

Scalable Content Switching

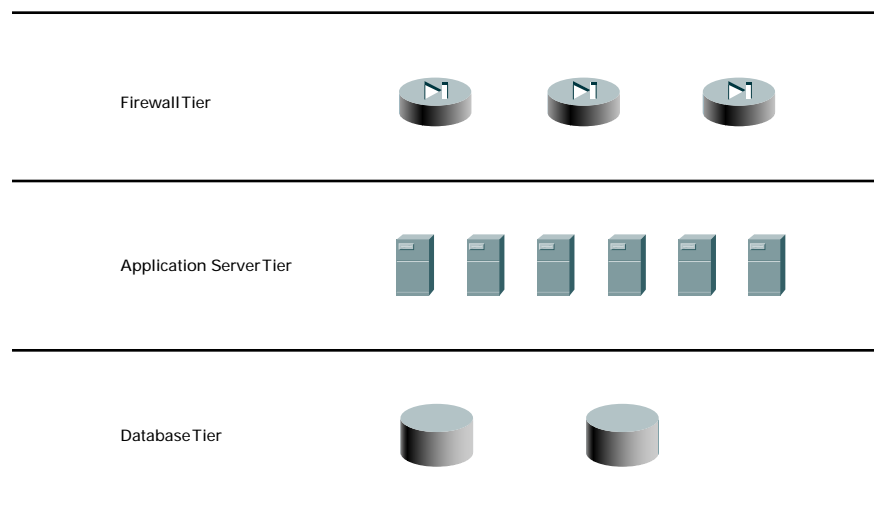
A Discussion of the Cisco CSS 11500 Series Content Services Switch Architecture

Introduction

As Web applications become increasingly more critical to business success, particularly in the areas of e-commerce, customer relationship management (CRM), and employee resource management (ERM), the performance, scalability, security, and availability of the application infrastructure become paramount concerns for IT and network managers. These requirements, in turn, drive the deployment of Web applications in tiers (layers).

Figure 1 shows a typical deployment. In this application, we see a security tier consisting of firewalls, an application server tier, and a database tier. Each tier may be scaled horizontally by adding additional devices appropriate for the tier. As load on the tier increases, more devices are added. In addition, having several devices at each tier provides functional redundancy. If a device in a given tier fails, another device can perform its functions.

Figure 1. Typical Web Application Infrastructure



For this approach to scaling and failover to work, there must also be a mechanism for intelligently routing messages between layers—content switches.

Content switches route messages to individual devices by inspecting the contents of the messages, and then forwarding them to specific devices based upon device or application requirements. Content switches monitor the health of each device and provide automatic



failover by routing messages to the remaining devices in the tier. Failover is also based on specific policies. Because of the critical role they play in enhancing Web application performance, availability, and security, content switches have become an indispensable part of e-business infrastructure.

Network designers building data centers to support these applications now need to scale content switching to accommodate the need for higher levels of performance. For designers who want to preserve a separation of content switching functions from Layer 2 and Layer 3 functions, modular content switches that can be incrementally scaled through the addition of additional processors, ports, or service modules are needed.

As network and application security concerns move to the fore and companies seek to protect both corporate and customer information, the percentage of Secure Sockets Layer (SSL) traffic is dramatically increasing. This trend has created a need to add scalable, cost-effective SSL termination capabilities to content switches, and the Cisco CSS 11500 Series Content Services Switch meets these requirements. The CSS 11500 switches are the industry's first compact, modular content switches that allow customers the flexibility to customize their installations for the precise number of ports, processing capabilities, or SSL termination capabilities they need.

This paper discusses in detail the unique architecture of the Cisco CSS 11500 Series. It is assumed that the reader has a basic understanding of IP, TCP, UDP, and HTTP (including SSL and the concepts of URLs and cookies). It is also assumed that the reader has a basic level of understanding of the *role* of content switches in a network but not necessarily *how* content switches operate to fulfill this role.

This paper is organized as follows:

- An overview of content switch operation
- Categories of processing in content switches
- The Cisco CSS 11500 modular multiprocessor architecture

Overview of Content Switch Operations

The device and application requirements that determine how messages are routed are expressed in terms of policies, and these policies are configured on the content switch. For example, when you route messages to a group of firewalls, you might want to balance the load across the available firewalls and at the same time ensure that all messages associated with a particular application connection traverse the same firewall in both directions. The reason is that firewalls need to be able to statefully inspect all the packets associated with a connection in both directions.

When you route messages to application servers, you might want to balance the load across the application servers, with the additional constraint that all connections from a particular client get routed to the same application server until a particular transaction or set of transactions is complete.



In general, the policies that can be implemented in content switches fall into one of the following five general categories:

- Load-balancing policies—These policies describe how connections and requests are to be distributed across an array of servers that are eligible to receive them. These policies are expressed in terms of the criteria that identify the requests to be distributed, the eligible devices capable of handling those requests, and the algorithms to be used to distribute them. Examples of load-distribution algorithms are round robin, weighted round robin, least connections, weighted least connections, least loaded, and predictive hash. The following example shows a load-balancing policy.

Policy 1

For each inbound request where:

Destination IP address = 192.32.12.3

Protocol = TCP

Destination port = 80 (HTTP)

Using the round robin load-distribution algorithm, balance the load across the following eligible servers:

IP address 10.10.10.1, port 80

IP address 10.10.10.2, port 80

IP address 10.10.10.3, port 80

- Persistence policies—These policies describe requirements to keep a series of connections or requests going to the same server in the array until a particular transaction or unit of work is complete. With persistence policies there must be a token passed in each request that can be used by the content switch as the means for maintaining persistence. Most often these tokens are IDs that represent a specific server or user session. They are typically found encoded in a URL or cookie, or in the case of secure traffic, in the unique SSL session ID. Either the server or the content switch may set these tokens. The following example shows a persistence policy.

Policy 2

For each inbound request where:

Destination IP address = 192.32.12.3

Protocol = TCP

Destination port = 80 (HTTP)

Using the round robin load-distribution algorithm, balance the load across the following eligible servers:

IP address 10.10.10.1, port 80

IP address 10.10.10.2, port 80

IP address 10.10.10.3, port 80

Maintain persistence to a selected server by setting the following token:

Cookie = CSS generated cookie

Note that a persistence policy works in conjunction with a load balancing policy. The load balancing policy is used to select the server that will be the initial point of contact. The persistence policy alerts the content switch to set a cookie to a unique value for this point-of-contact server and to direct subsequent requests received with this cookie to the same server.



- Server-failure policies—These policies are designed to give the operator control over switch behavior in the event of server failure. Different applications can require different treatments, and this is particularly true when using persistence policies. For example, what should be done when a server to which a client has a persistent connection mapped fails in the middle of a transaction? Possible options include reset the connection, issue an HTTP redirect (perhaps to a server that displays an error message), rebalance the connection to a new server using the load-balancing policy, or direct it to a special “sorry server” that becomes active if there are no other eligible servers for this policy. The following example shows a server-failure policy.

Policy 3

For each inbound request where:

Destination IP address = 192.32.12.3

Protocol = TCP

Destination port = 80 (HTTP)

Using the round robin load-distribution algorithm, balance the load across the following eligible servers:

IP address 10.10.10.1, port 80

IP address 10.10.10.2, port 80

IP address 10.10.10.3, port 80

Maintain persistence to a selected server by using the following token:

Cookie = “Session-ID”

When a persistent server fails, issue an HTTP redirect to 192.32.15.1/failuremessage.htm

- Content-specific policies—These policies are used to specify different treatment for different types of content. For example, you might want to direct all requests for cacheable content to a set of reverse proxy caches that offload the processing of static images from application servers. Or you might want to partition a Web server farm into static and dynamic sections. The following example shows a content-specific policy.

Policy 4

For each inbound request where:

Destination IP address = 192.32.12.3

Protocol = TCP

Destination port = 80 (HTTP)

URL = “*.GIF, *.JPEG” /* file extensions for cacheable image types

Using the URLHash load-distribution algorithm, balance the load across the following eligible caches:

IP address 10.10.20.1, port 80

IP address 10.10.20.2, port 80

IP address 10.10.20.3, port 80

Note: The URLHash distribution mechanism results in an optimal cache replacement policy by keeping requests for the same image files in the same cache.



- Device-specific policies—These policies are used to specify different treatment for different types of devices. Perhaps you want to direct clients using a specific wireless device to a set of servers that customize content for that device's specific formatting requirements. The following example of a device-specific policy accomplishes this goal.

Policy 5

Destination IP address = 192.32.12.3

Protocol = TCP

Destination port = 80 (HTTP)

User agent contains the string "PalmScape/PR5" /* Browser = Palm V

Using the round robin load-distribution algorithm, balance the load across the following eligible wireless gateways:

IP address 10.10.30.1, port 80

IP address 10.10.30.2, port 80

IP address 10.10.30.3, port 80

Content Rules, Services, and NAT

Some characteristics of these policies are particularly noteworthy. First, in the previous examples note that a policy matches a particular destination address, protocol, and port. This information, which represents the servers behind the content switch to the outside, is known as the content rule definition. The content services switch allows a content rule to match all IP addresses or an arbitrary range of IP addresses, an IP number, TCP or UDP port number, as well as URLs, cookies, domain names, and other HTTP headers.

The servers to which connections are forwarded are known as Services, and are defined by their IP addresses and ports. Content switches essentially accept connections that match virtual servers and direct them to real servers. In the process of doing this, they may modify the packets they forward.

Content switches use the Network Address Translation (NAT) mode (also called directed mode) of forwarding and the dispatch mode of forwarding. In NAT mode the destination address is "NATed" from the received destination (the one that matched the virtual server) to the address of the target real server. Port numbers may be changed as well. In addition, dependent header information (such as checksums) will be modified. The Media Access Control (MAC) address is also rewritten. In dispatch mode, only the MAC address is rewritten—the original destination IP address is preserved. This latter mode is important for load balancing devices such as transparent caches that may need the original destination address to fetch content upon cache miss.

Content switches are commonly required to modify additional fields in packets they forward. In addition to the destination address and MAC address information, common examples include source IP addresses, source and destination ports, sequence numbers, and IP, TCP, and UDP checksums.



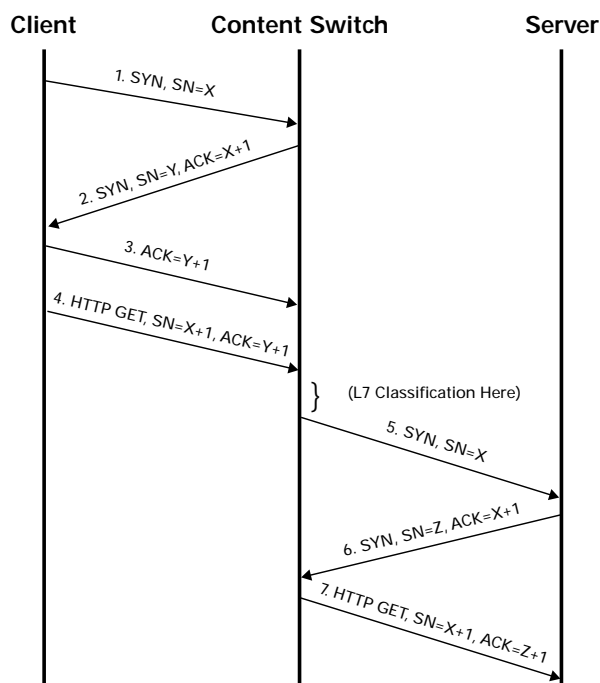
Policies, Protocol Layers, and Delayed Binding

New connections arriving at a content switch are first classified to determine whether matching policies exist, and then processed according to the actions specified by those policies. In the previous examples, you see that Policy 1 is invoked if the destination IP address = 192.32.12.3, the protocol = TCP, and the destination port = 80 (HTTP). The classification of this connection, therefore, can take place at Layer 3 and Layer 4 (IP address, protocol, and port). This classification can take place completely on the initial packet of the TCP connection (the SYN packet).

Policy examples 2 through 5 require additional classification based on data contained in the HTTP header. In examples 2 and 3, the required information is in the cookie header, in example 4 it's in the URL, and in example 5 it's in the user-agent header. Because HTTP is considered an application layer protocol, classification at this layer is referred to as Layer 7 classification. For these connections or requests to be properly classified, a process commonly known as "delayed binding" is used by the content switch to obtain the Layer 7 information.

In delayed binding, the content switch cannot apply Layer 7 policy to the connection until it has received the initial HTTP request and its associated headers (URL, cookies, user-agent, etc.), but the client will not send the HTTP information until it believes it has a connection established with the server. The content switch acts as a temporary proxy for the connection until it receives sufficient information to apply its policy, then binds the connection to the correct server as the policy dictates. Figure 2 illustrates the process of delayed binding used for Layer 7 classification.

Figure 2. Delayed Binding and Layer 7 Classification



In Steps 1 through 3, the client is establishing a TCP connection with the content switch as a proxy for the server. In Step 4, the client sends its HTTP request. The content switch now has sufficient data to apply its Layer 7 policy (in this case, resulting in the selection of a specific server). In Steps 5 through 7, the content switch establishes a connection with the selected server and forwards the HTTP GET to that server.



Sequence Number Remapping

The process of delayed binding introduces additional complexity to the forwarding process—sequence number remapping. For TCP connections, it is the responsibility of each of the parties participating to establish their own initial sequence number. The sequence number is then incremented by one for each byte transmitted over the connection. The receiver is responsible for sending back acknowledgements for each byte received. Because the content switch is acting as a proxy for the initial connection, it must supply an initial sequence number for its side of the connection. The initial sequence number established by the real server will be different. This implies that the content switch must translate these sequence numbers in both directions for all packets transferred.

In Figure 2 notice the sequence numbers (SNs) and acknowledgments (ACKs) for all participants (client, content switch, and server). In particular, note that the initial sequence number selected by the content switch (X) is different from the initial sequence number set by the server (Y). The content switch must compute the difference between these two initial sequence numbers and maintain that relationship by remapping the sequence numbers for each packet transferred in both directions. Note that during the delayed binding phase the number of packets exchanged between client and content switch (4) is greater than the number of packets exchanged between content switch and server (3). The reason is that the TCP specification allows the acknowledgement to be piggybacked with data, which the content switch takes advantage of. Second, in this example, the client sequence number is preserved without modification from client all the way to server. The reason for this is that it is the server side of the connection is being proxied by the content switch during the delayed binding process.

HTTP 1.1 and Connection Remapping

In earlier versions of HTTP there was a one-to-one relationship between HTTP requests and connections, but with HTTP 1.1 it is possible to transfer a series of HTTP requests and responses over a single TCP connection.

For certain types of applications such as caching applications, you may want to send each of the requests to a different server based on the load-balancing policy. In these applications it will be necessary to maintain the client side of the connection while remapping the server side of the connection to a new server. The content switch is responsible for correctly terminating the old server-side connection, re-computing all the appropriate NAT, port, and sequence number translations that will be applied to the new connection, establishing a connection with the new server, sending the new HTTP request, and then translating all the appropriate fields of subsequent packets associated with the connection in both directions.

Effectively, the Layer 7 classification and the server side of the delayed binding process are repeated for each new HTTP request (Steps 5 through 7 in Figure 2). The content services switch allows the operator to specify whether connection remapping should be turned on or off for a given virtual server. In the event connection remapping is disabled, Layer 7 policy is applied only to the first GET in the sequence.

Health Checking

Is it important for content switches to be able to detect the failure of servers or processes and to route around them. In complex Web applications, you might want to verify several layers of infrastructure that a target server depends on before you send requests to that server. To determine whether various components of application infrastructure are available, content switches need to provide flexible capabilities to test various types of servers.

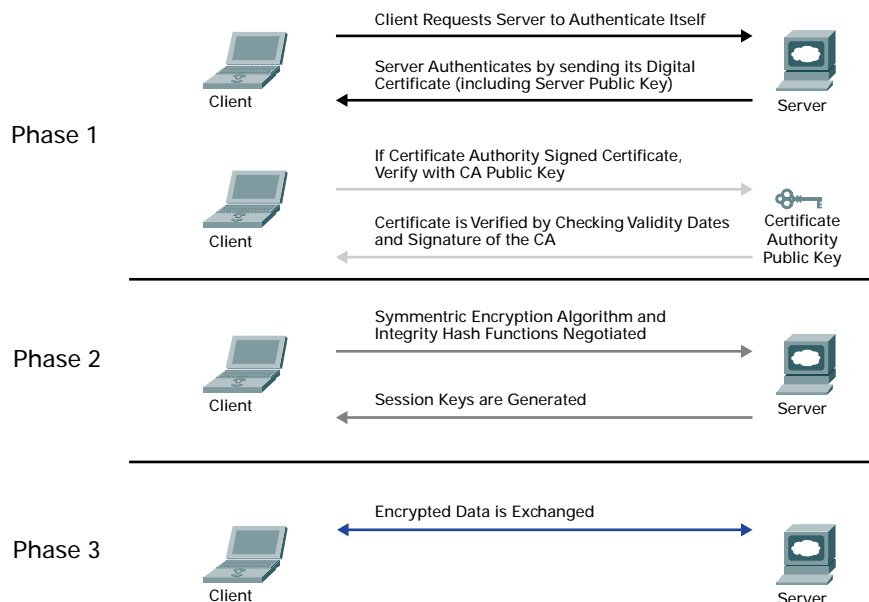


The content services switch offers a rich set of health checking capabilities including Layer 3, Layer 4, and Layer 7 health checks. Scripting functions also allow the operator to customize health checking. Health checks are sent at regular, configurable intervals. Failure of the health check results in labeling a particular server or set of servers unavailable. As with all active techniques of this kind, there is a trade-off between the frequencies at which health checks are sent and how fast a failure can be detected and the processing requirements associated with them.

SSL Termination

SSL (Secure Sockets Layer) is increasingly being used to improve the privacy and security of Web applications. SSL provides the mechanisms to verify identity (through public key cryptography) and to greatly enhance privacy (through the use of symmetric key cryptography). Figure 3 illustrates a typical SSL session.

Figure 3. SSL Session Establishment



Generally, SSL connections occur in the following three phases:

Phase 1, authentication—Through the use of certificate authorities and public key cryptography, clients and servers verify one another's identity and exchange public keys.

Phase 2, session setup—Using public key cryptography and the server's public key, session options are negotiated and the key is exchanged for a symmetrically encrypted data exchange.

Phase 3, session data exchange—Data between client and server is exchanged using the negotiated symmetric key and algorithm.

Software based implementations of SSL are extremely CPU intensive because of the complex mathematical operations that must be performed for both public key and symmetric cryptography. By offloading these functions to the content switch, CPU cycles can be regained for application processing. Even more important from a content switch perspective, however, is that no Layer 5 functions such as those required for the persistence policies, content-based policies, and device-based policies can be applied for encrypted traffic.



By terminating SSL traffic on the content switch, the HTTP header information becomes visible to the higher layer functions of the content switch enabling the same rich set of features to be applied to both encrypted and unencrypted data. Finally, by terminating SSL on the content switch, data center operators can reduce the administrative burden associated with managing certificates on multiple back-end servers.

Categories of Processing in Content Switches

In general, the processing that takes place as content switches perform various operations falls into one of the following three categories:

- Connection processing
- Forwarding processing
- Control processing

Following is a brief description for each, as well as an introduction to the types of performance metrics that will be useful in comparing different content switch architectures.

Connection Processing

Connection processing may be viewed as any processing associated with the establishment of new connections, the modification of existing connections, or the termination or removal of completed connections. Following is a list of tasks included in connection processing:

- Detection of new connections—Is this packet a TCP SYN or a UDP frame with a 5-tuple for which there is no forwarding state? If so, we may want to set up a connection.
- Virtual server lookup—Does this frame match a configured policy in the content switch? If so, can the switch apply its policy yet or does the switch need to do a delayed binding to get Layer 5 information?
- Delayed binding—Perform the three-way handshake and look at the received Layer 7 data. Does this match a configured policy? If so, apply the policy action.
- TCP state machine and denial-of-service detection—The content switch must ensure that the packets it is processing are part of a valid TCP connection. This is required for error detection purposes, out-of-order packet handling, as well as to detect any well-known denial-of-service attempts which exploit TCP vulnerabilities.
- Access control list (ACL) processing—Are there special filters that should be applied to this packet; for example, should this client be prevented from communicating to this server farm. If so, drop or continue processing the packet as required.
- SSL processing—Is this SSL traffic to be terminated? If so, perform all necessary cryptographic functions associated with establishing the SSL session and set up the symmetric exchange.
- Policy matching—Parse through the Layer 7 data for the required fields and determine whether those fields match the policy database.
- Frame buffering—Sometimes the Layer 7 data is not in the first application frame received. This requires the content switch to buffer frames until a match/no match decision can be made.
- Server selection—In most applications, the configured policy action will require the selection of a server based on load balancing, persistence, failover requirements, etc. The content switch must determine the server based on these requirements.



- Determine packet transforms—Should this connection be NATed? Do sequence numbers need to be translated? The content switch must determine which transforms need to be applied and what they are. The connection processing function must inform the forwarding process function of the connection parameters and transforms.
- Handle new HTTP requests on HTTP 1.1 connections—For active HTTP 1.1 connections the content switch must detect new HTTP requests and determine whether a new policy action needs to be taken; for example, selecting a new server. This also involves re-computing all packet transforms and ensuring the forwarding process is updated with the new information.
- Manage server-side connections—For connections where delayed binding has been applied, the content switch must perform the correct three-way handshake with the server, handling correctly any TCP errors. Likewise, for HTTP 1.1 connection remapping functions, the content switch must correctly terminate the old connections and establish the connection to the new server.
- Terminate connections—Connections may terminate normally (that is, a TCP FIN or RESET is received) or abnormally (a server or client fails or something fails in the network path so that no packets are received for a connection for a given timeout period). The content switch must clean up all states and reclaim all resources consumed by the connection.

Connection Processing Performance

The connections-per-second (CPS) metric is most commonly used with connection processing performance. For Layer 4 TCP and UDP traffic, CPS is the rate at which new

connections may be processed. For Layer 7 HTTP traffic, CPS is the rate at which new connections may be established or existing connections modified (as with HTTP 1.1 connection remapping).

Note that connection processing does not take place on every packet associated with every connection. It occurs on those packets associated with the beginning and ending of connections, and on those that signal the potential need for modification of connections in flight. Applications that use long-lived connections and few connection modifications will have lower connection processing requirements than those that are short lived or modified frequently.



Forwarding Processing

Processing associated with the forwarding of packets that are part of established connections may be viewed as forwarding processing. The following tasks are associated with forwarding processing:

- Classify packet—If the packet should be forwarded at Layer 2 without further higher layer processing, get the appropriate egress interface. If the packet is part of an existing connection, get packet transformation information and next hop information. If the packet is not part of an existing connection, pass the packet to the connection processing function.
- Apply packet transforms—Apply address changes, port changes, sequence number changes, and checksum changes.
- Encrypt and decrypt packet—If this packet is part of an SSL data stream, perform encryption or decryption as necessary.
- Queue and transmit packet—Send the packet to its destination.

Forwarding Processing Performance

The following performance metrics are important when discussing forwarding performance.

- NAT forwarding rate in packets per second (pps)—Smaller packet sizes require greater packet-per-second handling capacity for a given throughput. Unlike most LAN switches, content switches are typically performing a significant number of packet transformations on the packets they forward.
- NAT throughput (Mbps)—Total NAT throughput in megabits per second, regardless of packet size, must be understood.
- Simultaneous connections—Due to unique packet transformations, each connection managed and forwarded by a content switch requires its own forwarding table entry and connection state. Therefore, every content switch will have a limit (typically due to forwarding table memory size) on the number of simultaneous connections it can manage.

Note that forwarding processing occurs on every packet the content switch handles. Although this processing is less complex than the processing associated with connection management and control, it happens more frequently and should be optimized.

Control Processing

Control processing refers to all processing that is not associated with connection setup or forwarding but is essential to the proper operation, configuration, monitoring, or control of these processes. Control processing is generally asynchronous to the main tasks of the content switch (that is, handling connections and forwarding). Following is a list of tasks associated with control processing:

- System management processes—Tasks associated with the management of configuration files, system images, booting, diagnostics, hardware, etc.
- Operations management processes—Tasks associated with the management of the running system including configuration (command line, Web interface, application programming interfaces (APIs) etc.), statistics, Management Information Base (MIB) management, and Simple Network Management Protocol (SNMP).



- Routing and bridging management processes—Tasks associated with routing table management such as Open Shortest Path First (OSPF), Routing Information Protocol (RIP), and tasks associated with bridge table management such as Spanning Tree.
- Health checking—Scheduling and transmitting health check messages, examining results, moving servers in and out of service as required. Note that the processing associated with this can be extensive and includes the following:
 - Formatting queries
 - Establishing and maintaining connections
 - Transmission and receipt of messages
 - Parsing and interpretation of replies
 - Taking appropriate action (which may involve notification of several other parts of the system)

Control Processing Performance

The following are useful metrics to consider when discussing control-processing performance:

- Probes per second—The number of probes (active health check messages) a content switch can process per second. Note that processing requirements increase as active health checks become more complex (for example, scripting).
- Number of probes—There will typically be memory or configuration limits to the number of health checks that can be configured.

Note that probes have the lowest frequency of all the processing types discussed because their complexity may have the highest unit overhead.

The next sections discuss the details of the Cisco CSS 11500 Series modular content switching architecture. This architecture supports the specific types of processing tasks that content switching requires.

Cisco CSS 11500 Series Modular Content Switching Architecture

The Cisco CSS 11500 Series Content Services Switch is designed to provide modular performance as well as flexibility to accommodate most content switching requirements. In particular, it is flexible in terms of the types of topologies it supports as well as the types of services it provides to the server farm.

Flexible Topology Support

The Cisco CSS 11500 Series easily accommodates any of these three commonly used content switching topologies:

One-armed—The content switch is connected off to the side of the Layer 2/Layer 3 infrastructure. It is not directly in the path of traffic flow and only receives traffic that is specifically intended for it. Traffic which should be directed to it is controlled by careful design of virtual LANs (VLANs), virtual server addresses, server default gateway selection, or policy routes on the Layer 2/Layer 3 switch. The servers are connected to the Layer 2/Layer 3 infrastructure not to the content switch. Server-to-server traffic flows through the Layer 2 or Layer 3 switch without content switch intervention.

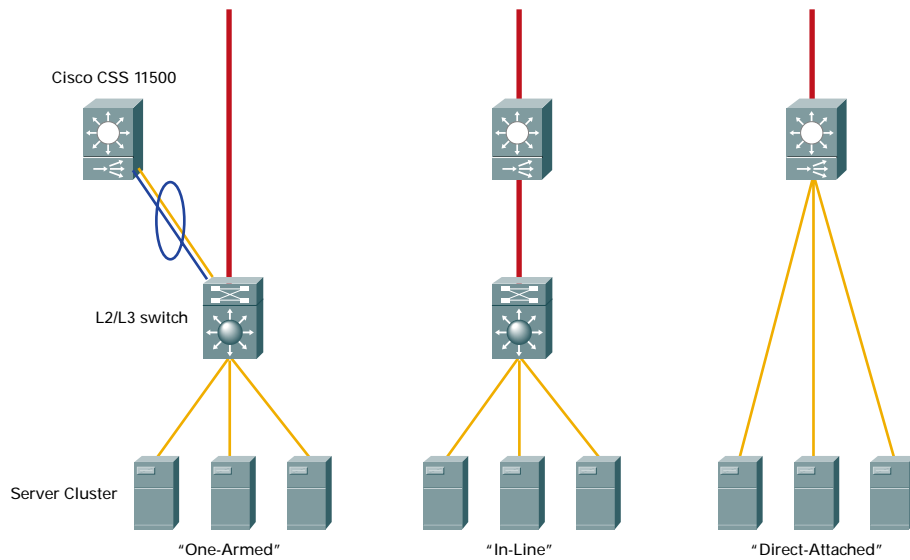
In-line—The content switch is connected in the path of all traffic to and from the server farm but it is not providing connectivity for the servers. The servers instead are connected to a Layer 2 or Layer 3 switch. Server-to-server traffic flows through the Layer 2 or Layer 3 switch without content switch intervention.



Direct-attached—The servers are directly connected to the content switch. All traffic, including server-to-server traffic flows through the content switch.

Figure 4 illustrates each of these topologies.

Figure 4. Content Switching Topologies



Following are a few points to note from an architectural perspective. First, the one-armed and in-line topologies require relatively few ports, because they must support only trunks. In contrast, the direct-attached topology requires content services switch ports for trunks and server connections. Second, forwarding and connection processing requirements are dictated by the size and performance requirements of the server farm, not necessarily the number of ports on the content services switch. In fact, the high-density fan-out capabilities of the Cisco Catalyst® switches might result in much larger server farms being supported by a content services switch than if they are direct attached. Therefore, it is desirable in these configurations to be able to add forwarding and connection processing power without adding ports.

Flexible Services Support

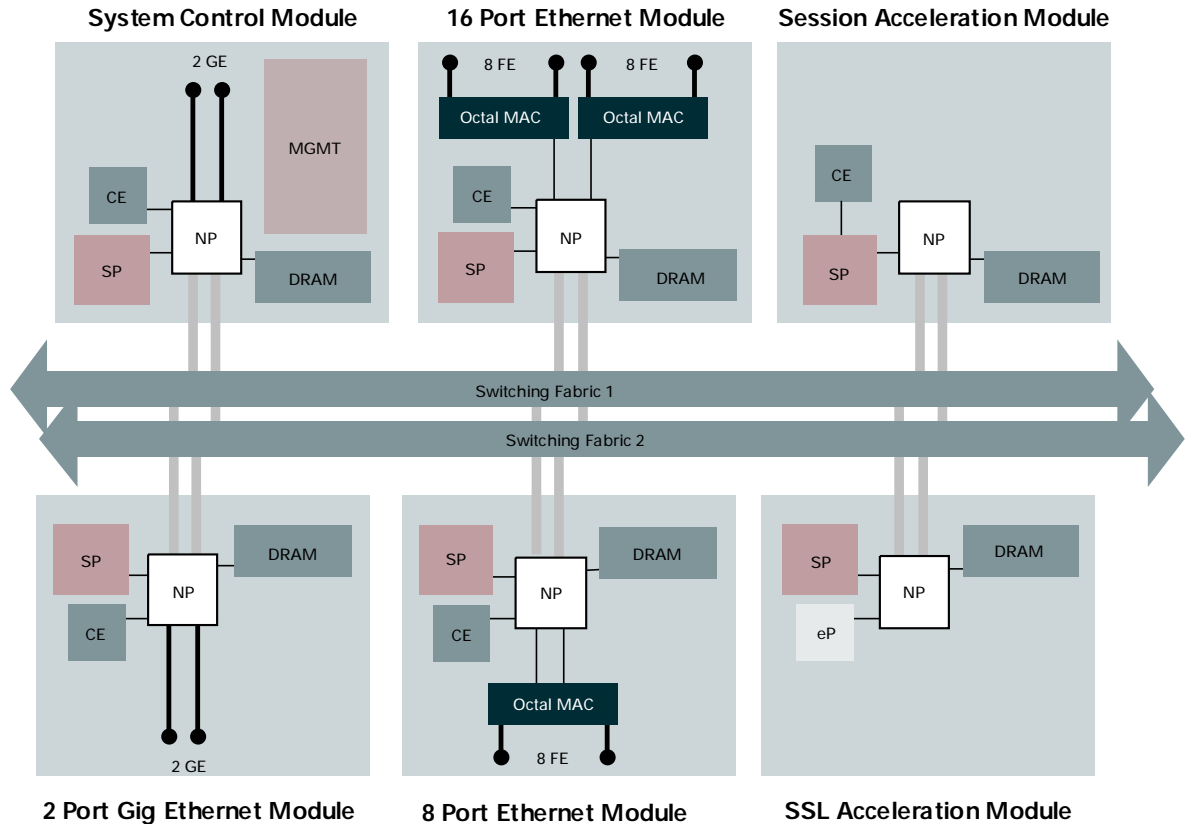
In addition, it is highly desirable to be able to add additional services to the network without adding additional devices. The content services switch has been built to support the addition of service modules to provide these services.

Cisco CSS 11500 Architectural Overview

The Cisco CSS 11500 Series uses a common architecture and software base. The CSS 11500 Series consists of the CSS 11501 Content Services Switch, a single-board, low-cost, fixed configuration content switch; the CSS 11503 Content Services Switch, a three-slot, two-rack-unit, chassis-based content switch; and the CSS 11506 Content Services Switch, a six-slot, five-rack-unit, chassis-based content switch. The architecture consists of one or more content switching modules connected by a high-speed switching fabric. These modules may provide connection and forwarding processing power, ports, or special services (that is, SSL termination) depending on the module. This is illustrated in Figure 5.



Figure 5. CSS 11506 Architecture



There are four different types of modules that can be used in the Cisco CSS 11500 modular chassis.

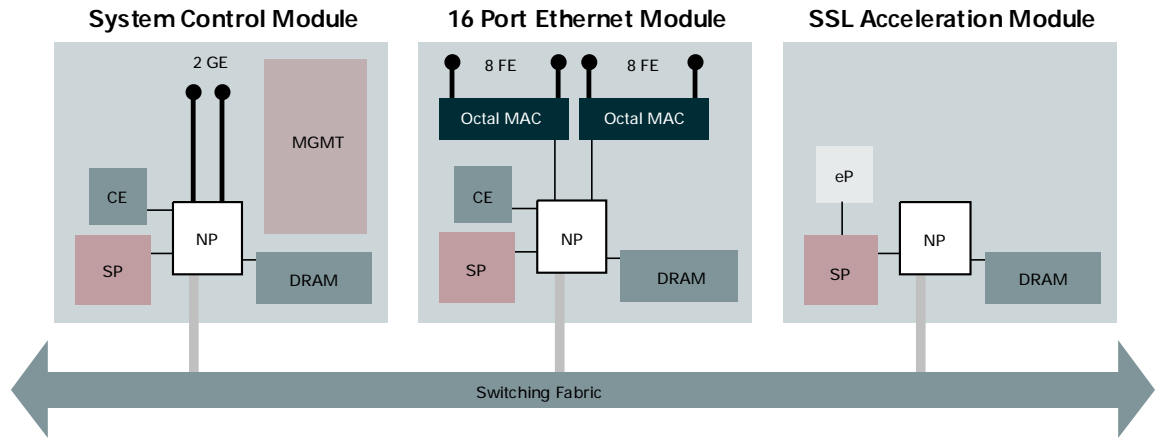
1. System control module—This module provides two Gigabit Ethernet ports as well as logic for management functions (PCMCIA Flash or hard disk, console interface, and management Ethernet port). In addition it has network processor (NP) and session processor (SP) resources for forwarding and connection processing respectively. The connection processor on this module also supports control processing.
2. I/O modules—There are three types of I/O modules available, including a two-port Gigabit Ethernet module, an eight-port 10/100 Ethernet module, and a 16-port 10/100 Ethernet module. Each module has a network processor for forwarding and a session processor for connection processing.
3. Session acceleration module—This module adds a network processor and session processor for additional forwarding and connection processing power. Multiple Session Acceleration Modules may be added to improve performance in one-armed and inline configurations.
4. SSL processing module—This module add cryptographic support for SSL termination and acceleration through an e-commerce processor. Several SSL processing modules may be added to the system for additional SSL performance and scalability.

This set of modules gives the application and network designer the flexibility to add the precise number of ports, the forwarding and connection processing power, and the SSL processing capabilities required for the application.



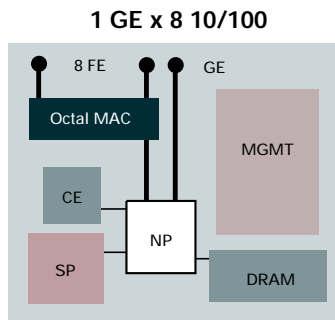
The Cisco CSS 11503 is a three-slot version of this modular architecture. Figure 6 shows the CSS 11503. Note that unlike the CSS 11506, the CSS 11503 does not support switching fabric, power supply, or system control module redundancy. It does, however, support several device-to-device redundancy strategies.

Figure 6. CSS 11503 Architecture



The Cisco CSS 11501 is not a modular chassis, but it does share the same architecture as other content services switches, as shown in Figure 7.

Figure 7. CSS 11501 Architecture



Major Architectural Elements

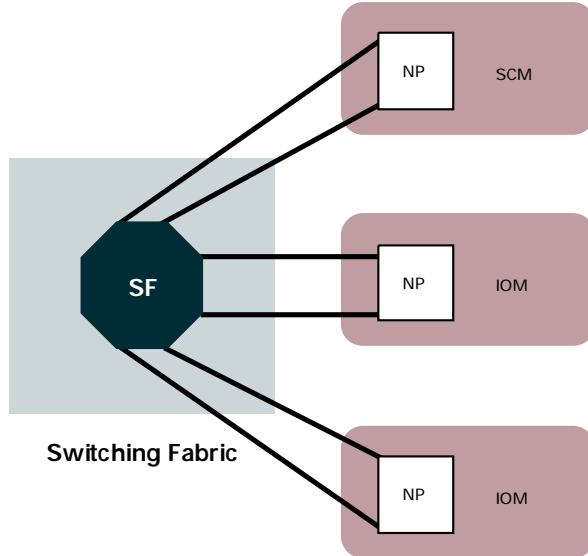
The following sections briefly discuss each of the major elements of the architecture.

Switching Fabric

The Cisco CSS 11503 makes use of a nonblocking switching fabric, which delivers 20-Gbps throughput to the system. Each module in the system has two 1.6-Gbps connections to the fabric for a total of 3.2 Gbps of full duplex bandwidth to each slot. The fabric supports four levels of priority. The CSS 11503 fabric connectivity is illustrated in Figure 8.

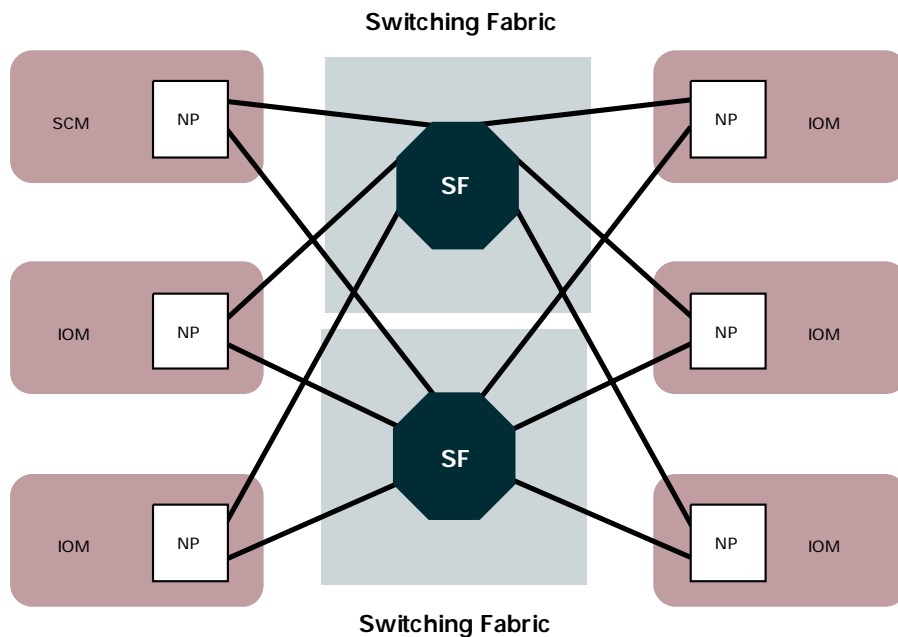


Figure 8. CSS 11503 Switching Fabric



The Cisco CSS 11506 supports two switch fabric cards for a total of 40 Gbps throughput. Each module in a CSS 11506 has one 1.6 Gbps link to each of the two fabric cards. Total bandwidth per slot is the same as the Cisco CSS 11503. The two fabrics operate in parallel to double the throughput of the system and to provide redundancy. In the event of a switch fabric failover, the modules will use the remaining module. Throughput in this mode is reduced by half. The CSS 11506 connectivity is illustrated in Figure 9.

Figure 9. CSS 11506 Redundant Switching Fabrics





Network Processors

Each network processor has four 200-MHz packet-processing engines, each of which has its own direct memory access and Hash table lookup co-processors. Each network processor has access to up to 256 MB of high-speed memory.

In Figure 9 you can see that one of the functions provided by the network processor is the connection of the module to the switching fabric. In addition, the network processor is responsible for all forwarding processing in the system. One of the innovations of the Cisco CSS 11500 Series is that all network processors on all modules, excluding the SSL module, are available for packet processing to all ports.

Classification Engine

Each I/O module and session module has a classification engine in addition to a network processor (see Figures 5 and 6). The network processor uses the classification engine as a co-processor to optimize processing of ACLs and Layer 2 MAC address lookups. This substantially offloads the network processor for other forwarding tasks.

Session Processor

Every network processor is paired with a session processor (see Figures 5 and 6). The session processor is a general purpose Reduced Instruction Set Computer (RISC) processor used for connection processing tasks. The session processor on the system control module is also responsible for control processing tasks. One of the innovations of the Cisco CSS 11500 Series architecture is that all session processors on all modules, excluding the SSL module, are available for connection processing to all ports.

E-Commerce Processor

The SSL acceleration module has its own e-commerce processor for offloading cryptographic processing from the session processor (see Figures 5 and 6). The e-commerce processor selected for use in the Cisco CSS 11500 Series has the following features:

- A high-performance public-key processor—A hardware engine used for processing the public keys used for SSL session establishment
- A bulk encryption processor—A hardware engine used for encryption of all data sent over an established SSL session

In conjunction with the session processor used on the SSL acceleration module, this e-commerce processor provides dramatic price-performance ratio improvements over software only solutions.

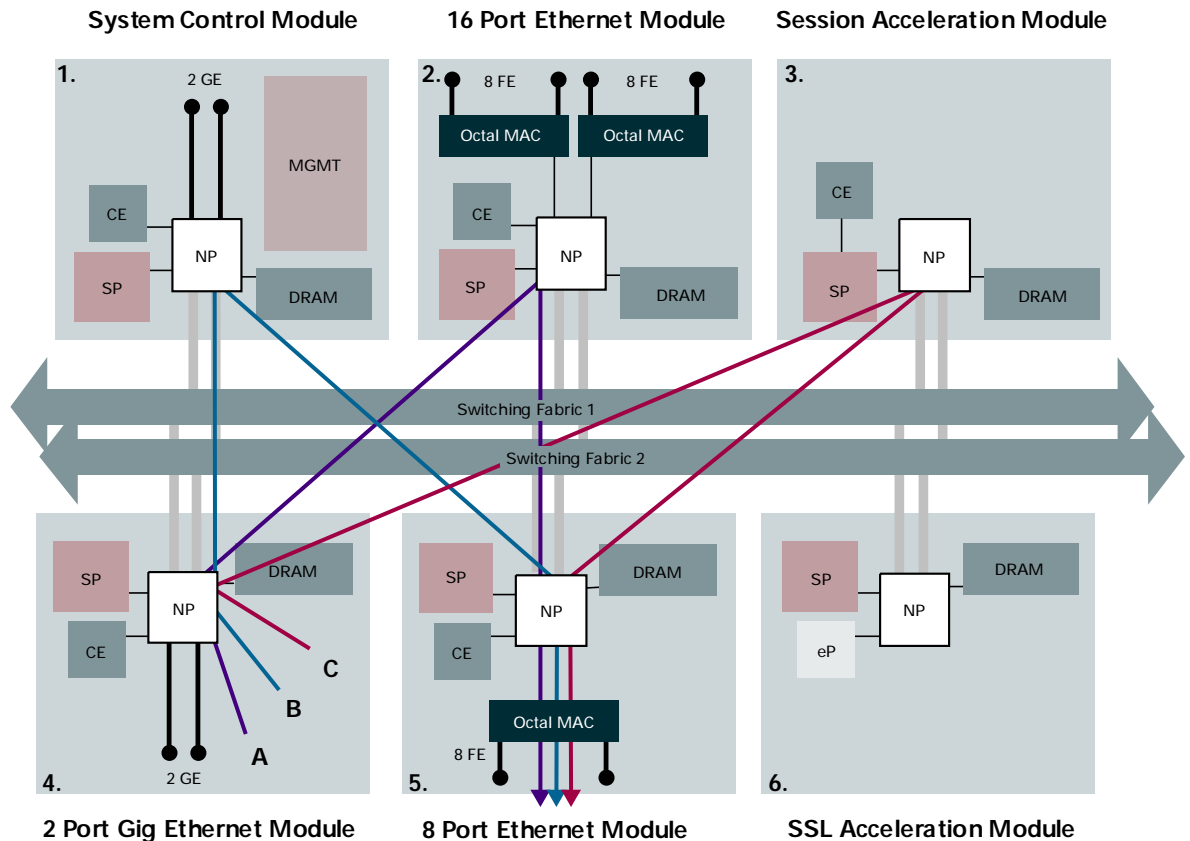
The following sections describe how the architecture operates.

Cisco CSS 11500 Series Operations

Distributed Processing

One of the important benefits of the Cisco CSS 11500 Series architecture is that all network processor and session processor resources in the system can be used for forwarding and connection processing tasks regardless of the ports where packets arrive. This is illustrated in Figure 10.

Figure 10. Cisco CSS 11500 Series Distributed Processing



Note that for each of the three connections shown (A,B,C), the ingress module (4) and egress module (5) are the same. The Cisco CSS 11500 uses a load distribution algorithm that takes advantage of different network processor and session processor resources for each of the connections. This allows all processing resources to be used in the system even when most of the traffic is arriving on a few interfaces. This is extremely important because in most content switching applications the highest volume of traffic is concentrated through a few uplink ports pointing toward the Internet or intranet. In addition, this architecture allows for forwarding and connection performance to be scaled as modules are added to the system.

Life of a Connection

The network processor and session processor associated with the ingress interface for a given connection are called the ingress network processor and ingress session processor, respectively. Similarly the network processor and session processor associated with the egress interface for a given connection are called the egress network processor and egress session processor. Finally, the network processor and session processor associated with forwarding and connection processing for a given connection (for example module 2 for connection “A” above) are referred to as the master network processor and master session processor for that connection.

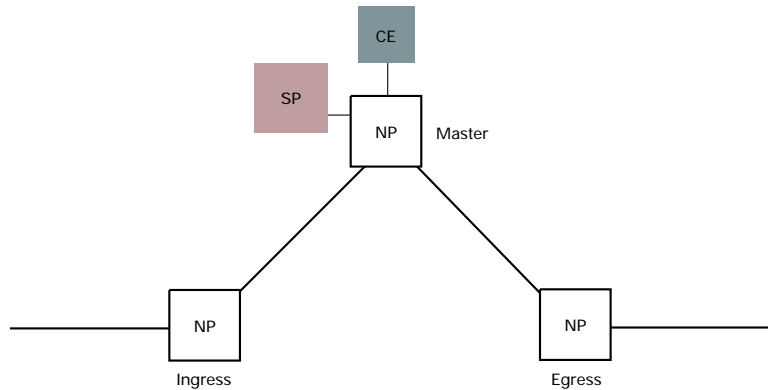
Note: It is possible that ingress, egress, and master network processors and session processors could be the same for a given connection.



Although all three network processors are involved in forwarding processing for a given connection, it is the master network processor that does the bulk of the work. In contrast, only the master session processor is involved in connection processing for a given connection.

The flow of packets through the system is illustrated in Figure 11.

Figure 11. Cisco CSS 11500 Series Packet Flow



For new connections or for modifications to existing connections the sequence of processing steps is as follows:

1. Packet arrives at the ingress network processor.
2. The ingress network processor selects the master network processor and forwards the packet to it.
3. The master network processor determines this packet needs connection processing and passes it to the master session processor.

Note: If this is a new connection the master network processor also uses the classification engine to do an ACL lookup and hands those results with the packet to the master session processor.

4. The master session processor compares the packet to its configured rules, determines which server should receive this packet, NATs the packet as required and forwards the packet to the egress network processor.
5. The master session processor gives the master network processor instructions on how to NAT the rest of the packets for this connection. Note that the return path for this connection is programmed as well.
6. The egress network processor transmits the packet to the server.



The following sequence of steps applies to packets that are part of an existing connection:

1. Packet arrives at the ingress network processor.
2. The ingress network processor selects the master network processor and forwards the packet to it.
3. The master network processor determines that this packet is part of an existing connection and retrieves the NAT information for it.
4. The master network processor NATs the packet and forwards it to the egress network processor.
5. The egress network processor transmits the packet to the server.

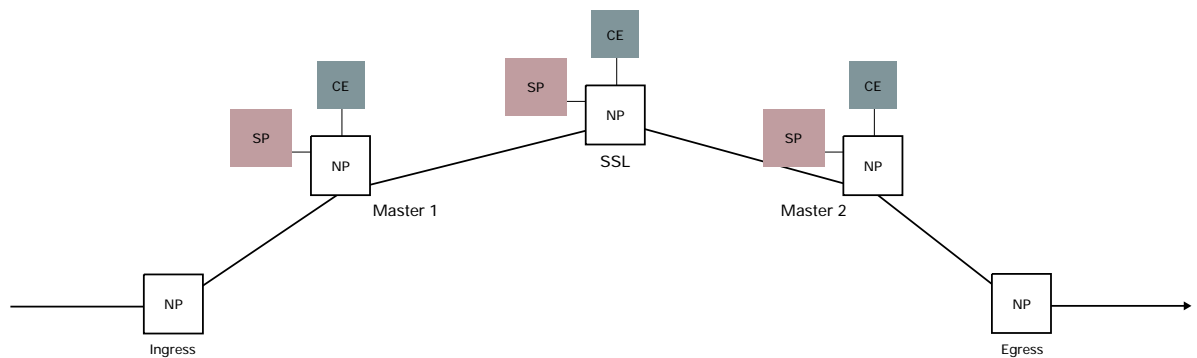
Note that for packets that are part of existing connections, the session processor is never invoked. This dramatically streamlines the processing path for the vast majority of packets associated with each connection.

Packets flowing in the reverse direction (for example, server to client) are handled the same way because this path was programmed at the time the initial connection was established.

SSL Connections

Figure 12 illustrates the flow of packets through the system for an SSL session. Note that for SSL sessions there are two master modules: one for the connection from the client to the SSL module, and one for the connection from the SSL module to the server. In practice, these master modules will often be the same, as either the ingress or egress module (or both). Multiple SSL modules may be used to increase the scalability and performance of SSL processing.

Figure 12. Cisco CSS 11500 Series SSL Packet Flow



The sequence of steps is the same as those described for both new connections and packets that are part of existing connections, with one exception. The process happens twice—once for each connection to the SSL module and once for each connection from the SSL module.

Because of the system's distributed processing and the large amount of unused bandwidth in the switching fabric, the additional processing causes no significant performance degradation. Furthermore, SSL processing typically occurs for only some connections. The unique architecture of the Cisco CSS 11500 Series allows non-SSL traffic to bypass the SSL modules.

Cisco CSS 11500 Series Architecture Summary

The Cisco CSS 11500 Series architecture uses several innovative techniques to take content switching to new price-performance ratios. The primary innovations are the mechanisms used to automatically distribute load across all the processing resources in the system, separate forwarding and connection processing on processors optimized for those tasks, and the use of special purpose hardware to optimize performance for expensive tasks such as ACL processing and encryption.

In summary, we can say that the Cisco CSS 11500 Series sets a new standard for content switching in the following areas:

- **Features**—The capability to program selected components yields tremendous flexibility in feature development. The ability of the architecture to accept service modules makes it an attractive platform for special services like SSL termination.
- **Flexibility**—The ease with which the Cisco CSS 11500 Series can be configured for one-armed, inline, and direct-attached configurations make it easy to insert into any data center architecture.
- **Price and performance**—The Cisco CSS 11500 Series' scalable, modular architecture helps ensure that it can match almost any price or performance requirement.

The Cisco CSS 11500 Series is one of several products in the Cisco content switching product line that addresses the growing customer demand for scalable Layer 4 to Layer 7 services as a complement to existing Layer 2 and Layer 3 infrastructure. Cisco content switches help enable businesses to build highly available and secure network data centers to support their Internet and intranet applications. The CSM provides an integrated services module for the Cisco Catalyst 6500 Series Switch and Cisco 7600 Internet Router, and delivers the highest Layer 4 to Layer 7 performance with extensive application features.

For more information about Cisco content switching, visit:

<http://www.cisco.com/go/contentswitch>



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-europe.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, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912
www.cisco.com
Tel: +65 317 7777
Fax: +65 317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the

Cisco Web site at www.cisco.com/go/offices

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia
Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • 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

Copyright © 2003 Cisco Systems, Inc. All rights reserved. CCIP, CCSP, the Cisco Powered Network mark, the Cisco Systems Verified logo, Cisco Unity, Follow Me Browsing, FormShare, iQ Breakthrough, iQ FastTrack, the iQ logo, iQ Net Readiness Scorecard, Networking Academy, ScriptShare, SMARTnet, TransPath, and Voice LAN are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, The Fastest Way to Increase Your Internet Quotient, and iQuick Study are service marks of Cisco Systems, Inc.; and Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, the Cisco IOS logo, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherSwitch, Fast Step, GigaStack, Internet Quotient, IOS, IP/TV, iQ Expertise, LightStream, MGX, MICA, the Networkers logo, Network Registrar, Packet, PIX, Post-Routing, Pre-Routing, RateMUX, Registrar, SlideCast, StrataView Plus, Stratum, SwitchProbe, TeleRouter, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0301R)

Printed in the USA