

## Connection Handling within the Cisco Application Control Engine Module Hardware

The Cisco® Application Control Engine (ACE) family of products is a data center solution for maximizing the availability, performance, and security of applications and Web services. This document provides a technical explanation of how connections are handled within network processors of the Cisco ACE Module. The details of how connections are load balanced, network addresses are translated, and Secure Sockets Layer (SSL) connections are terminated at the hardware architecture level are covered. This document is intended as a technical reference for networking professionals familiar with application delivery, network design, and facilitation of services, and those who are interested in better understanding how the Cisco ACE Module uses network processors to handle connection processing.

### Architecture

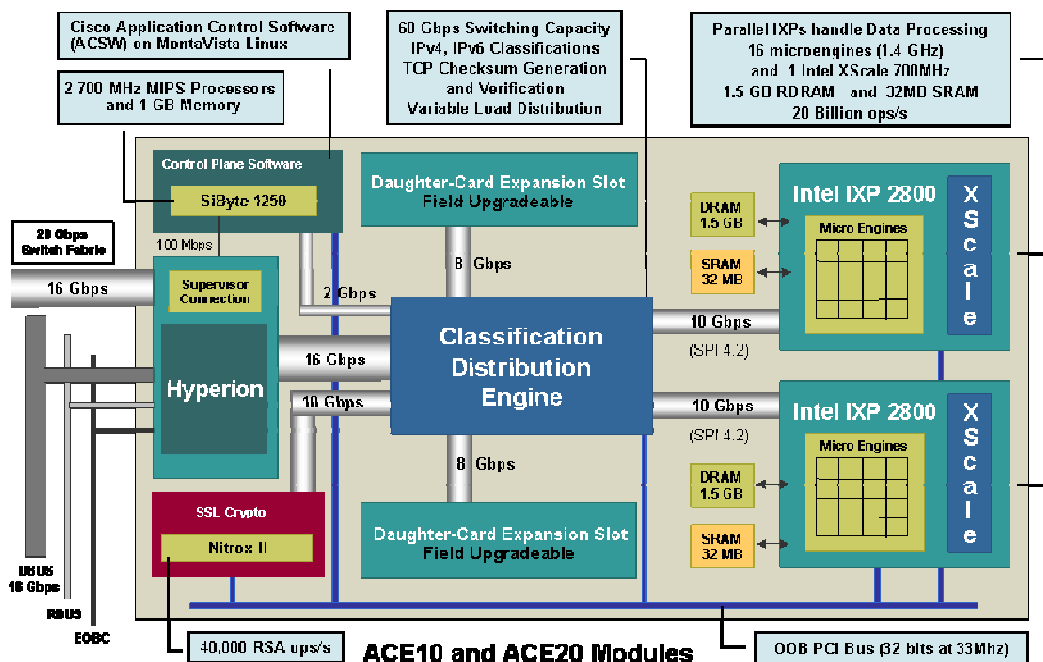
The Cisco ACE Module is a CEF720 line card connecting to the Cisco Catalyst® 6500 Series Switches. It receives all connections from clients and to the server over the Cisco Catalyst 6500 switching backplane using a Hyperion application-specific integrated circuit (ASIC), which is commonly found on WS-X6748 modules for the Cisco Catalyst 6500 Series. The Cisco ACE Module has interconnects to the Ethernet out-of-band channel (EOBC) (100 Mbps), shared bus (8 Gbps), and switch fabric (20 Gbps). The EOBC is used to allow the Cisco IOS® Software session command access and to allow the Cisco ACE Module to boot from the supervisor's storage if necessary. The shared bus interconnect is used only if communication to classic cards is required. The Cisco ACE Module uses a single interconnect to the switching fabric to provide a 20-Gbps connection to support up to 16-Gbps throughput.

After a connection is received by the Cisco ACE Module, there are five main components where a connection may be processed, depending on the nature of the connection:

- Classification and distribution engine (CDE): Application-specific field-programmable gate array (FPGA)
- Control plane: Dual-core SiByte MIPS processor clocked at 700 MHz, commonly referred to as the control plane
- Network processors: Two Intel Internet Exchange Processors (IXPs) 2800
- Crypto complex: Cavium NITROX II
- Two connectors: For optional daughter cards

Figure 1 shows the logical layout of the Cisco ACE Module and includes the five main components.

**Figure 1.** Hardware Architecture Overview of the Cisco ACE Module



As traffic enters the Cisco ACE Module, there is a clear separation between the in-band and out-of-band communications channels. In general, traffic destined for the Cisco ACE Module is categorized as either to the device or through the device.

- To the device, or traffic destined for an interface VLAN IP address configured on the module: This traffic matches a class map of type management, which is applied as a service policy to an interface VLAN. The management class map supports these protocols: HTTP, HTTPS, Internet Control Message Protocol (ICMP), Keepalive AP (KAL-AP) User Datagram Protocol (UDP), Simple Network Management Protocol (SNMP), Secure Shell (SSH) Protocol, and Telnet. This traffic is called control plane traffic.
- Through the device, or any traffic not matching a class map of type management and traversing the Cisco ACE Module: For load-balancing configurations, traffic must match a class-map matching on a virtual address. For access control lists (ACLs), inspections, Network Address Translation (NAT) or TCP/IP, normalization traffic can match a class map, or more commonly traverse an interface configured on the module. This type of traffic is commonly referred to as client-server traffic or load-balanced traffic and is known as data plane traffic.

### Control Plane

Traffic destined to the Cisco ACE Module and allowed by the management policy is sent directly to the control plane for processing, by the CDE. The control plane provides the user interface to the Cisco ACE Module. Features provided by the control plane include:

- Configuration manager
- Cisco ACE Extensible Markup Language (XML) interface

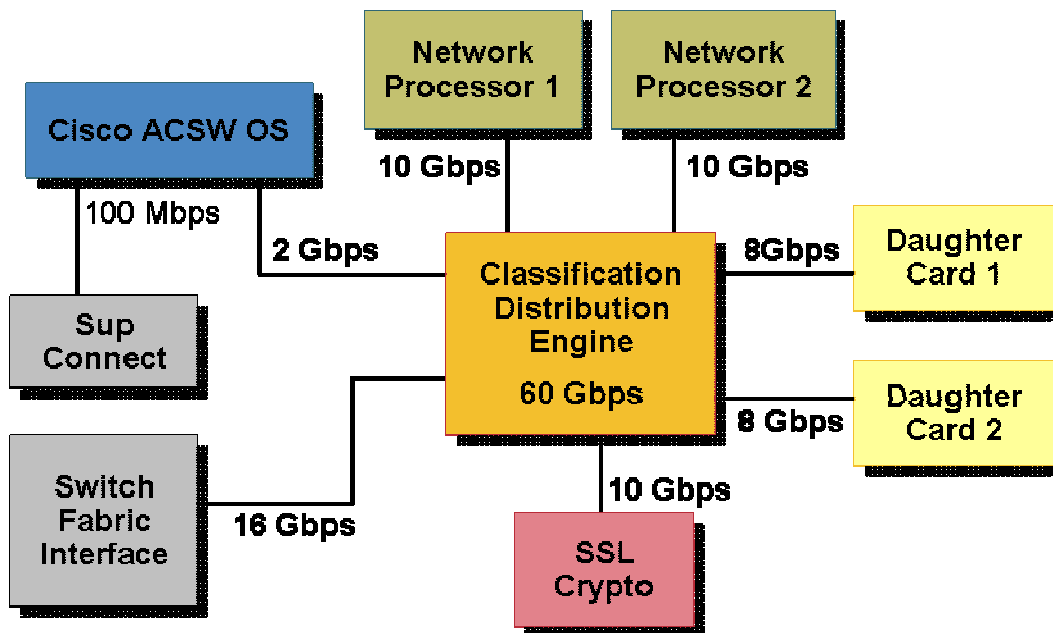
- Device management through the command-line interface (CLI), MIBs, syslogs, etc.
- Address Resolution Protocol (ARP) resolution
- Routing table maintenance and routing protocols (only static routing is supported at this time)
- ACL compilation

The control plane can initiate connections for health monitoring of servers, logging of syslog messages, SNMP notification, and so forth. The control plane never initiates or receives connections or packets directly from the Cisco Catalyst 6500 Series. Control plane traffic always crosses the network processors as they provide advanced protection for the control plane. For example, packets such as ARP requests are automatically rate limited so that they do not overwhelm the control plane. In addition, an out-of-band (OOB) connection interconnects the main components (except the daughter card slots) through a peripheral component interconnect (PCI) bus (32 bits at 33 MHz). The OOB is used commonly by the control plane to directly access the memory of the data plane. The configuration manager, running in the control plane, uses the OOB when pushing down a new configuration or the resulting ACL merge list from an access list compilation. Each of these control plane communications occurs outside the data path. Thus, control plane traffic, even oversubscription, will not affect data path traffic when traversing the Cisco ACE Module.

### Data Plane

The data plane consists of three of the five main components: the CDE, the two network processors, and the crypto processor (Cavium NITROX II), as shown in Figure 2.

**Figure 2.** Data Plane Components



New connections enter the Cisco ACE Module from the Cisco Catalyst 6500 Series switching fabric. The `svclc` (service line card) command is used to allow VLAN traffic to access the Cisco ACE Module. The Cisco Catalyst 6500 Series Supervisor Engine 720 performs a destination MAC address lookup and identifies the Cisco ACE Module as a next hop, which is similar to the way other service module traffic is handled. After it is allowed to the Cisco ACE Module, the initial packet of the connection goes to the CDE, where it is buffered for a very short time. The CDE is a FPGA developed by Cisco that can be thought of as a switch inside the switch. The CDE is a smart switch residing on the base board and acts as a central point of contact between all the main components located on the module. The CDE computes and, if necessary, adjusts the IP, TCP, and UDP checksums of every packet. It also performs virtual output queuing (VoQ) between the components—control plane, fast path, NITROX, and daughter card slots—to help ensure proper delivery of internal communications. The CDE appends a special header known as the IMPH header to each packet before sending it to the fast path. The IMPH header is 18 bytes long and contains information from the DBUS header (the header sent to the Cisco ACE Module by the Cisco Catalyst 6500 Series Switch) as well as special messaging directly understood by the fast path. Fields in the IMPH header can include notification of a checksum error, Layer 3 or 4 offsets, source and destination ports of the CDE, VLAN for determining the interface that the fast path will use, and so forth.

Most important, the CDE is responsible for determining where to send a particular packet. Currently, this decision is fairly simple as there are only two possible options: network processor 1 or network processor 2. After the Cisco ACE Module daughter cards are released, the CDE solution set will expand to include them. The control plane and crypto complex are not part of the CDE solution set, because they can receive traffic only from the network processors. All traffic entering the Cisco ACE Module always must traverse one (or both) of the network processors, including traffic destined for the control plane. Considering the control plane as a host behind a firewall, where the network processors provide the protection, makes the CDE destination selection easier to understand. To make its path decision, the CDE uses a hash algorithm using information from the incoming packet and produces a 5-bit-wide result. From this, up to 32 internal ports can be addressed (although only 7 exist currently; refer to Figure 1 for details). The hash algorithm is based on the following:

- Source and destination ports for UDP and TCP traffic
- Source and destination IP for IP traffic
- Source and destination MAC address for non-IP traffic

The load-sharing algorithm implemented by the CDE is not user configurable. The goal of the CDE is to help ensure that a given flow always hashes to the same network processor and to minimize the mega-proxy effects for commonly used protocols such as TCP and UDP. Although the CDE can be thought of as a switch, it does not perform any destination MAC address-based lookup. The CDE's main purpose is to compute a hash, append the IMPH header, and direct the packet to one of the network processors. In addition, the CDE implements VoQ: 16-KB-deep buffers are carved for each CDE port. These queues are implemented as First In First Out (FIFO) and perform tail-drop congestion management. Statistics are available for the internal CDE ports, such as queue-full condition indications and packet drops.

## Network Processors

After the CDE has made its path decision, the packet enters one of the network processors. Because all the real work of the Cisco ACE Module is done within the network processor, it is important to understand how processes are deployed within the network processors (Figure 3).

**Figure 3.** Process Deployment Within the IXP Network Processor

Receive	Fast Path	Fast Path	Fast Path
Fast Path	ICM	OCM	CCM
TCP	TCP	HTTP	HTTP
Unused (Future Expansion)	SSL Record Layer	IP Fragment- ation Timers	DNS and ICMP Inspection
Load Balancing	<b>XScale</b>		High-Availability Heartbeats
Application Inspection			SSL Handshake

Each network processor consists in 16 microengines and one XScale microprocessor. Each microengine runs at 1.4 GHz and has support for eight hardware threads. A thread is essentially a sequence of instructions capable of executing in parallel with other tasks. Threads are used within the microengine to allow for two or more simultaneously running tasks. The microengines are grouped in functional sets, as follows:

- One microengine for receiving incoming packets
- Four microengines for the fast-path: MAC rewrite, NAT, TCP normalization, etc. (basically all operations performed on a per-packet basis)
- Three microengines for the session management layer, further subdivided as follows:
  - Input connection manager (ICM)
  - Output connection manager (OCM)
  - Connection close manager (CCM)
- Two microengines for TCP termination with a full TCP stack
- Two microengines for HTTP parsing
- One microengine for SSL record layer processing
- One microengine for IP fragmentation reassembly and timer management
- One microengine for ICMP and DNS inspection
- One microengine unused in Cisco ACE Module Release 2.0, but will be available for future features

The XScale microprocessor is programmed to handle a certain set of features:

- Load-balancing algorithms
- SSL handshake
- FTP and Real-Time Streaming Protocol (RTSP) inspection
- HTTP inspection (although a considerable part is performed by the microengines)
- High-availability heartbeat generation
- Returned statistics for most connection-related commands

In addition, each network processor has access to 1.5 GB of RDRAM (433 MHz double data rate [DDR]) plus 32 MB of SRAM (200 MHz DDR). Memory is used to store items such as access list entries, routing table entries, ARP entries, and inspection policies. The SRAM provides faster access times and is used to store regular expressions and statistics on a per-virtual system basis, among other things.

### Connection Handling

As a connection is sent from the Cisco Catalyst 6500 Series Supervisor Engine 720 to the Cisco ACE Module, the initial connection packet is dispatched to either network processor 1 or network processor 2 by the CDE. The receive microengine accepts the packet from the CDE and stores it in a 256-MB buffer located in the DRAM of the network processor. After the packet is buffered, the receive microengine forwards a pointer to the packet to a fast-path microengine within the network processor. Within the fast path, checks are made to determine whether the packet matches an existing connection or whether it is a new connection requiring a connection setup. If the packet is a TCP or a UDP packet, a 5-tuple-flow lookup (source IP address, source port, destination IP address, destination port, and incoming VLAN) is performed to determine whether the flow has already been established.

If a new connection is required, the fast path sends the packet to the ICM. The ICM performs a 5-tuple lookup to determine which applications need to be applied to the flow. The output of the lookup is an access control entry, which is applied to permit or deny the traffic and also the list of actions to apply (such as load balance, inspection, and NAT).

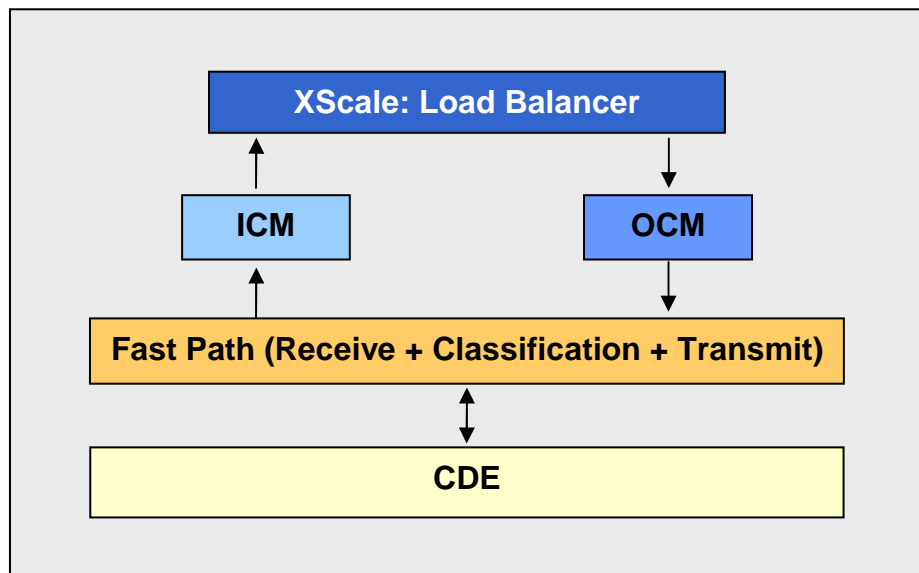
If the packet is denied, the packet is dropped, and a syslog message is sent to the syslog server directly from the ICM microengine, or the message is sent to the control plane, from which it is forwarded to the syslog server.

If the packet is permitted, further checks are performed against it, notably checks against invalid TCP flag combinations, sequence number verifications, TCP window checks, unicast Reverse Path Forwarding (RPF) check, etc. If the packet successfully passes the checks, a connection ID is created, and one of three other cases must exist:

- Connection is destined for a Layer 3 or 4 virtual IP address
- Connection is destined for a Layer 7 virtual IP address
- Connection is not destined for a virtual IP address
- Layer 3 or 4 Load Balancing

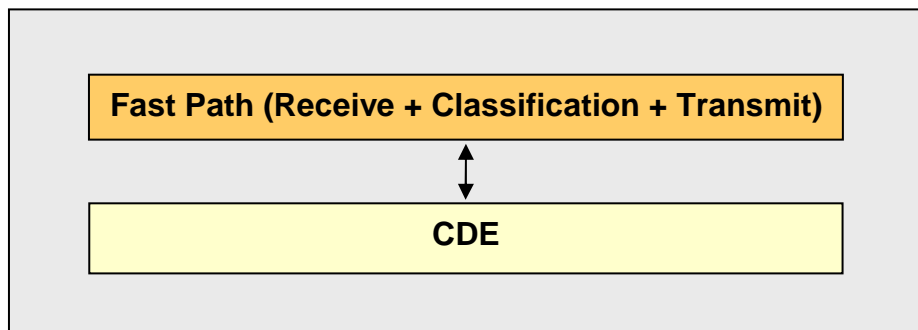
If the packet matches a virtual IP address with any port (Layer 3) or a well-defined destination port (Layer 4) the traffic is considered to be server load balancing (SLB) only. The logical packet flow is shown in Figure 4.

**Figure 4.** Processing a SLB-Only Connection

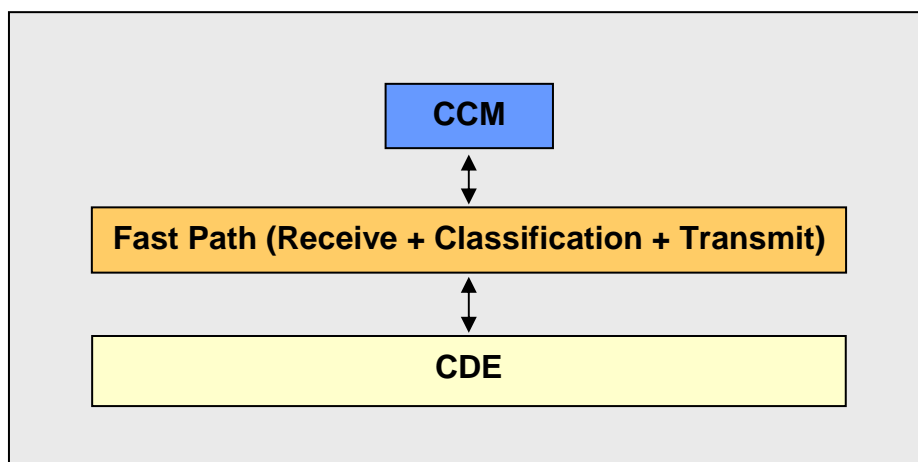


After the ICM microengine determines that the packet is destined to an SLB-only virtual IP address, the Layer 2 rewrite for server response traffic is set up in the inbound connection record by the ICM, and the ICM sends a message (not the full packet) to the XScale processor. Depending on the nature of the policy configured on the interface, the ICM microengine may send a reference pointer to the full packet to the XScale for more detailed processing, as is done with FTP- and RTSP-inspected traffic. The XScale processor is responsible for making the load-balancing decision. It returns a result (the real server that the connection should use) to the OCM microengine. If the Cisco ACE Module is configured to perform NAT or Port Address Translation (PAT) on the connection, the OCM will allocate a client NAT or PAT address to help ensure that the CDE sends return flows to the same network processor that the initial packet traversed. This process helps ensure network processor persistence per connection, even for NAT and PAT configurations. The OCM updates the connection object to include the Ethernet and IP rewrites and forwards the packet to the fast path. The fast path rewrites the Ethernet and IP headers and sends the packet to the CDE. The CDE, in turn, forwards the packet to the Cisco Catalyst 6500 Series Supervisor Engine 720 for correct Layer 2 or 3 forwarding to the real server.

As subsequent packets from this connection arrive, they are directed, by the CDE, to the same network processor that handled the initial packet. These packets are received by the receive microengine and pushed to the fast path. When the fast path makes the 5-tuple lookup, it finds an established connection. At this point, the fast path performs the appropriate rewrite (Ethernet, IP, and port) as defined by the connection object information. The packets are then sent to the CDE to be delivered to the server or client as required; see Figure 5.

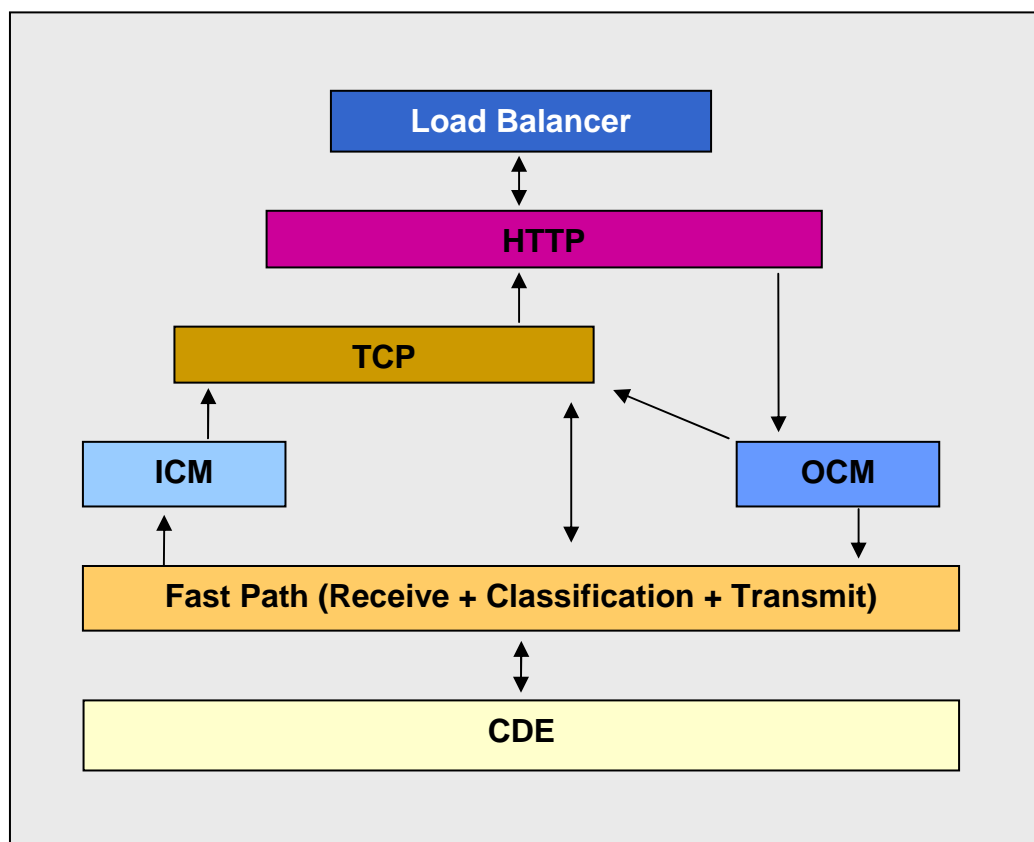
**Figure 5.** Handling Subsequent SLB-Only Packets

When the Cisco ACE Module receives a TCP packet with a Finish (FIN) or Reset (RST) flag set, the fast path instructs the CCM to tear down the connection, as shown in Figure 6.

**Figure 6.** Closing an SLB-Only Connection

### Layer 7 Load Balancing

If the initial connection packet matches a virtual IP address and port (any or well defined) and matches an application load-balancing or application inspection class map, it is considered a Layer 7 SLB connection. Typically, Layer 7 virtual IP addresses handle HTTP traffic where the URLs are used to distribute to application services. However, many types of Layer 7 virtual IP addresses are supported by the Cisco ACE Module: TCP reuse, HTTP pipelining, SSL acceleration, voice-over-IP (VoIP) protocol inspections, and so forth. For these connections, the initial connection packet does not contain enough data to make a load-balancing decision; thus, additional processing is required. Figure 7 illustrates the steps for handling a connection for a Layer 7 flow.

**Figure 7.** Processing a Layer 7 Virtual IP Address for HTTP Traffic

Upon receiving the initial packet for a Layer 7 SLB connection, the ICM performs the same check as for SLB-only connections. The packet is then forwarded to the TCP microengines for TCP termination of the connection between the client and the Cisco ACE Module. The TCP module initializes its state for this flow and responds to the sender with a synchronize acknowledgment (SYN ACK), sending the TCP packet to the fast path for Layer 2 or 3 encapsulation and then to the client through the CDE. The client TCP ACK is received by the fast path and sent to the TCP microengine upon the 5-tuple lookup. The TCP microengine applies flag and sequence number checks. The TCP state machine is also responsible for handling other tasks such as calculating the round-trip time (RTT), sending an ACK, and adjusting the window size. At this point, the client connection is set as a TCP proxy. The Cisco ACE Module is now the TCP termination point of the client sourced connection, which enables application layer load balancing and inspection.

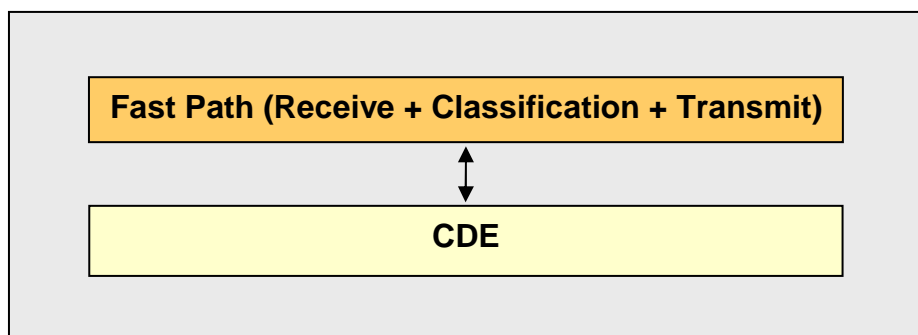
Upon receiving the client's HTTP request, the packet is sent to the HTTP microengine through the fast path and TCP microengines. If the TCP packets are out of order, the TCP microengine will buffer the packets until they can be reordered and sent to the HTTP microengine in the correct order. The HTTP microengine will parse the client request for a match on the URL, HTTP header, cookies, and so forth. After a match is found, the HTTP microengine sends a message to the XScale processor to perform load balancing.

The load-balancing destination decision is sent directly from the XScale processor to the OCM, which then sets up the outbound connection to the real server, where the Ethernet and IP layer rewrites are determined and updated in the connection object. Next, the OCM forwards the packet to the TCP microengine to establish a (or reuse any available) TCP connection to the real server. At this point, TCP initiates a connection through the fast path and the CDE to the real server.

When the server responds with a TCP SYN ACK, the CDE sends the packet to the receive microengine and fast path, which forwards the packet to the TCP microengine. The TCP microengine performs the same TCP checks as were done on the client side of the connection. If After the checks are passed, TCP begins sending the server the client HTTP request as forwarded from the HTTP microengine.

After the TCP microengine has sent all the client data to the server and seen all the TCP ACK packets from the server, TCP then configures the fast path with the appropriate TCP sequence number deltas for this flow. By instructing the fast path to manage the remainder of the flow, TCP has, in effect, unproxied the TCP connection. As a result, the remainder of the connection will be switched by the fast path, as shown in Figure 8.

**Figure 8.** Handling Unproxied Layer 7 SLB Packets



When the Cisco ACE Module receives a TCP packet with a FIN or RST flag set, the TCP microengine notifies the CCM, which then tears down the connection, similar to way that SLB-only connections are removed (see Figure 6).

To conserve resources, the Cisco ACE Module attempts to unproxy TCP connections as quickly as possible, enabling higher scalability and performance. However, to properly handle client traffic, it is often necessary to have parts of the connection proxied to allow the Cisco ACE Module to make a load-balancing decision. This process is called reproxy, and it is used most commonly to handle persistence rebalance, FTP inspection, and so forth. When a connection is initially proxied for Layer 7 load balancing, the connection is not unproxied until all the client data has been sent from the server and acknowledged. In addition, the Cisco ACE Module inspects the server HTTP response to determine the amount of data that the server will return to the client. After the data size is determined, the connection object is updated, so that when the fast path sees the anticipated TCP sequence number, it pushes the packet to TCP for reproxy and inspection. In the event that the HTTP server responds with a chunked-encoded HTTP response, the connection will continue to be fully proxied, because there is no way to calculate the TCP sequence number upon which to reproxy.

### **NAT Translated and Inspected Connections**

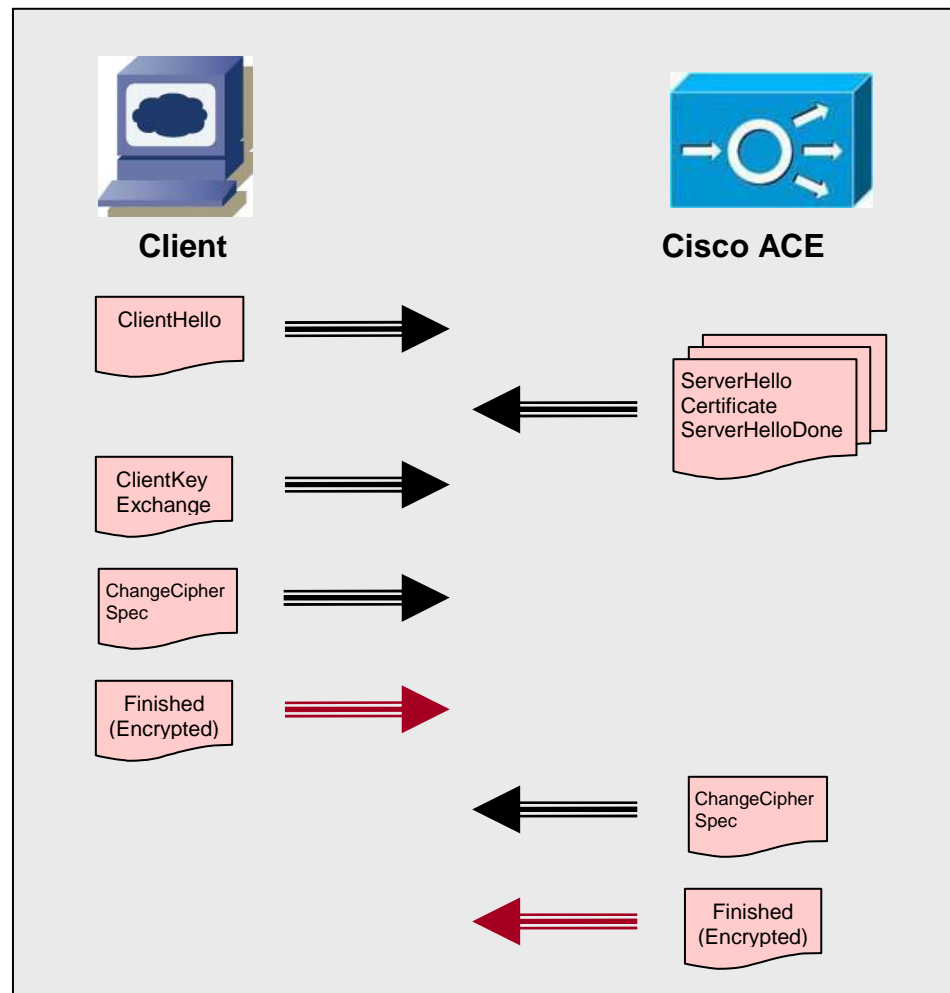
The previous sections discussed the actions that the Cisco ACE Module takes when a connection matches a virtual IP address. If the initial packet does not match any virtual IP address, the ICM engine directly instructs the OCM engine to insert a new connection entry and route the packet out. An indication is not sent to the XScale processor; instead, the data plane gathers the statistics and creates syslogs for the new packet. These statistics are moved to the service policy statistics after the connection is closed. The packet is routed out to the CDE with a new IMPH header that informs the CDE that the packet is destined for the Cisco Catalyst 6500 Series switching fabric.

This is the best-case scenario from a performance standpoint, as the life of the packet is very simple, and only a minimal number of engines are used.

### SSL-Terminated Connections

The Cisco ACE Module handles SSL-terminated connections at Layer 7 as full proxy TCP connections. Thus, the initial setup is very similar to the setup defined in the previous section discussing Layer 7 load balancing. The main difference in setup is that the ICM microengine determines that the virtual IP address requires SSL application handling. The ICM microengine then instructs the fast path to send the SSL ClientHello message to the TCP microengine. The TCP microengine then passes a reference to the message to the SSL record layer microengine, rather than sending the message directly to the HTTP microengine, as is done in Layer 7 load balancing. Figure 9 shows the external SSL messaging involved in terminating an SSL session.

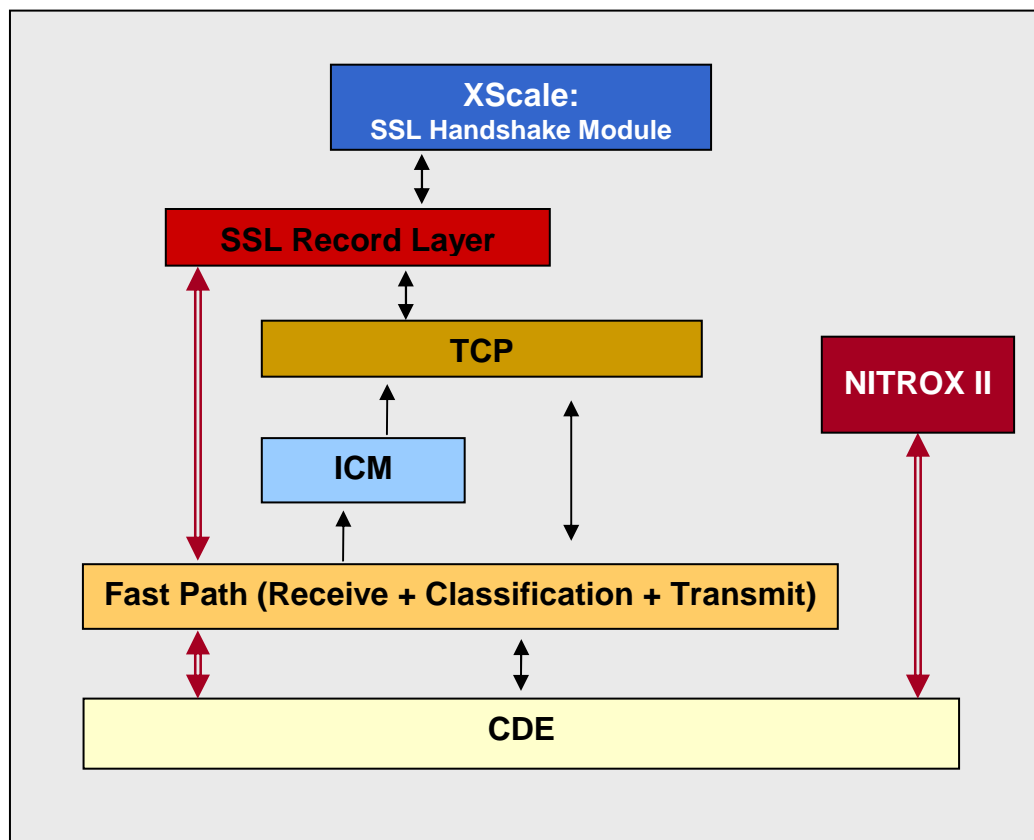
**Figure 9.** SSL Handshake Messaging



After the SSL Record Layer microengine receives the ClientHello message, it sends a reference pointer for the packet to the SSL handshake module, within the XScale processor, to process the SSL handshake message. The SSL handshake module picks a random number from pregenerated random numbers that were passed to it by the SSL record layer microengine. As random numbers are used, the list of number is repopulated by the SSL record layer microengine. The SSL handshake module then generates a ServerHello message, which is passed through the

SSL record layer for memory address translation and then to TCP for transmission. In addition, the SSL handshake module pulls the server certificate from the virtual IP address connection object to form the certificate message and generates a ServerHelloDone message. After these messages are created by the SSL handshake module, they are passed to the SSL record layer microengine, which passed them to TCP, the fast path, and the CDE for transmission to the client. Figure 10 shows the logical processing of the SSL termination handshake.

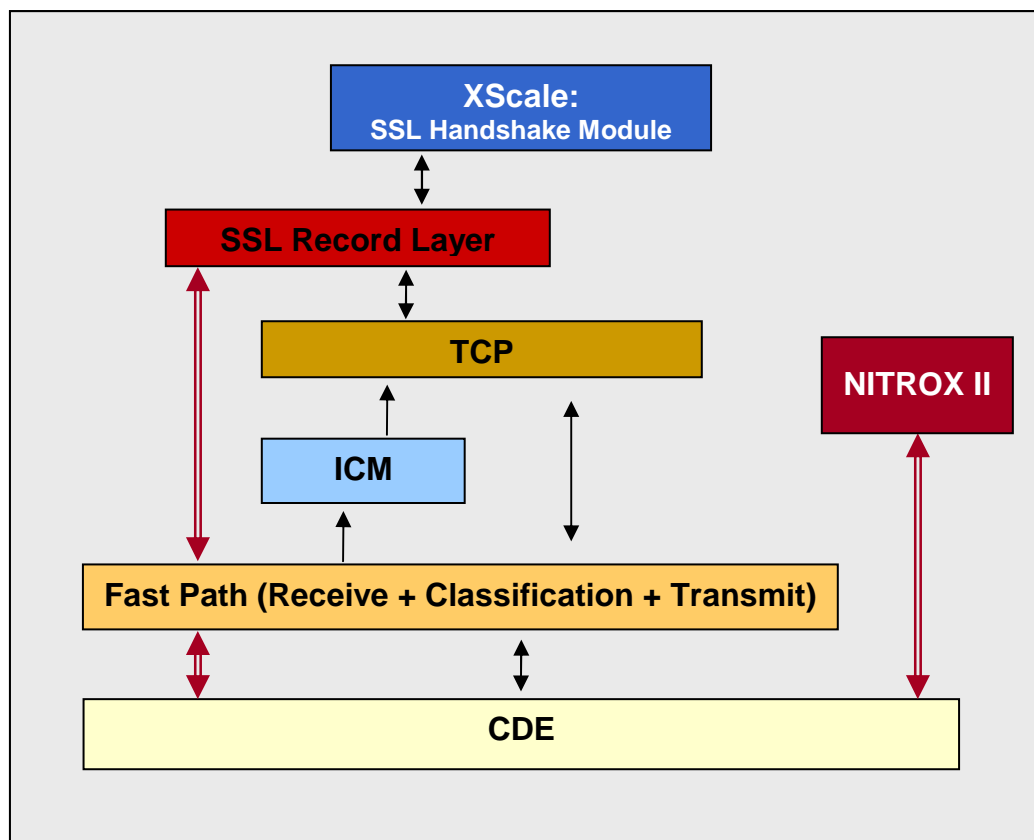
**Figure 10.** SSL Processing of ClientHello, ServerHello, Certificate, and ServerHelloDone messages



When the client responds, it typically sends the ClientKeyExchange (CKE), ChangeCipherSpec (CCS), and Finish messages as three SSL record messages combined in one TCP packet. The CDE sends the packet to the receive microengine and fast path, which forwards the packet to the TCP microengine. TCP then passes a reference to the packet to the SSL record layer microengine, where the messages are processed and passed to the SSL handshake module. The SSL record layer also creates IMPH headers to enable the CDE to send the CKE and Finish messages to the NITROX II. The NITROX II uses the CKE to compute the master secret, which is then stored in the DDR memory of the NITROX II. The SSL record layer microengine informs the handshake layer of the ClientKeyExchange message. When the client Finished message is received by the NITROX II, the Rivest, Shamir, and Adelman (RSA) parameters are verified to help ensure that they have been exchanged correctly. The NITROX II notifies the SSL record layer microengine of the verification outcome through the CDE. After it is notified, the SSL record layer microengine relays the notification to the SSL handshake module within the XScale processor. The SSL handshake module generates a ChangeCipherSpec message, which is passed to both the SSL record layer and TCP microengines. The message is then forwarded to the client, first through the fast path and then through the CDE.

If the RSA parameters are correct, the NITROX II generates the Finished message and sends it to the SSL record layer through the CDE. The SSL record layer notifies the SSL handshake module and passes the Finish messages to the SSL record layer microengine. The Finish message is then passed to TCP for forwarding to the client through the fast path and CDE. Figure 11 shows the completion of the SSL termination process.

**Figure 11.** SSL Processing of CKE and Finish Messages



As encrypted application packets arrive in the system through the CDE, receive microengine, and fast path, the TCP module applies appropriate checks and passes in order data to the SSL record layer. The record layer reads the SSL record header to determine the length of the record and keeps accumulating data from TCP until it has a complete record. The SSL application data message is then transmitted to NITROX II for decryption. The CDE works most efficiently when messages do not exceed 4 KB in length. Because the SSL protocol supports up to 16-KB messages, any messages over 4 KB are split and handled as multiple messages to allow the CDE to send them to the NITROX II. The NITROX II returns the decrypted packet to the record layer, which then sends the data to the HTTP microengine, if Layer 7 load balancing is also required, or to TCP, for SLB-only connections, for additional processing. At this point, the connection is processed as either Layer 4 or, more commonly, Layer 7 load balancing. If the traffic is Layer 7 load balanced, the decrypted HTTP request is sent to the HTTP microengine for processing as described earlier in this document. Regardless of the load-balancing processing used, after the correct real server is selected, the TCP microengine establishes a TCP connection to the real server through the fast path and CDE. In addition, the clear-text return traffic from the real server is sent to the SSL record layer, and then to the NITROX II, through the CDE for encryption. The NITROX II then uses the CDE to send the encrypted data back to the SSL record layer and then to the TCP microengine, fast path, and CDE for delivery to the client. The Cisco ACE Module

maintains the full proxy Layer 7 connection to allow for encryption and decryption of the client and server traffic.

## Conclusion

The Cisco ACE Module enables application and networking departments to achieve their business goals through a broad set of proven load-balancing and application switching technologies, integrated with leading-edge application acceleration and security capabilities. A primary design element of the Cisco ACE Module, and major differentiator with other solutions in the marketplace, is its network processor architecture and its complete separation between control and data planes. This design prevents control plane traffic, even in the event of oversubscription, from adversely affecting client-to-server traffic within the data plane. The Cisco ACE Module terminates TCP traffic once, decrypts SSL, inspects traffic to help ensure protocol adherence, applies application layer rules to help ensure proper application delivery and application persistence, and can then optimize TCP use between the Cisco ACE Module and the application server. This integrated approach minimizes latency and helps ensure the fastest application delivery solution for data center application service environments.



**Americas 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: 408 527-0883

**Asia Pacific Headquarters**  
Cisco Systems (USA) Pte. Ltd.  
168 Robinson Road  
#28-01 Capital Tower  
Singapore 068912  
www.cisco.com  
Tel: +65 6317 7777  
Fax: +65 6317 7799

**Europe Headquarters**  
Cisco Systems International BV  
Haarlerbergpark  
Haarlerbergweg 13-19  
1101 CH Amsterdam  
The Netherlands  
www-europe.cisco.com  
Tel: +31 0 800 020 0791  
Fax: +31 0 20 357 1100

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

©2007 Cisco Systems, Inc. All rights reserved. CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, IQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website 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. (0710R)