Content Load-Balancing Overview

Content load balancing is how the CSS handles requests for content to a specific destination. To assist you in understanding what occurs when load balancing occurs on the CSS, this chapter provides information about the relationship of service, owner, and content rules, and describes how the CSS handles TCP and UDP traffic. This chapter contains the following sections:

Information in this chapter applies to all CSS models except where noted.

Service, Owner, and Content Rule Overview

The CSS enables you to configure services, owners, and content rules in order to direct requests for content to a specific destination service (for example, a server or a port on a server). By configuring services, owners, and content rules, you optimize and control how the CSS handles each request for specific content. Services, owners, and content rules are described below:

- A **service** is a destination location where a piece of content resides physically (a local or remote server and port). You add services to content rules. Adding a service to a content rule includes it in the resource pool that the CSS uses for load-balancing requests for content. A service may belong to multiple content rules.

- An **owner** is generally the person or company who contracts the Web hosting service to host their Web content and allocate bandwidth as required. Owners can have multiple content rules.
- A **content rule** is a hierarchical rule set containing individual rules that describe which content (for example, an .html file) is accessible by visitors to the Web site, how the content is mirrored, on which server the content resides, and how the CSS should process requests for the content. Each rule set must have an owner.

  The CSS uses content rules to determine:

  – Where the content physically resides, whether local or remote

  – Where to direct the request for content (which service or services)

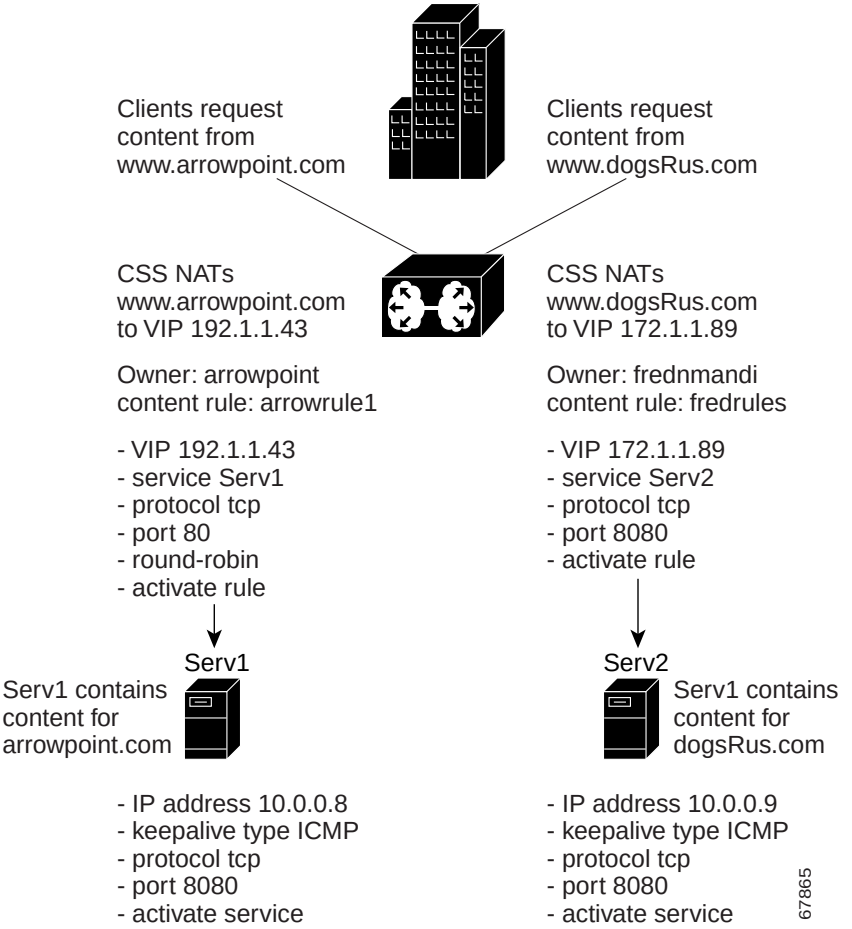
  – Which load balancing method to use

When a request for content is made, the CSS:

1. Uses the owner content rule to translate the owner Virtual IP address (VIP) or domain name using Network Address Translation (NAT) to the corresponding service IP address and port.
2. Checks for available services that match the content request.
3. Uses content rules to choose which service can best process the request for content.
4. Applies all content rules to service the request for content (for example, load-balancing method, redirects, failover, stickiness).

Figure 1-1 illustrates the CSS service, owner, and content rule concepts.

Figure 1-1 Services, Owners, and Content Rules



The following chapters provides information on configuring services, owners, and content rules:

- Chapter 1, Configuring Services
- Chapter 9, Configuring Owners
- Chapter 10, Configuring Content Rules

For information on how TCP and UDP traffic flows through the CSS, see the following "Overview of CSS Flow" section.

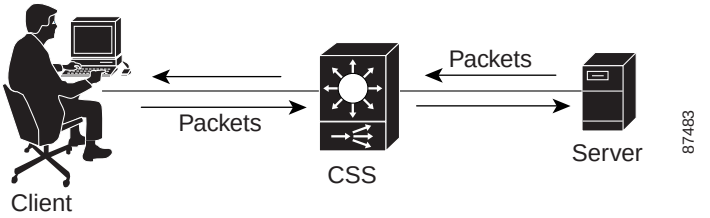
Overview of CSS Flow

A flow is the transfer of a sequence of related packets over a TCP or UDP connection between a source (client) and a destination (server) through the CSS. All packets in an ingress flow (traffic entering the CSS) share a common 5-tuple consisting of:

- Source address
- Destination address
- Protocol
- Source port
- Destination port

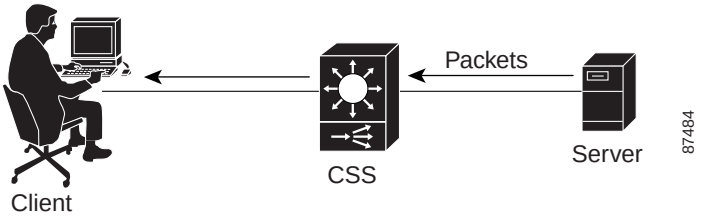
TCP flows are bidirectional (Figure 1-2). Packets move from the client to the server and from the server to the client through the CSS. Strictly speaking, a TCP connection consists of two flows, one in each direction. A TCP flow begins with a SYN and ends with an ACK to a FIN/ACK, or an RST.

Figure 1-2 Example of a TCP Flow



UDP flows (Figure 1-3) are typically unidirectional (for example, streaming audio transmitted by a server to a client). A UDP flow has no definitive beginning or end and is considered completed only after a period of time has elapsed during which the destination device receives no packets that share the same addresses, protocol, and ports that defined the original flow.

Figure 1-3 Example of a UDP Flow



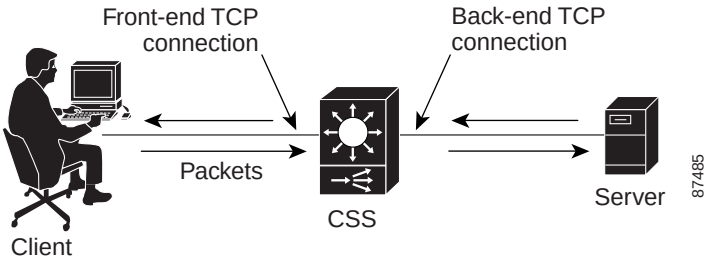
A CSS uses data structures called flow control blocks (FCBs) to set up and keep track of ingress flows. FCBs contain all the information the CSS needs to process and manage flows. The creation of an FCB from flow information is called *flow mapping*. The flow manager in each module session processor is responsible for FCB creation and flow mapping.

Each unidirectional flow uses one FCB. Therefore, a TCP flow uses two FCBs and a UDP flow typically uses one FCB. Front-end SSL, which runs over TCP, requires four FCBs and back-end SSL adds two more FCBs for a total of six FCBs per full-duplex SSL connection. For more information about SSL, refer to the *Cisco Content Services Switch SSL Configuration Guide*.

Each client-CSS-server connection consists of two parts (Figure 1-4):

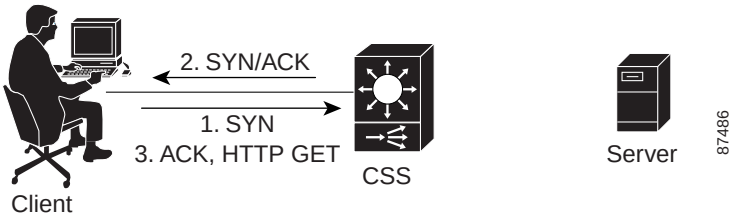
- Front-end - Connection between a client and the CSS
- Back-end - Connection between the CSS and a server

Figure 1-4 Example of a TCP Flow with Front-End and Back-End Connections



A Layer 5 flow begins with a client request for content. After the D-proxy resolves the DNS request (for example, a client types a URL in a Web browser) and points the client to the CSS virtual IP address (VIP), the CSS establishes the front-end TCP connection with the client using the TCP 3-way handshake (Figure 1-5).

Figure 1-5 Setting Up the Front-End TCP Connection - Delayed Binding



When it establishes a Layer 5 flow, a CSS “spoofs” the back-end TCP connection by acting as a proxy for the destination device (server) for the client SYN. In other words, the CSS responds to the client SYN with a SYN/ACK before the CSS sets up the back-end TCP connection with the server.

This process is referred to as *delayed binding*. Delayed binding causes the client to respond with an ACK and an HTTP GET request. This process allows the CSS to gather the information it needs to select the best service (a server port where content resides or an application running on a server such as FTP) for the content request.

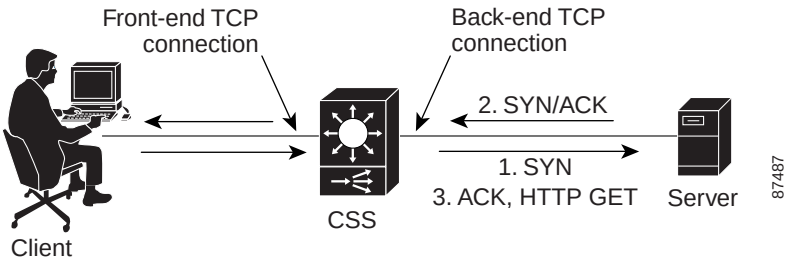
The CSS examines the HTTP header and URL in the HTTP request method (for example, GET, HEAD, or POST). Based on the information in the HTTP header, the URL, and the content rules configured on the CSS, the CSS selects the best site and the best service to satisfy the request. A CSS bases service selection (server load balancing) on factors such as:

- Content rule match
- Service availability
- Service load
- Cookies
- Source IP address

For more information about CSS server load balancing (SLB), see the "Service, Owner, and Content Rule Overview" section.

After the CSS selects the best service to provide the requested content to the client, the CSS establishes the back-end connection with the service using the TCP 3-way handshake and splices the front-end and back-end connections together. The CSS forwards the content request from the client to the service (Figure 1-6). The service responds to the client through the CSS. For the remaining life of the flow, the CSS switches the packets between the client and the service, and performs network address translation (NAT) and other packet transformations as required.

Figure 1-6 Setting Up the Back-End TCP Connection - Delayed Binding



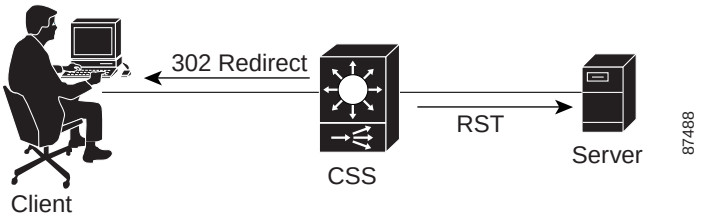
For subsequent content requests from the same client over the same TCP connection (HTTP 1.1 and higher), the CSS attempts to maintain the back-end connection with the service that provided the content for the first HTTP request by default. This condition is called *persistence*.

During the life of a persistent connection, a CSS must determine if it needs to move a client connection to a new service based on content rules, load balancing, and service availability. In some situations, moving the client connection is not necessary; in other situations, it is mandatory.

You can configure the CSS to perform one of the following functions when it becomes necessary to move a client to a new service:

- HTTP redirection - Using the **persistence reset redirect** command, a CSS closes the back-end connection by sending a RST to the service (Figure 1-7). The CSS sends a 302 redirect to the client's browser to tell the browser to reconnect using the same DNS name, but this time the HTTP request matches on a different content rule. The CSS then establishes a new flow between the client and the best service.

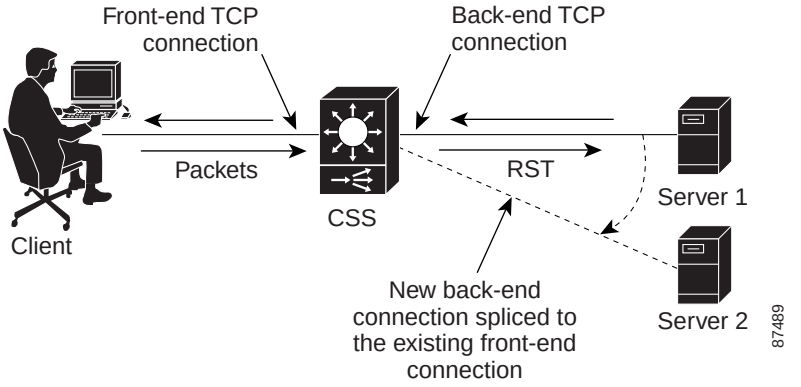
Figure 1-7 Example of HTTP Redirection



- Service remapping - Using the **persistence reset remap** command, a CSS closes only the back-end connection by sending a RST to the service (server 1 in Figure 1-8), then establishes a new back-end connection with service server 2 and splices the back-end and front-end connections together. The CSS forwards the content request from the client to server 2. Packets now flow between the client and server 2.

  For more information about persistence, HTTP redirection, and service remapping, see Chapter 10, Configuring Content Rules.

Figure 1-8 Example of Remapping the Back-end Connection



Periodically, the CSS flow manager tears down old, idle flows and reclaims the system resources (FCBs). This process is called *flow resource reclamation*. It is also referred to as *flow cleanup* or *garbage collection*. Flow resource reclamation involves removing FCBs from the TCP and UDP lists. For optimal performance, the CSS reuses FCBs that are no longer needed for flows.

Normally, flow cleanup occurs at a rate that is directly related to the total number of flows that are currently active on a CSS. A CSS always cleans up UDP flows. For TCP flows, a CSS reclaims resources when the number of used FCBs reaches a certain percentage of the total FCBs. A CSS also cleans up long-lived TCP flows that have received a FIN or a RST, or whose timeout values have been met. You can configure various commands to change the default flow-cleanup behavior of the CSS.

In some instances it may not be desirable for the CSS to clean up idle TCP flows. For example, during a connection to a database server that must permanently remain active even when no data passes through the connection. If you observe the CSS dropping long-lived idle connections that need to be maintained you can configure the following TCP flow commands:

- **flow permanent** command - Creates permanent TCP or UDP ports that are not reclaimed
- **flow-timeout-multiplier** command - Configures flow inactivity timeout values for TCP and UDP flows on a per content rule and per source group basis

Refer to Chapter 2, Configuring Flow and Port Mapping Parameters for information on the commands you can use to control how the CSS handles and cleans up TCP and UDP flows.