, NonStop TCP/IPv6 software allows multiple copies of Telserv to run, all sharing the same port number (23) and presenting a single IP host, using round-robin distribution. Then each Telserv process can run in a different processor in the NonStop Kernel cluster. With round-robin distribution, NonStop TCP/IPv6 software distributes incoming connection requests, in turn, to each instance of the Telserv, as the Telserv process creates new sockets, binds Port 23 to the socket, and then listens and accepts to get the connection. In this case, NonStop TCP/IPv6 software allows parallel Telserv processing and avoids overloading a single processor. It also enables a high degree of flexibility, allowing a system manager to spread the load by distributing the Telserv processing across many processors in the cluster.

Listener server model

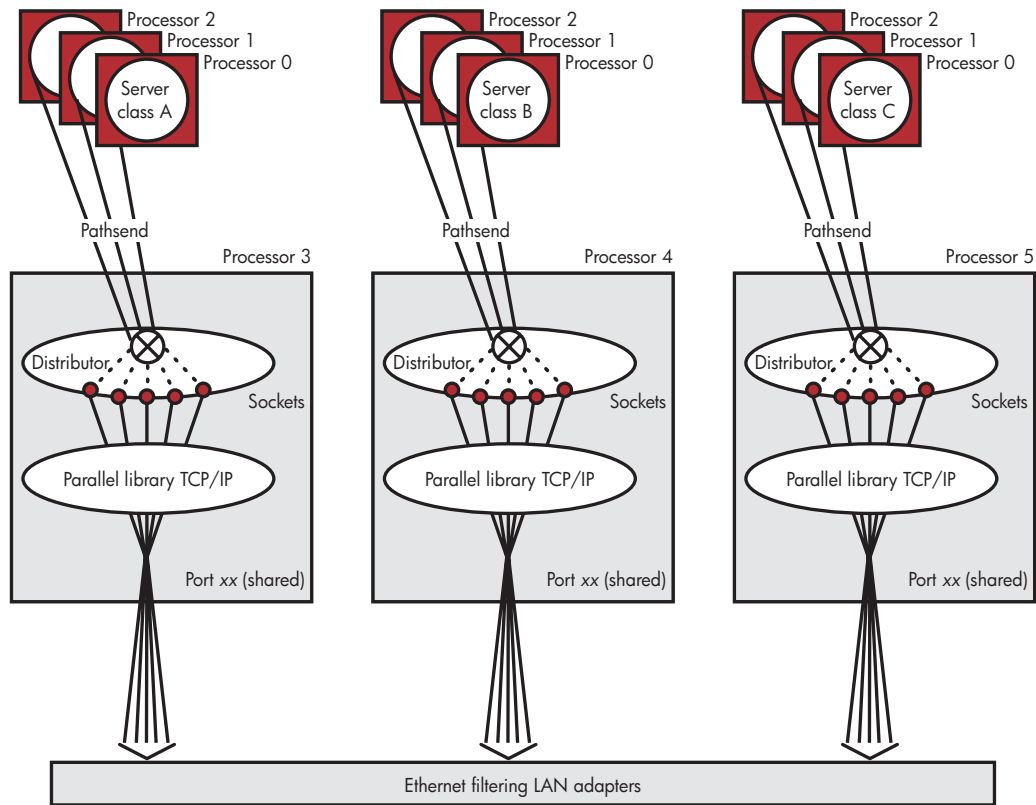
The listener server model is similar to the one used in UNIX[®] system-based servers in the form of the "inet daemon." This model listens on a well-known port number (or several port numbers) and handles many connections (see figure 2).

However, rather than performing the processing within the listening process, it spawns a new process for each incoming connection (or uses IPC to pass information to one of a set of pre-spawned processes) and hands off the newly created connection to the other process.

In the NonStop Kernel operating system, a good example of this kind of processing is the LISTNER process. This process creates a socket, binds the well-known port number to it, issues an `accept_nw1` to get the incoming connection information, and passes that information to a new process that it creates (which can be in a different processor in the cluster in the NonStop Kernel operating system). The newly created process then creates a new socket, binds the connection information (source IP and source-port number) to it, then issues an `accept_nw2` to migrate the connection to the new socket. The newly created process (for example, an instance of `ftpserv`) can then utilize the socket with the new connection to perform work completely in parallel with incoming connection processing by the LISTNER process. This is a highly scalable model, allowing parallel instances of the "worker" processes running on independent processors.

The standard listener model can benefit from the NonStop TCP/IPv6 architecture because this architecture eliminates the hop that occurs between the spawned server process and the processor where the TCP/IP process resides. That hop occurs on the server process's first socket call and on every subsequent send/receive call. In NonStop TCP/IPv6 software, those calls are handled locally within the server's processor. By taking advantage of reactive distribution, you eliminate that processor hop, improving performance for the standard listener model.

Figure 3. Distributor server model (NonStop TCP/IPv6 architecture).



Distributor server model

The distributor server model uses an interface process that serves as a distributor by using NonStop Kernel IPC to talk to multiple back-end server instances (see figure 3). In this model, the distributor binds to a well-known port and then accepts multiple connections by creating a socket for each connection. The distributor then performs the accepts, sends, receives, and so on, on those sockets on behalf of the back-end servers. It handles all data and control, and forwards the received data to back-end server processes using NonStop Kernel IPC. A good example of this model is a TCP/IP application developed for the Pathway environment.

The Pathway environment provides a rich set of server process management and load-balancing facilities. A front-end distributor process using a set of verbs that we collectively refer to as *Pathsend* communicates with the back-end servers. Pathsend allows load-balancing algorithms to distribute data to the back-end server instances.

The distributor model achieves some parallelism and load balancing because of its use of the multiple, back-end server instances. However, this model is limited because all data must flow through the distributor to the back-end server processes through Pathsend. This creates a potential bottleneck in the distributor.

Figure 4. Distributor server model (Conventional TCP/IP).

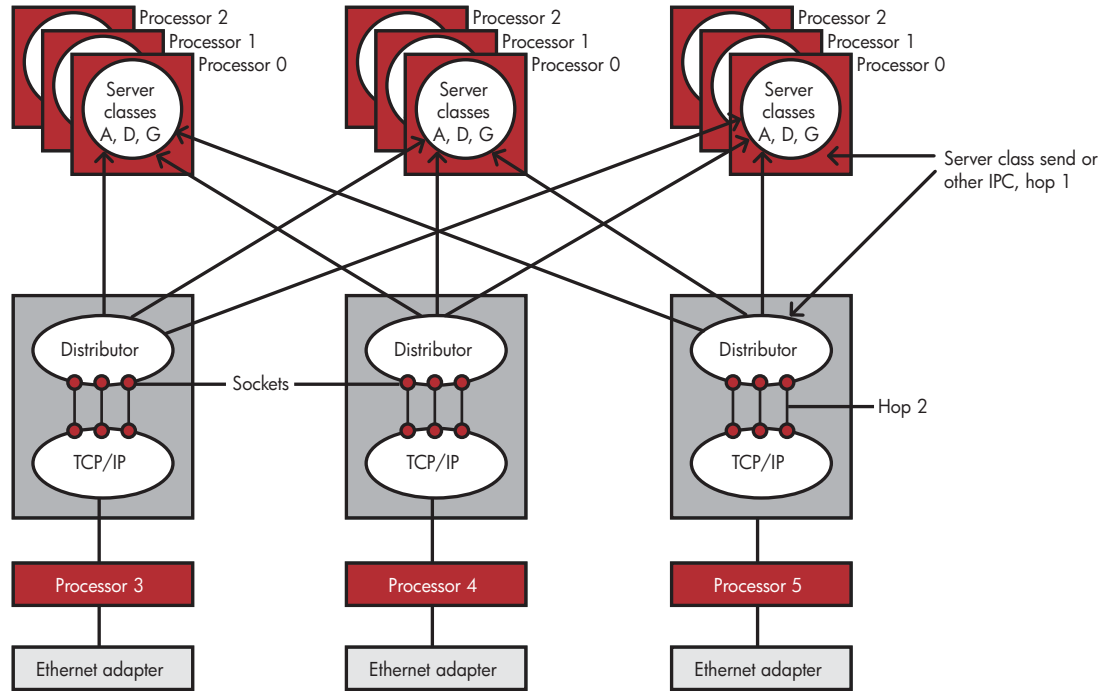


Figure 4 shows the distributor server model in Conventional TCP/IP. To accommodate growing computing needs in this Conventional TCP/IP example, three instances of the distributor are running in three processors (3, 4, and 5), distributing connections to three sets of server instances in three different processors (0, 1, and 2). Although you can achieve scalability in the Conventional TCP/IP model by running multiple distributors in this manner, two interprocess hops are involved in the data flow: one between the distributor and the server, and one between the distributor and the TCP/IP process. In addition, each processor must have its own physical interface (PIF) on the adapter. Finally, having run a TCP/IP process for each distributor, each distributor appears to the outside world to be on a different IP host.

The distributor server model can benefit from the NonStop TCP/IPv6 architecture in two ways:

- This architecture lets you run multiple distributor processes in multiple processors bound to the same port with round-robin filtering enabled. This allows you to spread the distributor's workload over as many processors as required, achieving unlimited scalability while presenting a single IP host image to the outside world.
- The architecture shortens the path length for data flow by eliminating the hop between the distributor and the TCP/IP process. You also eliminate the hop between the servers and the TCP/IP process, thereby improving performance.

Figure 5. Hybrid server model (Conventional TCP/IP).

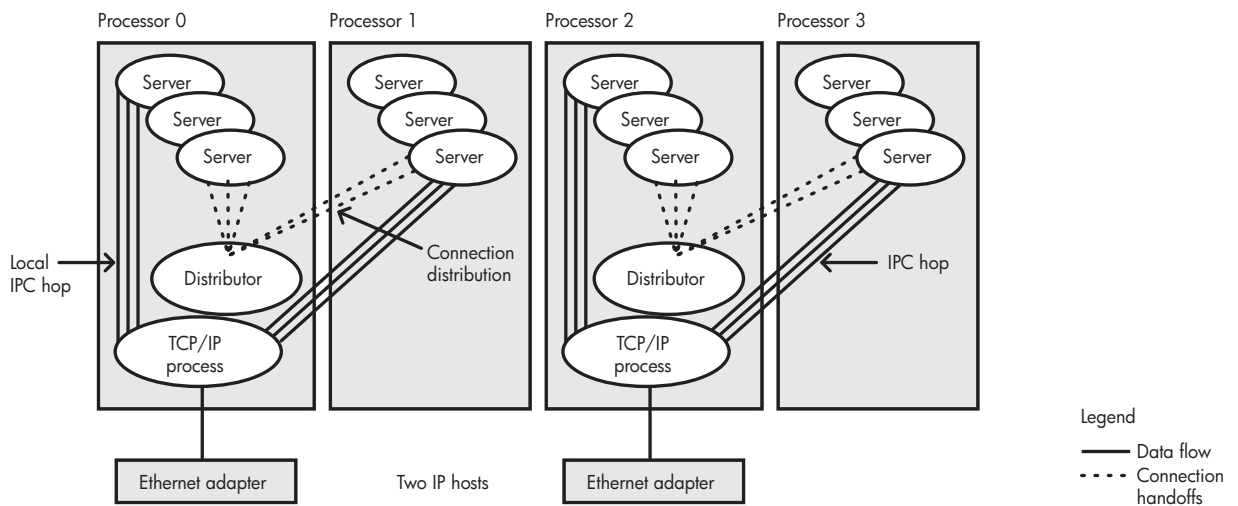
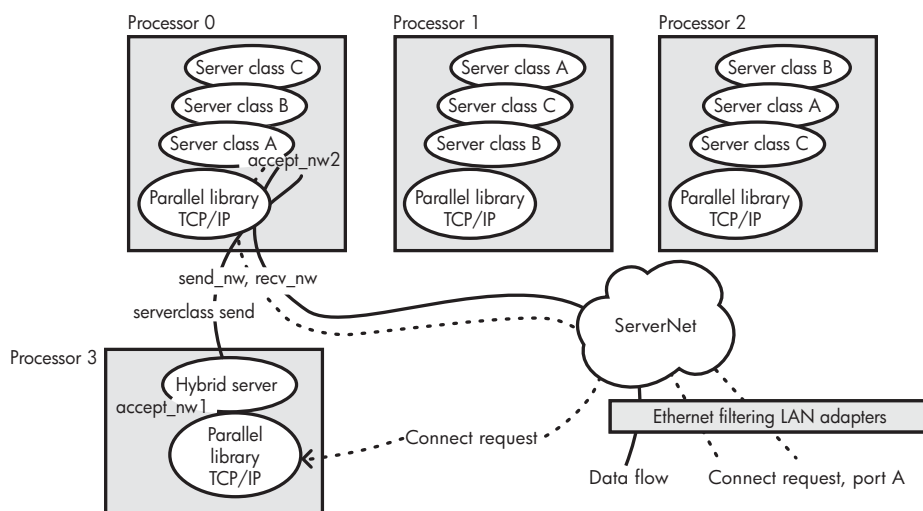


Figure 6. Hybrid server model (NonStop TCP/IPv6 architecture).



Hybrid server model

This model still has a distributor process using Pathway for managing and load balancing server process instances, but it uses the standard listener approach, handing off connections so that data does not have to flow through the distributor process. This model decouples the data and control flows, which facilitates low-overhead scalability when combined with NonStop TCP/IPv6 software.

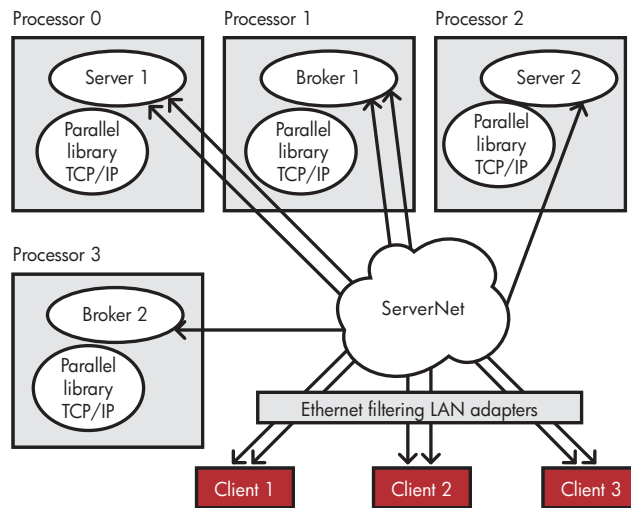
Figure 5 shows the hybrid server model in Conventional TCP/IP, with two distributors in Processors 0 and 2 distributing connections and data and control flow to servers in their own and one other processor. The servers inherit the connections and send data flow to the TCP/IP process. Note that each TCP/IP process represents a different IP host. Also note the remote IPC hop between the servers in the remote processors (1 and 3) and the processors with the TCP/IP stacks (0 and 2). In addition, note the local

IPC hop between the servers and the TCP/IP processes in the processors containing the TCP/IP stacks.

The hybrid server model can benefit from NonStop TCP/IPv6 software because this new architecture allows round-robin filtering and running multiple instances of the hybrid distributor in different processors bound to the same port. Round-robin filtering enables the adapter to distribute incoming connections to the multiple distributor instances for connection parallelism; then the distributors hand off the connections to available servers, allowing reactive distribution. The servers have direct access to the TCP/IP library in their own processors, eliminating both remote and local IPC hops. They all share the same port and present a single IP host image to the outside world.

Figure 6 shows a simplified configuration with one distributor in Processor 3. The distributor hands off connections to servers in Processors 0, 1, and 2. All connections go

Figure 7. Broker server model.



through the same Ethernet adapter PIF. Incoming connections are processed by the distributor, which creates sockets, binds the listening port to them, and listens on them. The distributor then issues an `accept_nw1` to get the incoming connection and retrieves the connection information. The connection information is passed (using Pathsend verbs) to a server class instance. The server class instance then creates a new socket, binds the connection information passed to it, and issues an `accept_nw2`, which transparently causes NonStop TCP/IPv6 software to migrate the connection to the new socket in the other processor. A single IP host image is presented to the outside world, and the interprocess hop between the remote processors (0, 1, and 2) and Processor 3 (containing the distributor) is eliminated as data flows directly to the TCP/IP library in its own processors out to the adapter.

The hybrid server model with NonStop TCP/IPv6 software is highly scalable, because it allows the main work of server processing to go on in parallel with new connection establishment. The distributor works together with the rich and powerful load-balancing capabilities of the transaction-processing monitor to balance service processing across the processors making up the NonStop Kernel cluster operating system. Note that you could also run multiple copies of the distributor, if connection processing remained a bottleneck, by using port sharing and round-robin distribution.

In summary, the hybrid server model can benefit from the NonStop TCP/IPv6 architecture in several ways:

- This architecture lets you run multiple distributor processes in multiple processors bound to the same port with round-robin filtering enabled. This allows you to spread the distributor's workload over as many processors as required, achieving extensive scalability while presenting a single IP host image to the outside world.

- The architecture shortens the path length for data flow by eliminating the hop between the distributors and the TCP/IP process. It also eliminates the hop between the servers and the TCP/IP process.
- The architecture also eliminates the extra hop from the distributor to the servers because of the connection handoff. You can also run multiple copies of the servers, allowing you to spread the server's workload over as many processors as required. This allows you to have extensive scalability while presenting a single IP host image to the outside world.
- By taking advantage of reactive distribution, you get scalability and better performance.

Broker server model

The broker server model has become common, with the advent of multi-tiered server architectures, especially within the distributed object paradigm (see figure 7). With this approach, the initial connection is from the client to a broker. The broker participates in a back-end load distribution algorithm, which may be distributed. The broker decides which of a family of server objects is the best one to serve the object request, and it responds to the client with the address of the server to contact. The client then makes a new connection to the server to complete the object request. In some cases, the respective connections are cached for efficiency.

In figure 7, Client 1 has contacted Broker 2 (running in Processor 3), has been directed to contact Server 1 (running in Processor 0) for service, and is exchanging service requests and getting service responses from Server 1. Similarly, Client 2 has contacted Broker 1 (running in Processor 1).

Due to the single IP image of the NonStop TCP/IPv6 architecture, Broker 1 is on the same IP host as Broker 2, which simplifies client configuration. Broker 1 also has directed Client 2 to contact Server 1. However, Client 3, which has contacted Broker 1, has been directed to contact Server 2 (running in Processor 2). Of course, we could also have been running more server process instances (in the same or different processors), as well as brokers, depending on the load and application requirements.

With NonStop TCP/IPv6 software and reactive distribution, you do not need to worry about process placement or "affinity" to processors. Note the high degree of parallelism, flexibility, and scalability that this model affords.

The broker model can benefit from the NonStop TCP/IPv6 architecture in several ways:

- The architecture lets you run multiple broker processes in multiple processors bound to the same port with round-robin filtering enabled. This allows you to spread the broker's workload over as many processors as required, achieving unlimited scalability while presenting a single IP host image to the outside world.
- The architecture shortens the path length for data flow by eliminating the hop between the broker processes and the TCP/IP process, as well as between the server processes and the TCP/IP process, thereby improving performance.
- The architecture lets you run multiple copies of the servers, allowing you to spread the server's workload over as many processors as required. This allows extensive scalability while presenting a single IP host image to the outside world.
- By taking advantage of reactive distribution, you get scalability and better performance.

Conclusion

NonStop TCP/IPv6 software is a major enhancement to the NonStop system's ability to serve the demands of the Internet world. A new and unique product, NonStop TCP/IPv6 software takes advantage of the architectural features of NonStop servers, including ServerNet technology, to increase the network bandwidth that can be processed with a single IP host image.

Parallel processing is a fact of life in today's server offerings from most vendors, but the architectures of these systems have limitations that make growth too difficult or expensive. Fault tolerance, responsiveness to extreme performance requirements, and the ability to react quickly to rapidly changing demands from the Internet are some of the critical factors to the viability of Internet companies and those companies that are increasingly affected by the Internet and its technologies.

For these companies, meeting those demands has been no small task. NonStop TCP/IPv6 software running on NonStop servers now makes that job easier. The load-balancing scenarios examined in this paper illustrate how applications can place their processing modules in different processors without paying a penalty for the allocations. By adding this software to your NonStop server, you can enhance every networking application architectural model, adding parallel processing without requiring the world to interact with multiple IP hosts on your system. Reactive distribution lets the server application processes themselves control responsiveness to workload and data flow. At the same time, NonStop TCP/IPv6 software eases the burden on the computing resources of the clustered system while enhancing fault tolerance through additional parallelism. This software delivers exceptional performance in one of the key networking technology facets of today's business-critical applications.

For more information, go to www.hp.com/go/nonstop.

© Copyright 2003 Hewlett-Packard Development Company, L.P. UNIX is a registered trademark of The Open Group. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

5981-8449EN, 05/2003

