The Evolution of High-End Router Architectures

Basic Scalability and Performance Considerations for Evaluating Large-Scale Router Designs

Introduction

The routers that power the Internet are evolving architecturally to keep pace with the escalating use of the Web. The tremendous growth of the Internet plus a whole new generation of innovative, revenue-generating application services is placing unprecedented performance demands on the Internet infrastructure. As a result, sophisticated new distributed router designs have emerged to meet the corresponding technical challenges in ways that also give service providers the ability to quickly scale their networks and bring new services to market.

E-commerce, voice over IP (VoIP), virtual private networks (VPNs), Web-based customer relationship management (CRM), streaming video and audio, videoconferencing, and a host of other Internet Protocol (IP)-centric network services are driving the need for high-bandwidth and traffic-prioritization capabilities. As a result, network service providers, Web hosting companies, and other large companies are demanding the following attributes from their IP network infrastructure equipment:

- The ability to harness the generous network capacity afforded by the latest fiber-optic technology for transporting large volumes of user traffic.
- The ability to scale networks quickly and cost-effectively with minimal impact on network operations as customer bases and traffic volumes grow.
- Capabilities for minimizing latency, jitter, and packet loss, thus enabling the successful support of delay-sensitive applications such as VoIP and real-time video; in addition, ensuring the delivery of packets in their proper sequence and being able to create differentiated classes of service by prioritizing traffic on a packet-by-packet basis are critical performance criteria.

To satisfy these requirements, certain high-end router architectures now support ultrafast fiber-optic interfaces of up to 10 Gbps speeds and achieve system throughput in excess of 350 million packets per second (pps). They achieve this, in large part, by distributing the processing loads associated with packet forwarding across modular line cards. Such distributed architectures yield several investment and performance benefits. For example, they enable service providers to scale their network capacity within a single router chassis as traffic volumes increase without a corresponding drain on central processing resources. This avoids throughput degradation caused by bottlenecks in the centralized processor.

Distributed architectures are optimized to preserve service providers' equipment investments and to protect against performance degradation as networks grow and users generate more packets for the router to process. Conversely, in router designs where the packet-forwarding function shares centralized processing resources, network performance can eventually suffer as traffic volumes increase. This is because bigger processing loads must contend for a single, finite pool of resources.

What follows is a basic discussion of key high-end router architecture alternatives and considerations. The purpose is to educate networking professionals with responsibility for the evaluation, recommendation, and/or selection of networking equipment about some of the architectural trade-offs among the primary types of router designs. This information is intended to help the reader formulate a checklist of points to consider when researching the attributes of large-scale equipment that will be expected to support gigabit-speed interfaces and rapidly increasing traffic volumes.

Router Basics

A router has three fundamental jobs. The first is to compute the best path that a packet should take through the network to its destination. This computation accounts for various policies and network constraints. For example, user policy could dictate that the network path should maximize network efficiencies, deliver the fastest possible response times to users, minimize bandwidth usage costs, or meet some other set of criteria.

The second job of the router is to actually forward packets received on an input interface to the appropriate output interface for transmission across the network. Forwarding relies on the best-path information precomputed in the route processor. The third major router function is to temporarily store packets in large buffer memories to absorb the bursts and temporary congestion that frequently occur and to queue the packets using a priority-weighted scheme for transmission.

The basic system components in charge of carrying out these three functions are *the route processor, forwarding engine*, and *buffer memory system*.

Route Processing

Routers determine best paths by sharing information about network conditions with neighboring routers. The route processor—a network card outfitted with a processor and memory—controls this function. The route processor is, in effect, the "brains" of the router and is dedicated to communicating with neighboring routers. This communication enables route processors to build a comprehensive route database, or routing table, that enables the forwarding engine to send packets across optimum paths through the network.

In router architectures today, route processing is a centralized function. The reason for this is that having a single repository for all updated routing-table information for each system significantly reduces system complexity. It is not necessary to scale route-processing resources in direct proportion to the speed of the line cards being added to the router in order to maintain system throughput. The reason is that the route processor performs no functions on a packet-by-packet basis. Rather, it updates and consults routing tables, and its timing is independent of the packet-forwarding process.

To optimize overall system and network uptime, however, it is often advisable to configure routers with redundant centralized route processors. Doing so simply involves plugging a second, identical route-processing card into a second slot in the router and configuring the system so that the primary route processor fails over to the backup in the event of an outage on the primary card.

The route processor runs routing algorithm software containing routing protocols, which enable the sharing of network status information among neighboring routers. Common routing protocols in use include Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS), and Routing Information Protocol (RIP).

Packet Forwarding

Except for the single slot in a router used to house the route processor (or two slots, in the case of a redundant configuration), all other slots can hold network line cards that forward packets from one network interface to another. A forwarding engine controls this function by consulting a Forwarding Information Base (FIB), which is a summary of the routing tables created by all the route processors in the devices throughout the network. Based on destination address information in the IP packet header, the forwarding engine consults the FIB to select the appropriate output interface and forwards the packet there.

As traffic loads grow, it follows that the amount of processing power required for greater volumes of FIB lookups also increases. Newer, distributed system designs have come into play in routers that must handle packet forwarding in volumes that surpass what a single, centralized, and shared forwarding engine can reliably support. The section “Switch Fabric Architectures” discusses distributed router designs in detail.

Packet Buffering

As mentioned, the router requires a buffering system to do its job. In fact, because of the predominant use of the bursty Transmission Control Protocol (TCP) transport protocol, one of the most important requirements for a router used in the core of the Internet is support for a significant amount of buffering. Each input interface on a line card receives packets that are examined by a forwarding engine and directed to the output interface associated with its destination network. If multiple packets arriving on different interfaces need to be forwarded to the same output interface simultaneously, a buffer must be available as a temporary waiting area in which packets queue up for transmission. The order in which they are transmitted is determined by the policy settings configured by the network administrator.

Without sophisticated queuing options, packets simply line up and are transmitted in the order in which they are received. For many data applications, this approach is not a particularly troublesome issue. The reason is that the performance of file transfers, Web browsing, and other applications that can tolerate some delay is not affected by small amounts of latency and jitter (the degree to which latency varies). However, delay-sensitive applications, such as VoIP, require more sophisticated queuing capabilities, which will be discussed later.

Alternative Architecture Designs

There are two fundamental approaches to building a large-scale router: 1) *centralized shared-memory and forwarding architectures* and 2) *switch fabric architectures with distributed buffer memory and forwarding* capabilities.

Historically, routers have relied upon centralized, shared-memory designs for buffering packets. In such designs, all packets received from all interfaces are written to a common memory pool. They are then read from this shared memory and sent to an output interface.

The very high interface speeds and throughput required for Internet-scale backbone routers, however, are driving the need for more distributed router designs that relieve the line cards from having to contend for a single, shared pool of memory and other processing resources. Distributed architectures divide up router functions (in varying ways, depending on the specific router design) and house them in discrete physical line cards that plug into a router core chassis. These modular line cards are placed into an enclosure that provides common services such as power, cooling, and management to all the modules.

In a distributed design using a switching fabric, each line card has two major memory systems of its own for packet buffering. A *receive memory* stores packets received from an interface, and a *transmit memory* stores packets ready for transmission to their destination across the output interface. Each line card also has a copy of the route processor FIB so that it can look up precomputed best paths for packet forwarding.

Let's take a closer look at the two main architectures as they have evolved, the various ways to implement them, and the respective advantages and disadvantages of each option.

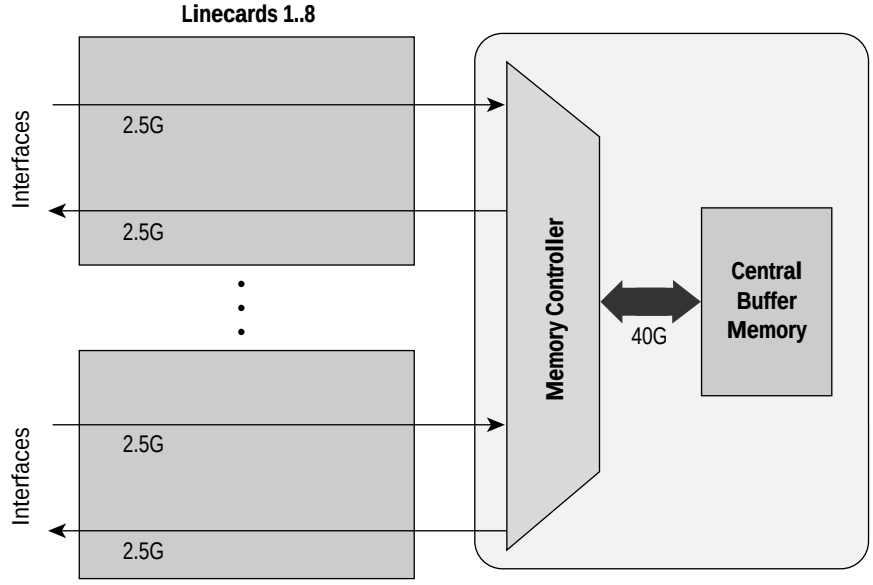
Centralized, Shared-Memory Architectures

Centralized, shared-memory architectures represent a common—but aging—generation of router design. Two basic approaches have been implemented with shared-memory router architectures: *physically centralized shared memory and logically centralized shared memory.*

Physically Centralized Shared-Memory Architecture

This design approach requires all packets to pass in and out of a single, large, physical memory system. (See Figure 1.) Having a central control point keeps memory management simple and contained, making this architecture advantageous. However, for large-scale systems, this design carries the expensive drawback of requiring an extremely large, complex, and power-hungry memory system.

Figure 1 Physically Centralized Shared-Memory Architecture: This design has the advantage of simplified memory management. The downside is that the architecture requires an extremely high-speed memory system with associated complexity.



From a management and complexity perspective, the physically centralized shared-memory architecture is theoretically ideal. However, it is impractical in very large-scale implementations (systems supporting aggregate bandwidth above about 20 Gbps). Among the reasons is that the extremely high volume of bandwidth required to support the memory system requires a corresponding number of parallel memory devices. Building memory systems to this scale introduces major inefficiencies.

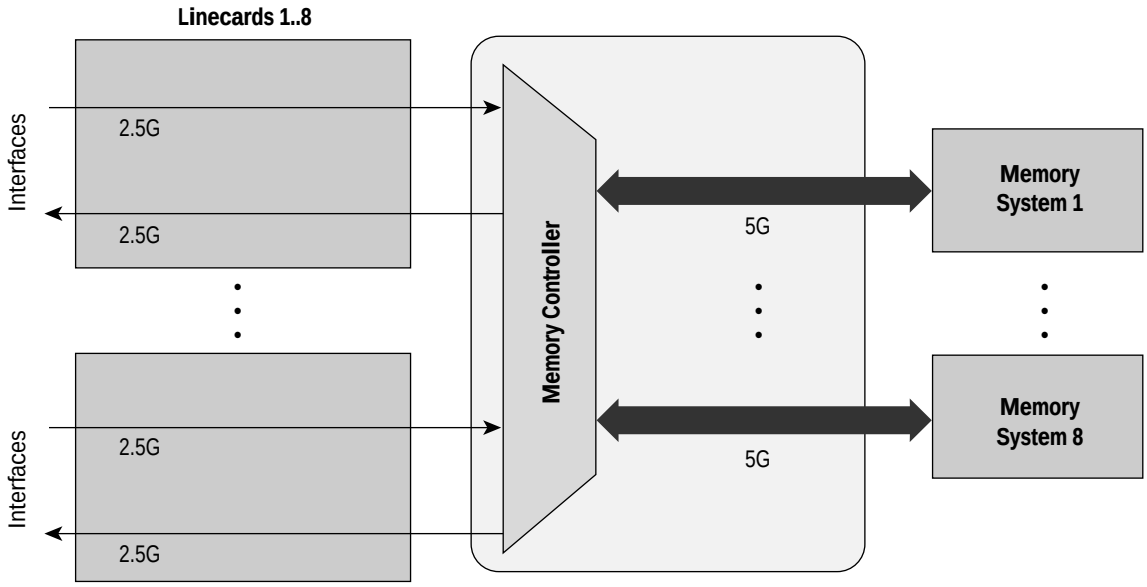
Aside from adding system costs, performance is ultimately reduced because of these inefficiencies. If a lower-speed memory system is used, however, contention for centralized resources will eventually cause performance degradation as traffic loads increase. Centralized, shared-memory architectures, then, are appropriate design choices only for routers with relatively low aggregate connection speeds to the buffer memory system.

Logically Centralized Shared-Memory Architecture

Another approach to implementing a shared-memory system—which is far more practical to build—is to break the main memory into multiple smaller pieces and then physically apportion the pieces across different components in the router. The most convenient approach is to place each memory on one of the system line cards. (See Figure 2.) Though the memory is physically dispersed in this scenario, the router continues to logically treat the sum of the pieces as a single, shared memory driven by a common memory controller. Data is logically written to and from the single, main memory, but the memory controller divides the data evenly between each physical memory.

The rationale behind this approach is that spreading the packet-buffering workload over multiple memory systems results in each memory system having a manageable amount of associated bandwidth, while the sum of the connection speeds is sufficient to handle total system requirements. However, the issue of contention for shared-memory resources as a system scales in capacity remains a scalability stumbling block in high-end systems.

Figure 2 Logically Centralized Shared-Memory Architecture: Logically centralized shared-memory architectures alleviate the memory scalability problem associated with physically centralized designs. However, in large-scale implementations, they introduce system inefficiencies, data blocking, and packet-sequencing problems.



In logically centralized shared-memory designs, it is common to break each packet into small, fixed-sized cells and to write each cell to a different memory system using a "fair," round-robin distribution approach. Doing so optimizes the parallel use of the entire combined bandwidth of the system by avoiding some blocking of smaller packets that can get "stuck" behind larger packets. This reduces latency in the system, constituting the main advantage of the cell-based design.

A disadvantage to any memory system using a cell-based approach is that inefficiencies crop up when a packet is slightly larger than the size of a multiple of the cell size.

When a packet is divided into equal-length cells, the final cell will rarely be filled efficiently with data. For example, if the cell size used is 64 bytes, a packet of 65 bytes will require two cells. This means that 128 bytes of memory bandwidth will be needed to move 65 bytes of actual data, resulting in an efficiency rate of just over 50 percent.

Another disadvantage to consider here is that each fixed-length cell must have a link associated with it, a scenario that allows the multiple cells to recombine to form an entire packet. For a memory system of the size required for an Internet backbone router, the number of cells will be very large. This means that the additional storage required for links can be a challenge to implement, both in terms of the amount of storage required and the bandwidth needed to access this data.

Another consideration in this scenario is that because packets are chopped into cells and distributed evenly to all memories, each piece of memory becomes a potential single point of system failure. For example, removing a line card would cause the system to drop the packets stored not only in the memory of the line card removed, but also in all other system memories. Traffic traversing fully functioning line cards, then, is penalized for the downtime associated with a single line card. Retransmissions are required for all packets in all memories, resulting in degraded performance.

Note that sustaining service levels for IP multicast traffic is particularly problematic in any shared-memory architecture. The reason is that each multicast packet is written to a shared memory just once, but needs to be read multiple times by different line cards, all from the same address. This situation increases the chances of other packets getting blocked. On the other hand, crossbar switch fabrics (discussed later) use distributed processing and can replicate the same data to many line cards simultaneously to overcome this obstacle.

Packet Sequence Integrity and Related Scalability Issues

Shared-memory designs can be prone to packet-sequencing problems when scaled to greater capacities. Sequencing problems—whereby packets arrive at their destinations out of order—occur, for example, when the equipment vendor uses a scalability approach that involves folding multiple shared-memory systems into a larger system to create a single, larger-capacity router. In such cases, the multiple routers that have been joined together have not been designed as a single unit with cohesive *packet-distributing* and *packet-resequencing* functions. Resequencing is required in this type of design approach to scaling centralized shared-memory routers because IP packets can arrive at an output interface in any order and must be properly reordered before being delivered to user systems.

In large-scale shared-memory architectures, the packet distributor is tasked with receiving inbound packets from a physical interface and deciding which of the multiple integral “routers” to send them to. It generally chooses the system using simplistic round-robin algorithms or similar approaches. The packet resequencer will receive packets from each of the independent systems and forward them in the correct order to the output interface. But packets traveling through the independent internal systems will arrive at their common destination line card at different times. Among the reasons are varying degrees of load and congestion in the separate forwarding and memory units within the system, as well as differences in processing time for packets of varying length. This situation makes forwarding the packets to the output interface in the correct order much more difficult to achieve.

The packet-sequencing problems in shared-memory systems often occur because the system is designed to attempt—at all costs—to sustain a certain throughput speed to avoid *head-of-line (HOL)* blocking. HOL blocking occurs when there is data on its way to an output interface that has transmit memory bandwidth available but is blocked by other data that must wait for transmit memory to become available on the particular output interface to which it is being directed. If HOL blocking or other delays occur, and a particular packet in a sequence has not yet arrived for transmission, the system will likely wait only a minimum specified time before transmitting the other packets in the sequence to sustain system throughput goals.

This situation is fatal to applications that cannot tolerate out-of-sequence packets, such as VoIP and video, which run over the User Datagram Protocol (UDP). UDP does not provide acknowledgements of packet delivery, so retransmissions do not occur if packets are dropped or reordered. End users sending real-time traffic will notice the degradation of the transmission in these cases; in fact, transmissions will often be incomprehensible.

For TCP applications, this setup causes network inefficiencies by making a large number of packet retransmissions necessary, a scenario that degrades throughput.

Because it causes available system resources to sit idle, HOL blocking results in performance degradation that grows proportionally larger as traffic volumes increase.

Switch Fabric Architectures

A switch fabric is a mechanism for allowing each line card (with its own forwarding engine) to transmit data to any other line card as needed. As mentioned earlier, in the switch fabric architecture, each line card contains both a receive memory and a transmit memory. In addition, each has a connection to the system switch fabric. (See Figure 3.)

Nonblocking Crossbar Switch Fabrics

There are many different kinds of switch fabric architectures. One type is constructed using a high-performance, *nonblocking* design that eliminates IP traffic jams within the router. Nonblocking architectures, theoretically, could be constructed simply by enabling each line card to absorb into its transmit memory the volume of bandwidth that equals the cumulative amount of bandwidth of all the line cards in the system. In such a scenario, there would never be contention for the resources that allow packets in a receive memory to cross the switch fabric to a transmit memory.

However, avoiding all contention for bandwidth through the use of extremely high-speed memory systems is expensive and impractical to implement, particularly given that there are ways to achieve almost the same effect using other design components and strategies, including *virtual output queues (VOQs)* and *intelligent schedulers*.

Buffering, Queuing, and Scheduling

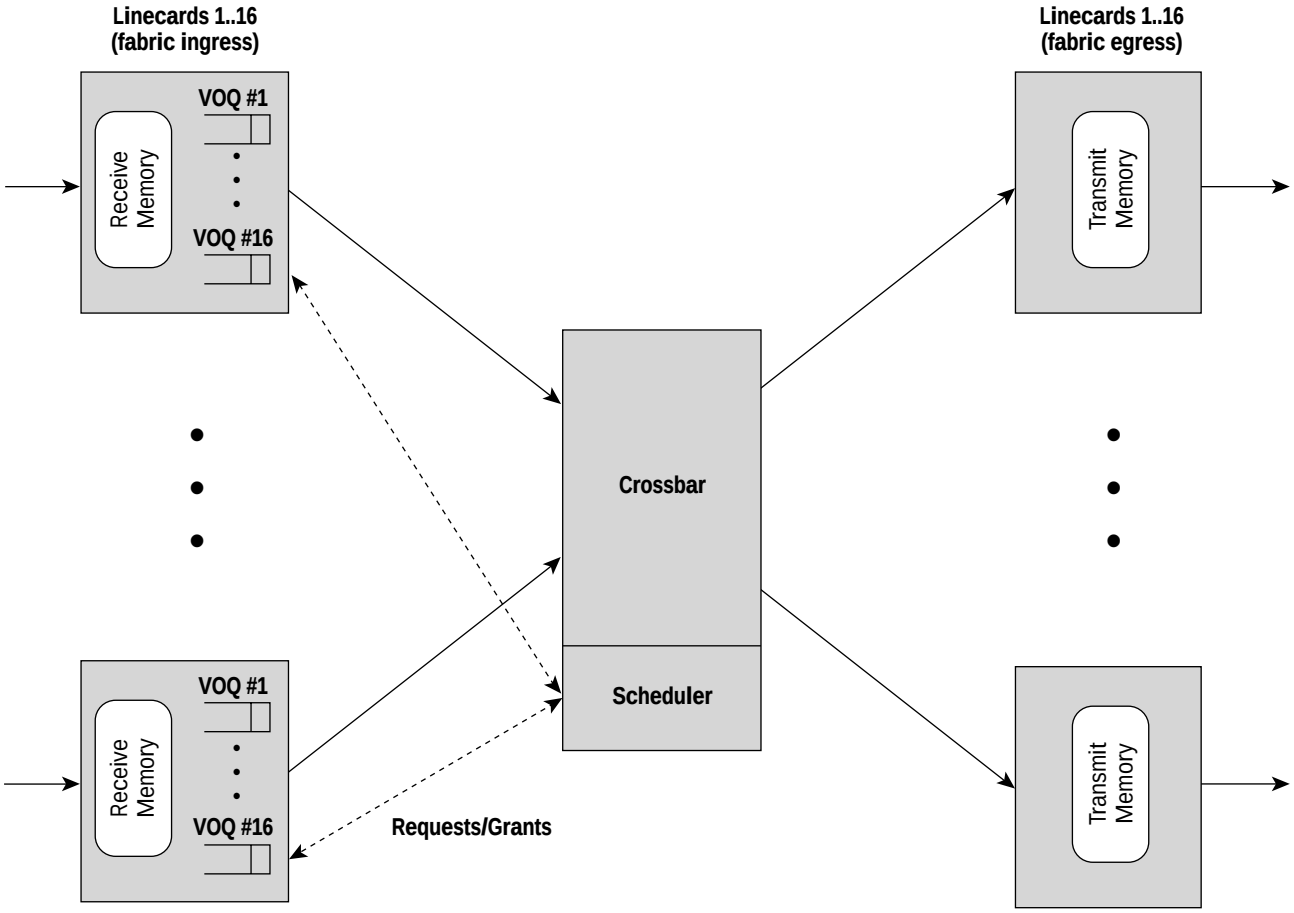
Implementing nonblocking systems in a more efficient manner relies on sophisticated buffering, queuing, and scheduling techniques. For example, if the rate at which traffic being presented to a line card exceeds the rate at which the transmit memory can accept it, there will be instances in which not all the data offered into the fabric can be immediately forwarded. In these cases, the data must be held in the receive buffer of the sending line card until there is finally an opportunity to forward it to the transmit memory of the appropriate line card. Without the proper queuing schemes, this situation will result in HOL blocking.

The key to avoiding this situation without having to build an extremely high-speed and expensive memory system is to divide the traffic in the receive memory of each line card into multiple VOQs and to implement a corresponding intelligent fabric scheduler. The way to configure such a system is to build a VOQ associated with each destination line card. As each packet is written into the receive memory, it is then automatically placed into the VOQ associated with its destination line card.

Meanwhile, a request is sent to a central scheduler to indicate that this line card now has data that needs to be sent to a particular destination line card. The scheduler can then make intelligent decisions as to which line cards should send packets to which other line cards and when, maximizing the total throughput achieved within the router.

VOQs are analogous to the dedicated turn lanes at street intersections. They prevent traffic waiting in line to turn in one direction from holding up traffic headed in different directions. The traffic light can be compared to the switch fabric scheduler in that it maximizes the total traffic “throughput” across the intersection based on knowledge about the traffic waiting in each lane of the intersection.

Figure 3 Nonblocking Architecture Components: Using a combination of a crossbar switch fabric, virtual output queues on each line card, and a central intelligent scheduler, router architectures can become nonblocking in nature without huge investments in very-high-speed memory systems.



Class-of-Service Support

Sophisticated queuing mechanisms also contribute to the ability of a crossbar switch architecture to enable the deployment of differentiated classes of service (CoS) for customer IP traffic. CoS support enables service providers to deliver and maintain contractually binding service-level agreements (SLAs) that result in additions to the service provider's portfolio of differentiated services. CoS capabilities also give customers greater flexibility in mixing and matching service and budget considerations.

Support for CoS at the system level can be delivered with a small number of hardwired queues that use a rudimentary, Weighted Round Robin (WRR) algorithm for transmitting packets. A superior approach for scalable high-end systems, however, is to have even more queues available used in combination with Modified Deficit Round Robin (MDRR). MDRR is an algorithm that combines a special low-latency queue for delay-sensitive traffic, predictive congestion-control mechanisms (such as Weighted Random Early Detection, or WRED), and other sophisticated capabilities. These technologies deliver the lowest latency and jitter levels possible, enabling a variety of IP services to run at optimum performance.

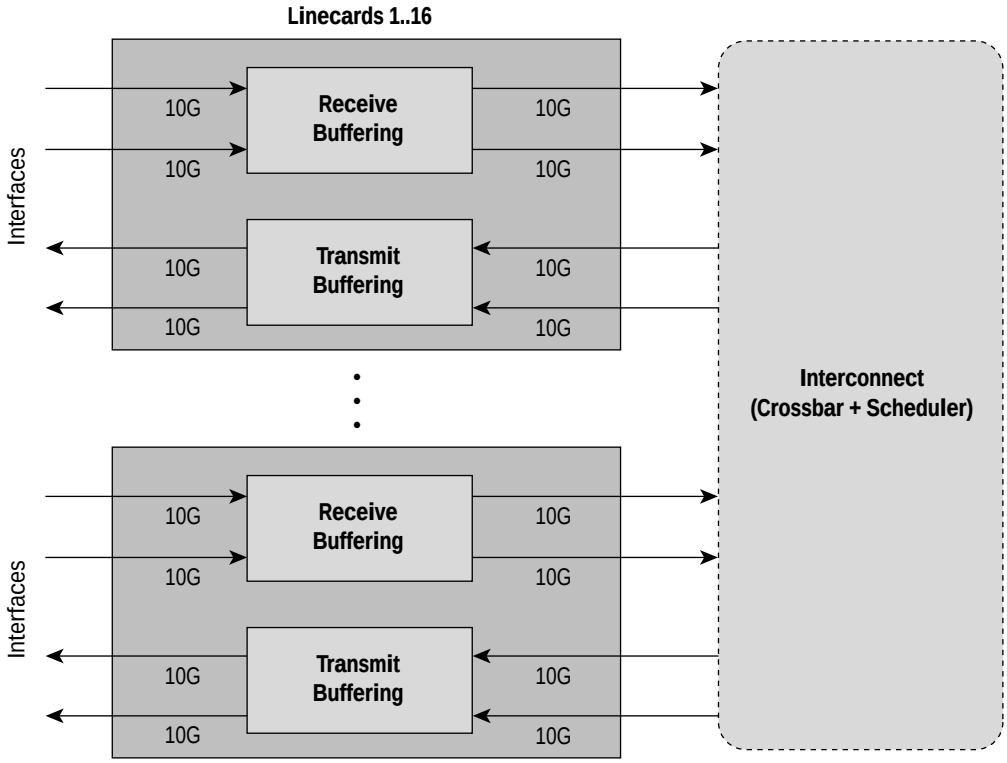
Increased Throughput

Distributed nonblocking switch fabric architectures, along with the vendor-specific application-specific integrated circuit (ASIC) design of the components of a system, play a pivotal role in enabling system throughput of hundreds of millions of packets per second. This is the level of performance required by today's Internet-class systems, which will continue to grow over time.

For example, Cisco 12000 series Internet routers now support line-rate 10 Gbps (OC-192c/STM-64c) fiber-optic interfaces, each with ASICs developed by Cisco. Each line card supports throughput of 25 million pps. With support for 15 line cards in a single chassis (in configurations not using a slot for a redundant route processor), total system performance is 375 million pps. (See Figure 4.)

The ability to deliver this ultrahigh throughput results from combining a crossbar switch fabric with a sophisticated VOQ and scheduling system to prevent HOL blocking.

Figure 4 Cisco 12000 series Architecture: The high-end Cisco 12000 series Internet router use a switch fabric architecture to distribute packet processing across multiple line cards, thus significantly enhancing system scalability and investment protection. A crossbar interconnect is used in combination with VOQs and an intelligent central scheduler to eliminate data blocking and provide high scalability and line-rate performance, even in a fully loaded system.



Other Checklist Items

In addition to scalability, system efficiency, and performance, a few other important considerations must be accounted for when evaluating system architectures. These include the following:

- Whether there is *backward compatibility* between new switching fabrics and existing line cards must be determined to preserve significant investments in router infrastructures.

- The ability to *"hot swap"* line cards; in other words, to replace or add line cards without interrupting system operations. This capability is obviously an important contributor to the overall uptime and integrity of any system.
- The *form factor and number of slots* of a particular chassis must be determined. For example, having more slots in a single chassis contributes to overall network scalability. This is because as a growing number of system slots are used to house line cards that interconnect routers within a single point of presence (POP), fewer slots are left over to house interfaces that interconnect POPs. With more slots available, there is more bandwidth for inter-POP transmission loads. In addition, larger routers enable simplified networks with fewer routing domains to administer.
- The *variety of configurations* available for achieving a certain throughput is an important factor that must be considered. For example, some service providers prefer to divide 10 Gbps bandwidth across four OC-48c/STM-16c (2.5 Gbps) ports on a single line card rather than running an OC-192c/STM-64c interface with the full 10 Gbps. If a service provider, for example, is selling OC-48c/STM-16c leased-line services to a large Web hosting or e-commerce customer, the OC-48c/STM-16c interface might be needed to terminate that service in the POP.

Conclusion

High-end router architectures best suited for carrying Internet-scale traffic loads are distributed in nature. In other words, the processing and buffering memory associated with the router packet-forwarding function is physically and logically distributed across each line card in a system. This setup prevents the service provider using the router from suffering performance degradation as increased traffic volumes begin to drain shared, finite CPU and buffering resources.

In addition, the use of crossbar switch fabrics with buffers on their line cards and sophisticated queuing and central scheduling capabilities further enhance the efficiency of large-scale distributed systems. Crossbar switch fabrics do so by enabling any line card in the system to send packets to any other line card with no data blocking.

Different router architecture characteristics certainly have different levels of importance, depending on the application of the router in the overall network system. The characteristics that are most commonly regarded as important for large Internet backbones are performance, reliability, and scalability. The use of distributed forwarding and buffering with a scheduler-based crossbar switch fabric, such as that of the Cisco 12000 series Internet router, is the architecture that best satisfies all these collective requirements.

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems Europe
11, Rue Camille Desmoulins
92782 Issy Les Moulineaux
Cedex 9
France
www.cisco.com
Tel: 33 1 58 04 60 00
Fax: 33 1 58 04 61 00

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems Australia, Pty., Ltd
Level 17, 99 Walker Street
North Sydney
NSW 2059 Australia
www.cisco.com
Tel: +61 2 8448 7100
Fax: +61 2 9957 4350

Cisco Systems has more than 200 offices in the following countries. Addresses, phone numbers, and fax numbers are listed on the Cisco.com Web site at www.cisco.com/go/offices.

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE Finland • France • Germany • Greece • Hong Kong • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

1/01 LW