Understanding Slow Start

When you configure a NetScaler to use a metric-based LB method such as Least Connections, Least Response Time, Least Bandwidth, Least Packets, or Custom Load, the LB method will initially start out as Round Robin for what is called a slow start.

As mentioned earlier, NetScaler appliances use the configured load balancing method to determine the appropriate service for forwarding an incoming request. Load balancing environments are dynamic, and the NetScaler needs to manage the events that may overload the server. For example, when you configure the Least Connections LB method, the NetScaler selects the service that has the least number of connections. If a new server is added to the server farm, the NetScaler selects the new server with the least number of connections, and, therefore, may overload the new server.

To avoid overloading servers, the NetScaler performs slow start. During the slow start phase, the NetScaler distributes requests by using Round Robin, regardless of the metric-based LB method configured on the virtual server. However, the weight assigned on the services is used by Round Robin. After the number of incoming requests or connections per second exceeds a given threshold, the NetScaler stops slow start and operates using the configured load balancing method.

Slow start occurs when:

- You configure a new metric-based load balancing method.
- You bind a service to the vserver.
- The state of the server changes from DOWN to UP.

To compute slow start, the number of services bound to the vserver is multiplied by 100. For a new virtual server with the LB method determined by dynamic traffic parameters, slow start allows time to collect a valid data sample before the correct method is applied.

Note: When slow start is in operation, the output for the `show lb vserver <vserver name>` command will specify the current method as Round Robin.

In GSLB setup, metric-based load balancing methods do not work correctly if MEP is DOWN except Custom Load LB method, and they will operate only in RoundRobin. For Custom Load if MEP is DOWN and custom load monitors that use SNMP to get statistics are bound to service, Custom Load LB method is used for load balancing. If local load monitors bound to service and MEP is DOWN, then Round Robin is used. For more information about GSLB, see Chapter 8, “Global Server Load Balancing.”

Configuring the Least Connection Method

When NetScaler is configured to use the least connection method, it selects the service with the least number of active connections to ensure that the load of the active requests is balanced on the services. This method is the default load balancing method because it provides the best performance. For TCP, HTTP, HTTPS, and SSL_TCP services, the following connections are also considered for the least connection method:

- **Active connections to a service.** Active connections represent the requests sent by the client to a service. However, in case of HTTP and HTTPS services, active connections represent only the outstanding HTTP or HTTPS requests to services.
- **Waiting connections to a service in the surge queue.** Connections can build up in the surge queue at any time because of any of the following reasons:
  - A connection limit exists on the service.
  - The surge protection feature is configured.
  - The server does not open new connections as in case of Apache's connection limit.

When a NetScaler uses the least connection method, it considers such waiting connections as belonging to a service. Therefore, it does not open new connections to the selected service in a timely manner.

For UDP services, the connections considered for the least connection method include all sessions between the client and a service. These sessions are logical, time-based entities and are created for the UDP packet that arrives first. When the UDP packet arrives first, the session is created for the combination of the source IP address and port and the destination IP address and port.

For Real-Time Streaming Protocol (RTSP) connections, NetScaler uses the number of active control connections to determine the least number of connections to an RTSP service.

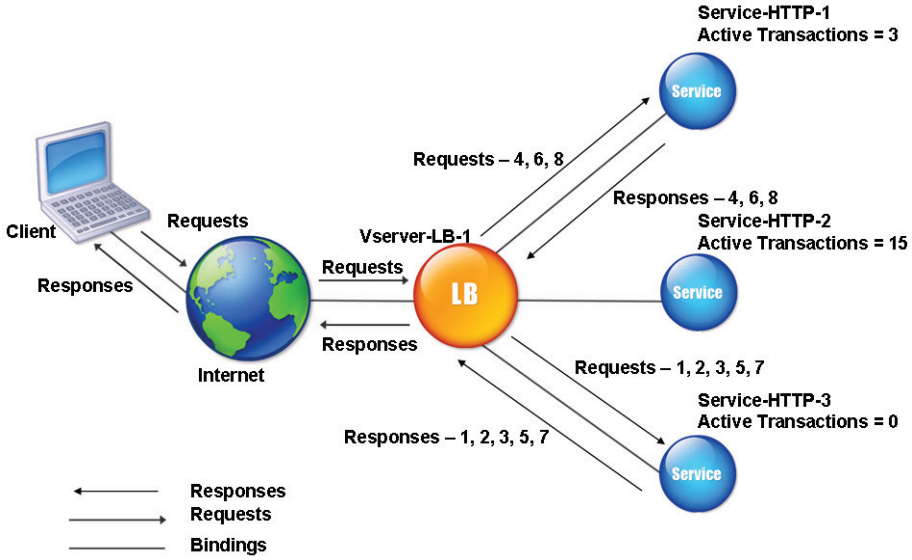
Note: RTSP is not supported in NetScaler 9.1 nCore.

The following example shows how a NetScaler selects a service for load balancing by using the least connections method. Consider the following three services:

- Service-HTTP-1 is handling 3 active transactions.
- Service-HTTP-2 is handling 15 active transactions.

- Service-HTTP-3 is not handling any active transactions.

The following diagram illustrates how the NetScaler uses the least connection method and forwards the requests to the three services.



Least connection mechanism

The NetScaler selects the service by using the value (*N*) of the following expression:

N = Number of active transactions

The requests are delivered as follows:

- Service-HTTP-3 receives the first request because the service is not handling any active transactions.

  Note: The service with no active transaction is selected first.

- Service-HTTP-3 receives the second and third requests because the service has the next least number of active transactions.
- Service-HTTP-1 receives the fourth request. Because Service-HTTP-1 and Service-HTTP-3 have same number of active transactions, NetScaler performs load balancing in a round robin manner. Therefore, Service-HTTP-3 receives the fifth request, Service-HTTP-1 receives the sixth request, Service-HTTP-3 receives the seventh request, and Service-HTTP-1 receives the eighth request and so forth.

Service-HTTP-2 is not considered for load balancing because it is loaded more (handling 15 active transactions) as compared to the other two services. However, if Service-HTTP-2 completes the active transactions, NetScaler considers the service for load balancing. Also, when the services are handling the same number of active transactions, NetScaler selects the service in a round robin manner.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Examples of Least Connection Method Service Selection: N

Request received	**Service selected**	Current ***N*** ***(Number of active transaction)*** value	**Remarks**
Request-1	Service-HTTP-3 (*N* = 0)	*N* = 1	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-3 (*N* = 1)	*N* = 2	
Request-3	Service-HTTP-3 (*N* = 2)	*N* = 3	
Request-4	Service-HTTP-1 (*N* = 3)	*N* = 4	Service-HTTP-1 and Service-HTTP-3 have the same *N* values.
Request-5	Service-HTTP-3 (*N* = 3)	*N* = 4	
Request-6	Service-HTTP-1 (*N* = 4)	*N* = 5	
Request-7	Service-HTTP-3 (*N* = 4)	*N* = 5	
Request-8	Service-HTTP-1 (*N* = 5)	*N* = 6	
Service-HTTP-2 is selected for load balancing when it completes the active transactions or when the *N* value of other services (Service-HTTP-1 and Service-HTTP-3) is equal to 15.			

Using Weights with the Least Connection Method

The NetScaler also performs load balancing by using the number of connections when weights are assigned to services. The NetScaler selects the service by using the value (N_w) of the following expression:

N_w = *(Number of active transactions) * (10000 / weight)*

The following example shows how the NetScaler selects a service for load balancing by using least connections method when the weights are assigned to services.

In the preceding example, suppose Service-HTTP-1 is assigned a weight of 2, Service-HTTP-2 is assigned a weight of 3, and Service-HTTP-3 is assigned a weight of 4. The requests are delivered as follows:

- Service-HTTP-3 receives the first because the service is not handling any active transactions.

  Note: If services are not handling any active transactions, NetScaler selects them in a round robin manner regardless of the weights assigned to them.

- Service-HTTP-3 receives the second, third, fourth, fifth, sixth, and seventh requests because the service has least N_w value.
- Service-HTTP-1 receives the eighth request. Because Service-HTTP-1 and Service-HTTP-3 have same Nw value, the NetScaler performs load balancing in a round robin manner. Therefore, Service-HTTP-3 receives the ninth request.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

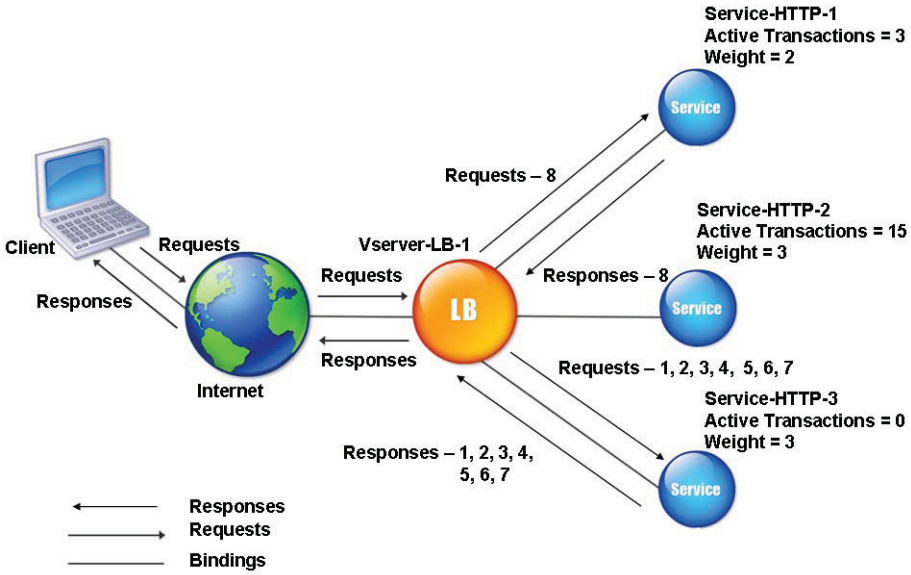
Example of Least Connection Method Service Selection: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-3 (N_w = 0)	N_w = 2500	Service-HTTP-3 has the least N_w value.
Request-2	Service-HTTP-3 (N_w = 2500)	N_w = 5000	
Request-3	Service-HTTP-3 (N_w = 5000)	N_w = 7500	
Request-4	Service-HTTP-3 (N_w = 7500)	N_w = 10000	
Request-5	Service-HTTP-3 (N_w = 10000)	N_w = 12500	
Request-6	Service-HTTP-3 (N_w = 12500)	N_w = 15000	
Request-7	Service-HTTP-1 (N_w = 15000)	N_w = 20000	Service-HTTP-1 and Service-HTTP-3 have the same N_w values.
Request-8	Service-HTTP-3 (N_w = 15000)	N_w = 17500	

Example of Least Connection Method Service Selection: N_w

Request received	Service selected	Current N_w *(Number of active transactions) * (10000 / weight)* value	Remarks
Service-HTTP-2 is selected for load balancing when it completes the active transactions or when the N_w value of other services (Service-HTTP-1 and Service-HTTP-3) is equal to 50000.			

The following diagram illustrates how the NetScaler uses the least connection method when weights are assigned to the services.



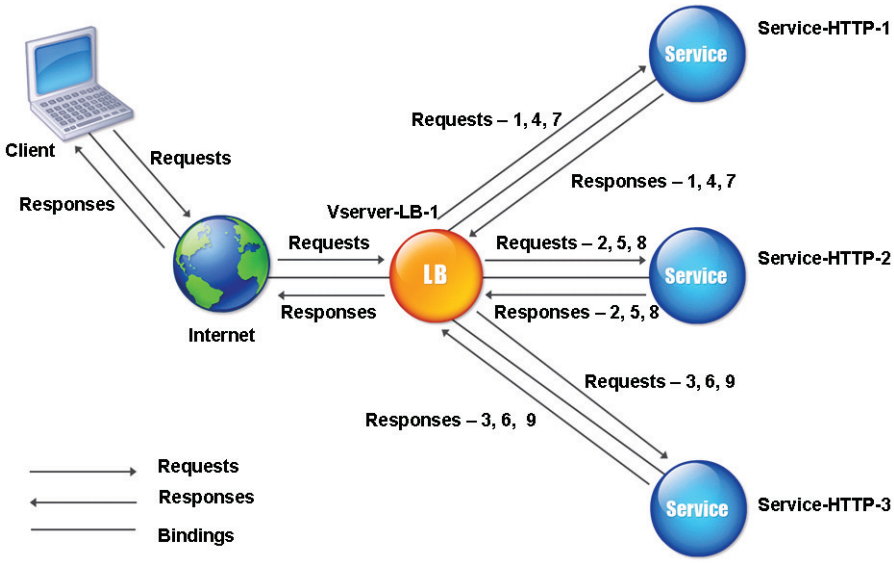
Least connections mechanism when weights are assigned

To configure the least connection method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Least Connection**.

Configuring the Round Robin Method

When the NetScaler is configured to use the round robin method, it rotates incoming requests to the managed servers, regardless of the load. For example, the first request is sent to Service-HTTP-1, the second to Service-HTTP-2, the third to Service-HTTP-3, and so forth. When requests have been sent to all of the servers, the cycle begins again from Service-HTTP-1.

The following diagram illustrates how the NetScaler uses the round robin method and forwards requests to the three services.



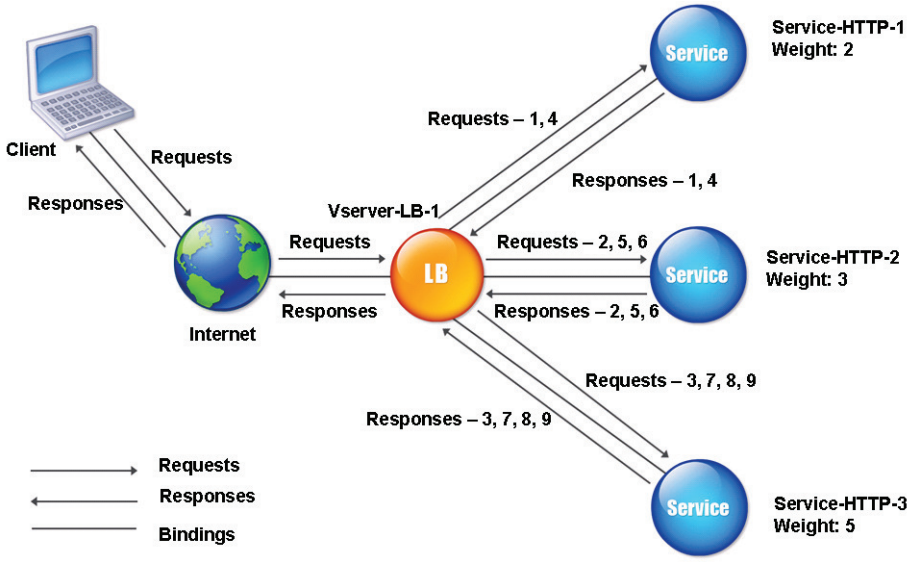
Round robin mechanism

A NetScaler also performs weighted round robin if different weights are assigned to the services. For example, Service-HTTP-1 is set to a weight of 2, Service-HTTP-2 to a weight of 3, and Service-HTTP-3 to a weight of 4, and the services are bound to Vserver-LB-1. The requests are delivered as follows:

- Service-HTTP-1 receives the first request.
- Service-HTTP-2 receives the second request.
- Service-HTTP-3 receives the third request.
- Service-HTTP-1 receives the fourth request.
- Service-HTTP-2 receives the fifth request.
- Service-HTTP-3 receives the sixth request.
- Service-HTTP-2 receives the seventh request.
- Service-HTTP-3 receives the eighth and ninth requests.

Note: You can configure weights on services to prevent multiple services from using the same server and overloading the server.

A new cycle then begins, using the same pattern. The following diagram illustrates the weighted round robin method.



Round robin mechanism when weights are configured

To configure the round robin method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Round Robin**.

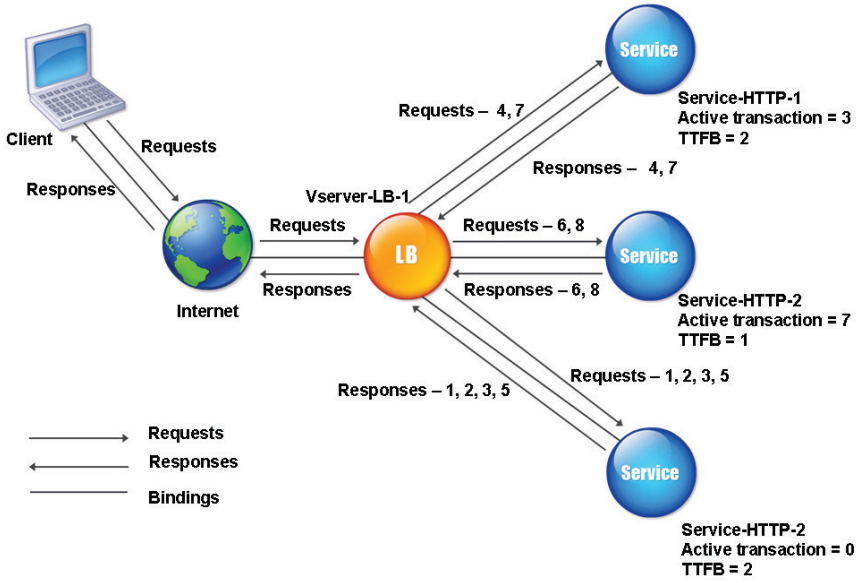
Configuring the Least Response Time Method

When the NetScaler is configured to use the least response time method, it selects the service with the least number of active connections and the least average response time. You can configure this method for HTTP and Secure Sockets Layer (SSL) type of services only. The response time also called Time to First Byte, or TTFB is the time interval between sending a request packet to a service and receiving the first response packet from the service. The NetScaler uses response code 200 to calculate TTFB.

The following example shows how a NetScaler selects a service for load balancing by using least response time method. Consider the following three services:

- Service-HTTP-1 is handling three active transactions and TTFB is two seconds.
- Service-HTTP-2 is handling seven active transactions and TTFB is one second.
- Service-HTTP-3 is not handling any active transactions and TTFB is two seconds.

The following diagram illustrates how the NetScaler uses the least response time method and forward requests to the three services.



Least response time mechanism

The NetScaler selects the service by using the value (*N*) of the following expression:

*N = Number of active transactions * TTFB*

The NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because the service is not handling any active transaction.

  Note: The service with no active transaction is selected first.

- Service-HTTP-3 receives the second and third requests because the service has the least *N* value.
- Service-HTTP-1 receives the fourth request. Because Service-HTTP-1 and Service-HTTP-3 have same *N* value, the NetScaler performs load balancing in a round robin manner. Therefore, Service-HTTP-3 receives the fifth request.
- Service-HTTP-2 receives the sixth request because the service has the least *N* value.

- Service-HTTP-1 receives the seventh request. Because Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have same *N* value, the NetScaler performs load balancing in a round robin manner. Therefore, Service-HTTP-2 receives the eighth request.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Examples of Least Response Time Method: N

Request received	**Service selected**	Current ***N*** ***(Number of active transaction * TTFB)*** value	**Remarks**
Request-1	Service-HTTP-3 (*N* = 0)	*N* = 2	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-3 (*N* = 2)	*N* = 4	
Request-3	Service-HTTP-3 (*N* = 3)	*N* = 6	
Request-4	Service-HTTP-1 (*N* = 6)	*N* = 8	Service-HTTP-1 and Service-HTTP-3 have the same *N* values.
Request-5	Service-HTTP-3 (*N* = 6)	*N* = 8	
Request-6	Service-HTTP-2 (*N* = 7)	*N* = 8	Service-HTTP-2 has the least *N* value.
Request-7	Service-HTTP-1 (*N* = 8)	*N* = 15	Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have the same *N* values.
Request-8	Service-HTTP-2 (*N* = 8)	*N* = 9	

Using Weights with the Least Response Time Method

The NetScaler also performs load balancing by using the number of connections, TTFB, and weights if different weights are assigned to the services. The NetScaler selects the service by using the value (N_w) of the following expression:

$N_w = (N) * (10000 / weight)$

The following example shows how the NetScaler selects a service for load balancing by using the least response time method when weights are assigned on the services. In the preceding example, suppose Service-HTTP-1 is assigned a weight of 2, Service-HTTP-2 is assigned weight of 3, and Service-HTTP-3 is assigned weight of 4.

The NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because it is not handling any active transaction.

  Note: If services are not handling any active transactions, the NetScaler selects them regardless of the weights assigned to them.

- Service-HTTP-3 receives the second, third, fourth, and fifth requests because the service has the least N_w value.
- Service-HTTP-2 receives the sixth request because the service has the least N_w value.
- Service-HTTP-3 receives the seventh request because the service has the least N_w value.
- Service-HTTP-2 receives the eighth request because the service has the least N_w value.

Service-HTTP-1 has the least weight and the N_w value is the highest. Therefore, the NetScaler does not select it for load balancing.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

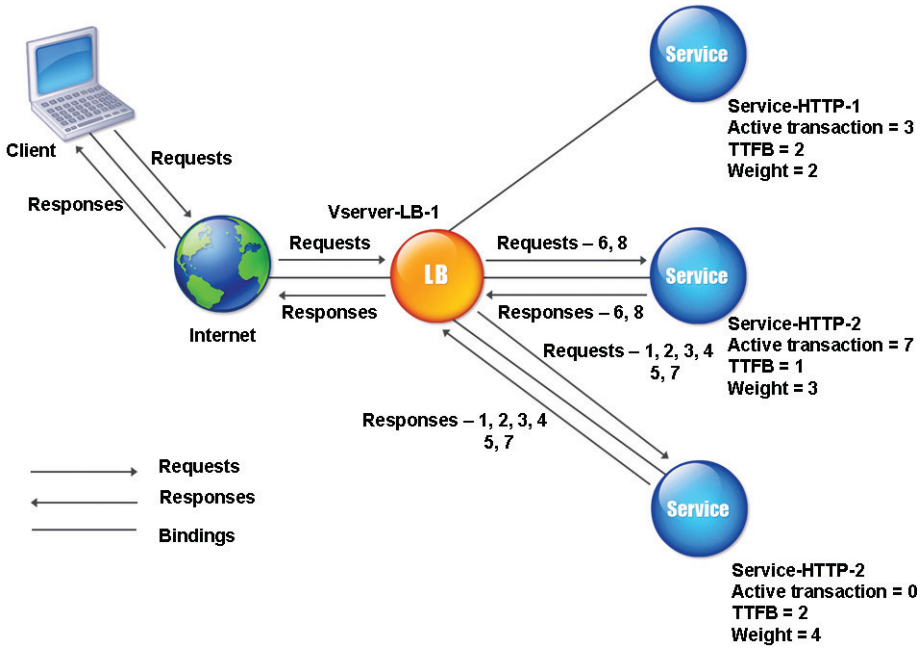
Examples of Least Response Time Method: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-3 (N_w = 0)	N_w = 2500	Service-HTTP-3 has the least N_w value.
Request-2	Service-HTTP-3 (N_w = 2500)	N_w = 5000	
Request-3	Service-HTTP-3 (N_w = 5000)	N_w = 15000	
Request-4	Service-HTTP-3 (N_w = 15000)	N_w = 20000	
Request-5	Service-HTTP-3 (N_w = 20000)	N_w = 25000	
Request-6	Service-HTTP-2 (N_w = 23333.34)	N_w = 26666.67	Service-HTTP-2 has the least N_w value.

Examples of Least Response Time Method: N_w

Request received	Service selected	Current N_w *(Number of active transactions) * (10000 / weight)* value	Remarks
Request-7	Service-HTTP-3 (N_w = 25000)	N_w = 30000	Service-HTTP-3 has the least N_w value.
Request-8	Service-HTTP-2 (N_w = 26666.67)	N_w = 33333.34	Service-HTTP-2 has the least N_w value.
Service-HTTP-1 is selected for load balancing when it completes the active transactions or when the N_w value of other services (Service-HTTP-2 and Service-HTTP-3) is equal to 105000.			

The following diagram illustrates how the NetScaler uses the least response time method when weights are assigned on the services.



Least response time mechanism when weights are assigned

To configure the least response time method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Least Response Time**.

Configuring the Least Response Time Method with Monitors

When the NetScaler is configured to use the least response time method with monitors, it selects the service with the least number of active transactions and the fastest average response time of monitors. With this load balancing method, the NetScaler uses the existing monitoring infrastructure. Therefore, before you use this method, you must bind application-specific monitors to each service and enable least response time method mode on these monitors. The NetScaler then makes load balancing decisions based on the response times received from the monitoring probes. For more information about configuring monitors, see the section "Configuring Monitors in a Load Balancing Setup," on page 162.

Least response time method with monitors can be used to select non-HTTP and non-HTTPS services unlike the least response time method without monitors. You can also use this method when several monitors are bound to a service. The vserver reads the response times of all monitors and calculates an average response time for each service. Monitors determine response times according to different protocols.

The following table summarizes how response times are calculated for various monitors.

Monitor Response Time Calculations

Monitor	Response time calculation
PING	Time difference between the ICMP ECHO request and the ICMP ECHO response.
TCP	Time difference between the SYN request and the SYN+ACK response.
HTTP	Time difference between the HTTP request (after the TCP connection is established) and the HTTP response.
TCP-ENV	Time difference between the time the data send string is sent and the data receive string is returned. A tcp-ecv monitor without the send and receive strings is considered to have an incorrect configuration.
HTTP-ECV	Time difference between the HTTP request and the HTTP response.
UDP-ECV	Time difference between the UDP send string and the UDP receive string. A udp-ecv monitor without the receive string is considered to have an incorrect configuration.
DNS	Time difference between a DNS query and the DNS response.
TCPS	Time difference between a SYN request and the SSL handshake completion.

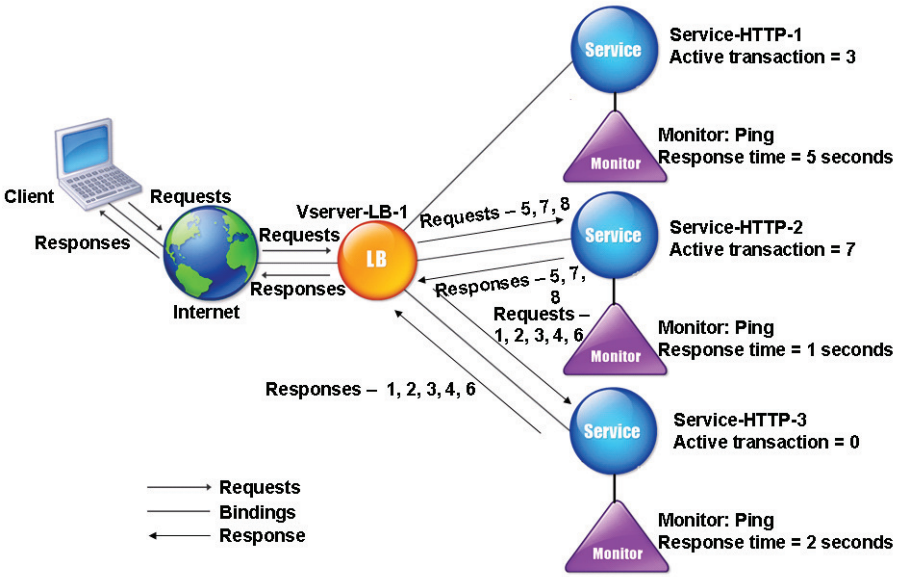
Monitor Response Time Calculations

Monitor	Response time calculation
FTP	Time difference between the sending of the user name and the completion of user authentication.
HTTPS (monitors HTTPS requests)	Time difference is same as the HTTP monitor.
HTTPS-ENV (monitors HTTPS requests)	Time difference is same as the HTTP-ECV monitor.
USER	Time difference between the time when a request is sent to the dispatcher and the time when the dispatcher responds.

The following example shows how the NetScaler selects a service for load balancing by using the least response time method with configured monitors. Consider the following three services:

- Service-HTTP-1 is handling 3 active transactions and the response time is five seconds.
- Service-HTTP-2 is handling 7 active transactions and the response time is one second.
- Service-HTTP-3 is not handling any active transactions and the response time is two seconds.

The following diagram illustrates how the NetScaler uses the least response time method and forward requests to the three services when monitors are configured to calculate the response time.



Least response time mechanism using monitors

The NetScaler selects the service by using the value (*N*) of the following expression:

*N = Number of active transactions * Response time that is determined by the monitor*

The NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because the service is not handling any active transaction.

  Note: The service with no active transaction is selected first.

- Service-HTTP-3 receives the second, third, and fourth requests because the service has the least *N* value.
- Service-HTTP-2 receives the fifth request because the service has the least *N* value.

- Now both Service-HTTP-2 and Service-HTTP-3 have the same *N* value, so the NetScaler performs load balancing in a round robin manner. Therefore, Service-HTTP-3 receives the sixth request.
- Service-HTTP-2 receives the seventh and eighth requests because the service has the least *N* value.

Service-HTTP-1 is not considered for load balancing because it is loaded more (the highest *N* value) as compared to the other two services. However, if Service-HTTP-1 completes the active transactions, the NetScaler considers the service for load balancing.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Least Response Time Method Using Monitors: N

Request received	**Service selected**	**Current N (Number of active transaction) value**	**Remarks**
Request-1	Service-HTTP-3 (*N* = 0)	*N* = 2	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-3 (*N* = 2)	*N* = 4	
Request-3	Service-HTTP-3 (*N* = 4)	*N* = 6	
Request-4	Service-HTTP-3 (*N* = 6)	*N* = 8	
Request-5	Service-HTTP-2 (*N* = 7)	*N* = 8	Service-HTTP-1 and Service-HTTP-3 have the same *N* values.
Request-6	Service-HTTP-3 (*N* = 8)	*N* = 10	
Request-7	Service-HTTP-2 (*N* = 8)	*N* = 9	Service-HTTP-2 has the least *N* value.
Request-8	Service-HTTP-1 (*N* = 9)	*N* = 10	
Service-HTTP-1 is selected for load balancing when it completes the active transactions or when the *N* value of other services (Service-HTTP-2 and Service-HTTP-3) is equal to 15.			

Using Weights with the Least Response Time Method

The NetScaler also performs load balancing by using the number of active transactions, response time, and weights, if different weights are assigned to the services. The NetScaler selects the service by using the value (N_w) of the following expression:

$N_w = (N) * (10000 / weight)$

The following example shows how the NetScaler selects a service for load balancing by using the least response time method when weights are assigned to the services.

In the preceding example, suppose Service-HTTP-1 is assigned a weight of 2, Service-HTTP-2 is assigned a weight of 3, and Service-HTTP-3 is assigned a weight of 4.

The NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because it is not handling any active transaction.

  Note: If services are not handling any active transactions, the NetScaler selects them regardless of the weights assigned to them.

- Service-HTTP-3 receives the second, third, and fourth, requests because the service has the least N_w value.
- Service-HTTP-2 receives the fifth request because the service has the least N_w value.
- Service-HTTP-3 receives the sixth request because the service has the least N_w value.
- Service-HTTP-2 receives the seventh and the eighth requests because the service has the least N_w value.

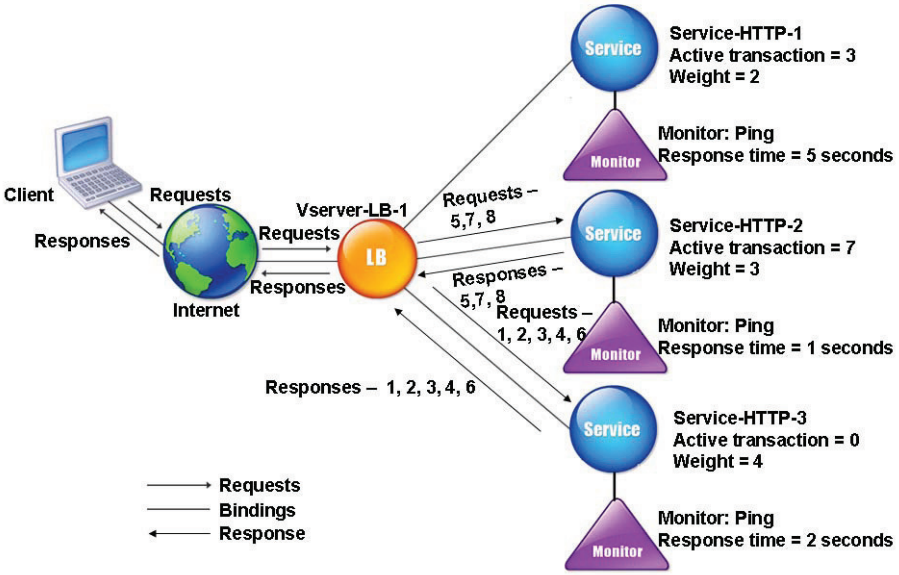
Service-HTTP-1 has the least weight and the highest N_w value. Therefore, the NetScaler does not select it for load balancing.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

Least Response Time Method Using Monitors: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-3 (N_w = 0)	N_w = 5000	Service-HTTP-3 has the least N_w value.
Request-2	Service-HTTP-3 (N_w = 5000)	N_w = 10000	
Request-3	Service-HTTP-3 (N_w = 15000)	N_w = 20000	
Request-4	Service-HTTP-3 (N_w = 20000)	N_w = 25000	
Request-5	Service-HTTP-2 (N_w = 23333.34)	N_w = 26666.67	Service-HTTP-2 has the least N_w value.
Request-6	Service-HTTP-3 (N_w = 25000)	N_w = 30000	Service-HTTP-3 has the least N_w value.
Request-7	Service-HTTP-2 (N_w = 23333.34)	N_w = 26666.67	Service-HTTP-2 has the least N_w value.
Request-8	Service-HTTP-2 (N_w = 25000)	N_w = 30000	Service-HTTP-2 has the least N_w value.
Service-HTTP-1 is selected for load balancing when it completes the active transactions or when the N_w value of other services (Service-HTTP-2 and Service-HTTP-3) is equal to 75000.			

The following diagram illustrates how the NetScaler uses the least response time method when weights are assigned on the services.



Least response time mechanism using monitors when weights are assigned

To configure the least response time method using monitors, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Least Response Time**.

Configuring the Hash Methods

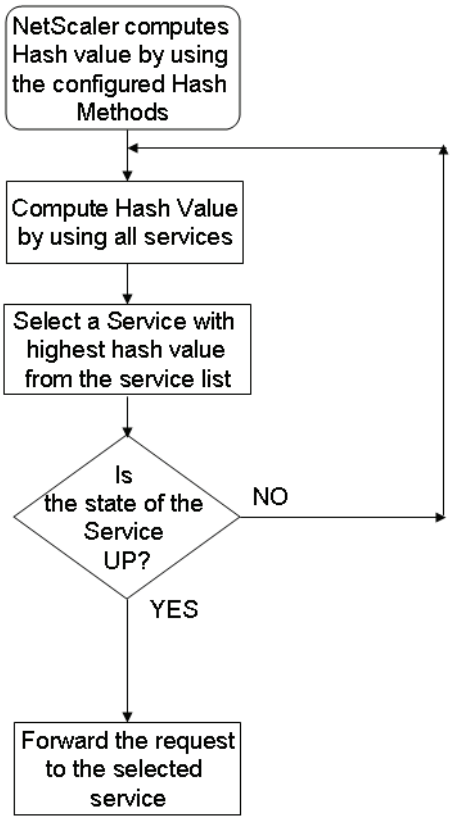
You can use hashing methods in a cache environment where a cache serves a wide range of content from the Internet or origin servers. Caching the requests reduces the request and response latency and ensures better resource utilization (CPU) at the cache. The NetScaler provides the following hashing methods:

- URL hash method
- Domain hash method
- Destination IP hash method
- Source IP hash method
- Source IP Destination IP hash method
- Source IP Source Port hash method
- Call ID hash method
- Token method

These hashing algorithms ensure minimal disruption when the services added and deleted. When the NetScaler is configured to use the hashing methods, the NetScaler lists the services used in the configuration and calculates two hash values by using:

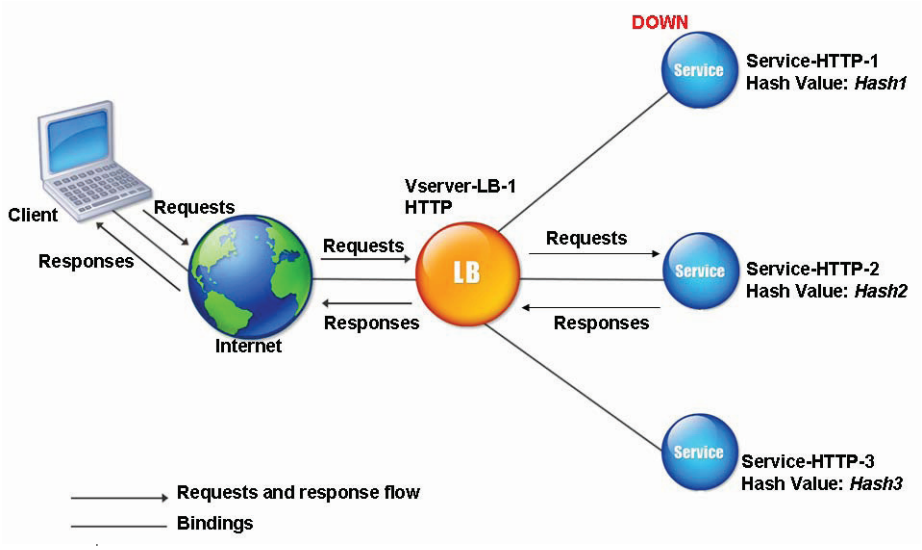
- The service IP address and port.
- The incoming URL, domain name, or source and destination IP address, based on the configured hash method.

The NetScaler then generates a new hash value by using the preceding hash values and forwards the request to the service with highest hash value. To traverse the list of services and compute a hash value for every request, the NetScaler populates a cache after selecting the service which processes the request. The subsequent requests with the same hash value are sent to the same service as shown in the following flow chart.



Sequence of steps in the working of hashing methods

Hashing methods can be applied to IPv4 and IPv6 addresses. To understand how the NetScaler distributes traffic when hashing methods are configured, consider a scenario where three services are bound to a vserver and any hash method is configured. The services are Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3, and the hash value is Hash1. When the configured services are UP, Hash1 is sent to Service-HTTP-1 using the hashing result. If Service-HTTP-1 is down, the NetScaler calculates the hash value for the last log of the number of services. The NetScaler selects the service with the highest hash value, for example Service-HTTP-2 as shown in the following diagram.



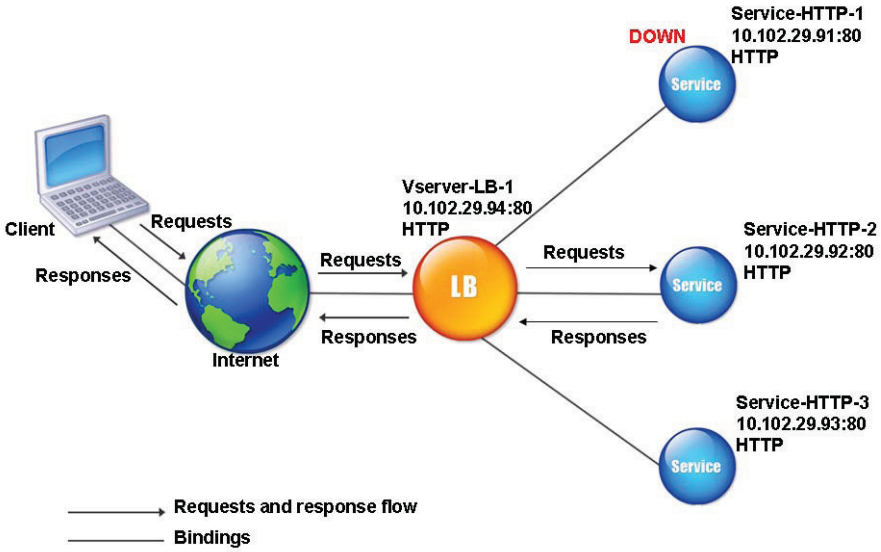
Entity model for hashing methods

Note: If NetScaler fails to select a service by using hash method it uses the least connections method to select the service. It is recommended that when you adjust server pools by removing services, you should adjust the pools during low traffic periods to enable the caches to repopulate without impacting the performance.

Configuring the URL Hash Method

When the NetScaler is configured to use the URL hash method, it selects a service based on the hashed value of an incoming HTTP URL. The NetScaler caches the hashed value of the URL, and subsequent requests that use the same URL make a cache hit and are forwarded to the same service. By default, the NetScaler calculates the hash value based on the first 80 bytes of the URL. You must specify the Hash Length parameter to calculate a different URL value. If the NetScaler cannot accurately parse the incoming request, it uses the round robin method for load balancing. Consider a scenario where three services are bound to a vserver and the URL hash method is configured. The services are

Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3, and the hash value is URL1. When the services are UP, URL1 is sent to Service-HTTP-1 using the hashing result. If Service-HTTP-1 is down, the URL1 is sent to Service-HTTP-2 using the secondary hash result, as shown in the following diagram.



URL hashing

If Service-HTTP-1 and Service-HTTP-2 are down, URL1 is sent to Service-HTTP-3. If the services are then UP, URL1 is sent to the services in the following ways:

- If the Service-HTTP-2 is up, the URL1 is sent to Service-HTTP-2.
- If the Service-HTTP-1 is up, the URL1 is sent to Service-HTTP-1.
- If Service-HTTP-1 and Service-HTTP-2 are up at the same time, URL1 is sent to Service-HTTP-1.

To configure the URL hash method, use the Hash Length parameter as described in the following table.

Hash Length Parameter

Parameter	Specifies
Hash Length (`hashLength`)	The number of bytes that are hashed in the URL or domain name. Valid values are from 1 to 4K bytes. The default is 80 bytes. It must not exceed 4096 bytes.

To configure the URL hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **URL Hash**.

Configuring the Domain Hash Method

When the NetScaler is configured to use the domain hash method, it selects a service based on the hashed value of the domain name in the HTTP request. The domain name is taken either from the incoming URL or from the Host header of the HTTP request. If the domain name appears in both the URL and the Host header, the NetScaler gives preference to the URL.

If you configure domain name hashing and an incoming HTTP request does not contain a domain name, the NetScaler defaults to the round robin method for that request.

The hash value is calculated using the name length or hash length value, whichever is smaller. By default, the NetScaler calculates the hash value from the first 80 bytes of the domain name. To specify a different number of bytes in the domain name when calculating the hash value, use the Hash Length parameter.

To configure the domain hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Domain Hash**.

Configuring the Destination IP Hash Method

When the NetScaler is configured to use the destination IP hash method, it selects a service based on the hashed value of the destination IP address. You can use the subnet mask parameter to mask the destination IP address and then calculate the hash value. When the requests from different networks arrive at the NetScaler, it identifies the requests belonging to a subnet using the subnet mask and forwards the requests to the same server based on the hashed value.

This method is appropriate for use with the cache redirection feature of the NetScaler. For more information about cache redirection, see "Cache Redirection," on page 691. To configure the destination IP hash method, use the Netmask parameter as described in the following table.

Netmask Parameter

Parameter	Specifies
Netmask (`netmask`)	Mask that the destination IP address before calculating the hash value so that all IP addresses belonging to a particular network are directed to the same server.

To configure the destination IP hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Destination IP Hash**.

Configuring the Source IP Hash Method

When the NetScaler is configured to use the source IP hash method, it selects a service based on the hashed value of the client IPv4 or IPv6 address. You must use the optional subnet mask parameter to direct the requests from source IP addresses that belong to a particular network, to one server. To configure the source IP hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Source IP Hash**.

Configuring the Source IP Destination IP Hash Method

When the NetScaler is configured to use the source IP destination IP hash method, it selects a service based on the hashed value of the source and destination IP addresses (either IPv4 or IPv6). Hashing is symmetric, so it yields the same value if the source and destination IP addresses are reversed. This ensures that all packets flowing from a particular client to the same destination are directed to the same server. To configure the source IP destination IP hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Source IP Destination IP Hash**.

Configuring the Source IP Source Port Hash Method

When the NetScaler is configured to use the source IP source port hash method, it selects the server based on the hash value of the source IP (either IPv4 or IPv6) and source port of the incoming request. This ensures that all packets on a particular connection are directed to the same server. This method is used in the connection Mirroring and firewall load balancing. For more information about connection Mirroring, see the section "Configuring Stateful Connection Failover," on page 123.

To configure the source IP source port hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Source IP Source Port Hash**.

Configuring the Call ID Hash Method

When the NetScaler is configured to use the call ID hash method, it selects the service based on the hash value of the call ID in the SIP header, so that a particular SIP session is directed to the same proxy server. This method is applicable for SIP load balancing. For more information about SIP load balancing, see the section "Monitoring SIP Services," on page 176. To configure the call ID hash method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Call ID Hash**.

Configuring the Least Bandwidth Method

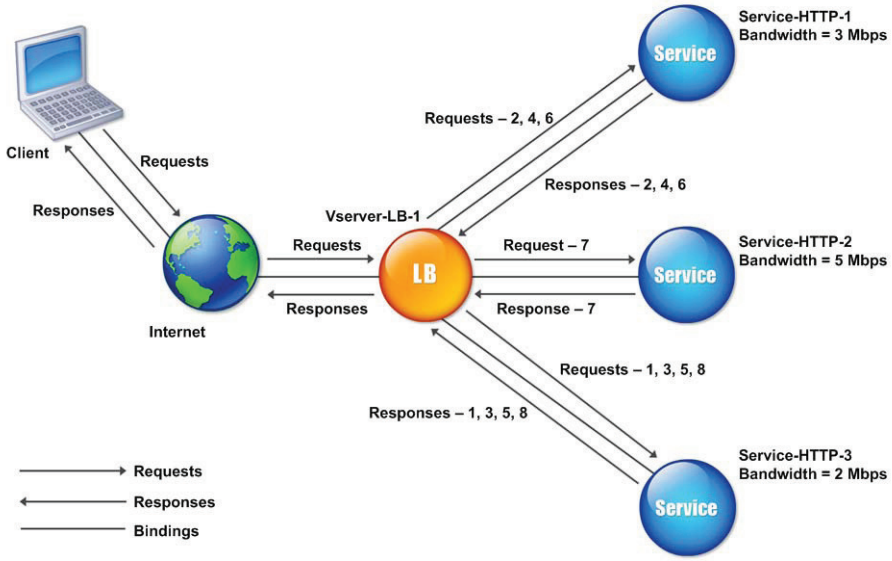
When the NetScaler is configured to use the least bandwidth method, it selects the service that is currently serving the least amount of traffic, measured in megabits per seconds (Mbps). The following example shows how the NetScaler selects a service for load balancing by using the least bandwidth method.

Example:

Consider three services, Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3.

- Service-HTTP-1 has 3 Mbps bandwidth.
- Service-HTTP-2 has 5 Mbps bandwidth.
- Service-HTTP-3 has 2 Mbps bandwidth.

The following diagram illustrates how the NetScaler uses the least bandwidth method to forward requests to the three services.



Least bandwidth mechanism

The NetScaler selects the service by using the bandwidth value (*N*) which is the sum of the number of bytes transmitted and received per 14 second. If each request requires 1 Mbps bandwidth, the NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because the service has the least *N* value.

- Now Service-HTTP-1 and Service-HTTP-3 have same *N* value and the NetScaler performs load balancing in a round robin manner. Service-HTTP-1 receives the second request, Service-HTTP-3 receives the third request, Service-HTTP-1 receives the fourth request, Service-HTTP-3 receives the fifth request, and Service-HTTP-1 receives the sixth request.
- Now Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have same *N* value and the NetScaler performs load balancing in a round robin manner. So, Service-HTTP-2 receives the seventh request and Service-HTTP-3 receives the eighth request.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Examples of Least Bandwidth Method: N

Request received	**Service selected**	Current ***N*** ***(Number of active transaction)*** value	**Remarks**
Request-1	Service-HTTP-3 (*N* = 2)	*N* = 3	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-1 (*N* = 3)	*N* = 4	Service-HTTP-1 and Service-HTTP-3 have the same *N* values.
Request-3	Service-HTTP-3 (*N* = 3)	*N* = 4	
Request-4	Service-HTTP-1 (*N* = 4)	*N* = 5	
Request-5	Service-HTTP-3 (*N* = 4)	*N* = 5	
Request-6	Service-HTTP-1 (*N* = 5)	*N* = 6	Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have the same *N* values.
Request-7	Service-HTTP-2 (*N* = 5)	*N* = 6	
Request-8	Service-HTTP-3 (*N* = 5)	*N* = 6	

Note: If you enable the RTSP NAT option on the vserver, the NetScaler uses the number of data and control bytes exchanged to determine the bandwidth usage for RTSP services. For more information about RTSP NAT option, see "Managing RTSP Connections," on page 140. RTSP is not supported in NetScaler 9.1 nCore.

Using Weights with the Least Bandwidth Method

The NetScaler also performs load balancing by using the bandwidth and weights if different weights are assigned to the services. The NetScaler selects the service by using the value (N_w) of the following expression:

$N_w = (N) * (10000 / weight)$

The following example shows how the NetScaler selects a service for load balancing by using the least bandwidth method when weights are assigned on the services.

Example:

In the preceding example, suppose Service-HTTP-1 is assigned a weight of 2, Service-HTTP-2 is assigned a weight of 3, and Service-HTTP-3 is assigned a weight of 4.

The NetScaler delivers the requests as follows:

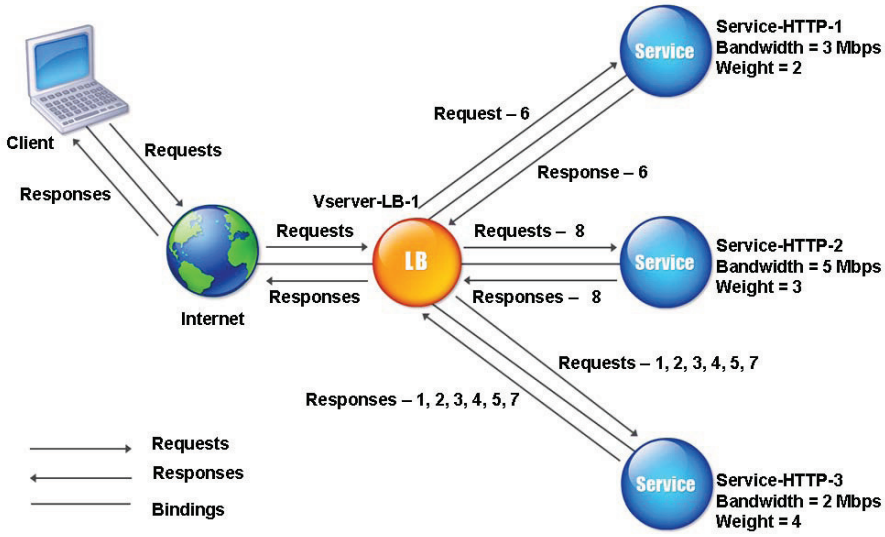
- Service-HTTP-3 receives the first second, third, fourth, and fifth requests because the service has the least N_w value.
- Service-HTTP-1 receives the sixth request because the service has the least N_w value.
- Service-HTTP-3 receives the seventh request because the service has the least N_w value.
- Service-HTTP-2 receives the eighth request because the service has the least N_w value.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

Examples of Least Bandwidth Method: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-3 (N_w = 5000)	N_w = 5000	Service-HTTP-3 has the least N_w value.
Request-2	Service-HTTP-3 (N_w = 5000)	N_w = 7500	
Request-3	Service-HTTP-3 (N_w = 7500)	N_w = 10000	
Request-4	Service-HTTP-3 (N_w = 10000)	N_w = 12500	
Request-5	Service-HTTP-3 (N_w = 12500)	N_w = 15000	
Request-6	Service-HTTP-1 (N_w = 15000)	N_w = 20000	Service-HTTP-1 and Service-HTTP-3 have the same N_w value.
Request-7	Service-HTTP-3 (N_w = 15000)	N_w = 17500	
Request-8	Service-HTTP-2 (N_w = 16666.67)	N_w = 20000	Service-HTTP-2 has the least N_w value.

The following diagram illustrates how the NetScaler uses the least bandwidth method when weights are assigned on the services.



Least bandwidth mechanism when weights are assigned

To configure the least bandwidth method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Least Bandwidth**.

Configuring the Least Packets Method

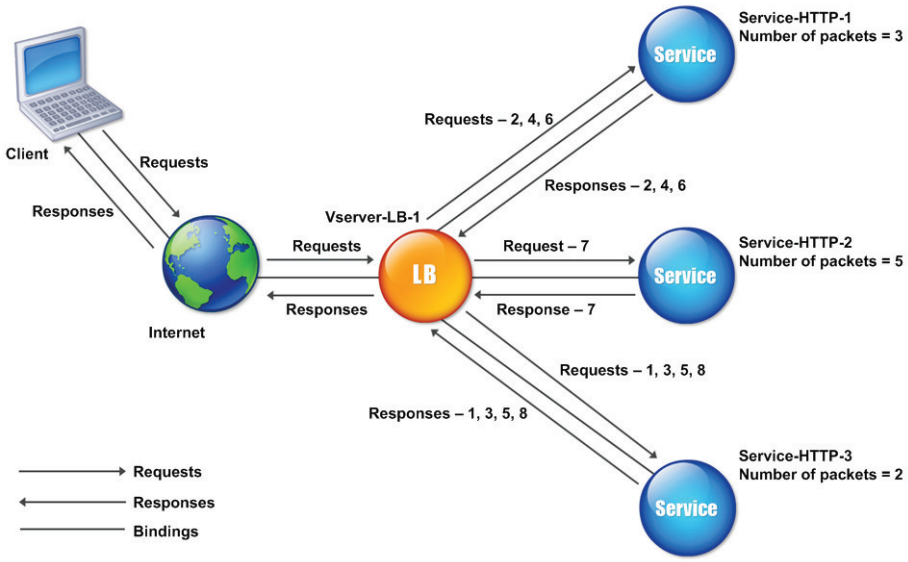
When the NetScaler is configured to use the least packets method, it selects the service that is currently serving the least packets in the last 14 seconds second. The following example shows how the NetScaler selects a service for load balancing by using the least packets method.

Example:

Consider three services, Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3.

- Service-HTTP-1 is handling three packets in last 14 seconds.
- Service-HTTP-2 is handling five packets in last 14 seconds.
- Service-HTTP-3 is handling two packets in last 14 seconds.

The following diagram illustrates how the NetScaler uses the least packets method and forward requests to the three services.



Least packets mechanism

The NetScaler selects the service by using the number of packets (*N*) which is the sum of the number of packets transmitted and received in last 14 seconds.

The NetScaler delivers the requests as follows:

- Service-HTTP-3 receives the first request because the service has the least *N* value.
- Now Service-HTTP-1 and Service-HTTP-3 have same *N* value and the NetScaler performs load balancing in a round robin manner. Service-HTTP-1 receives the second request, Service-HTTP-3 receives the third request, Service-HTTP-1 receives the fourth request, Service-HTTP-3 receives the fifth request, Service-HTTP-1 receives the sixth request.
- Now Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have same *N* value and the NetScaler performs load balancing in a round robin manner. So, Service-HTTP-2 receives the seventh request, and Service-HTTP-3 receives the eighth request.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Examples of Least Packets Method: N

Request received	**Service selected**	Current ***N*** ***(Number of active transaction)*** value	**Remarks**
Request-1	Service-HTTP-3 (*N* = 2)	*N* = 3	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-1 (*N* = 3)	*N* = 4	Service-HTTP-1 and Service-HTTP-3 have the same *N* values.
Request-3	Service-HTTP-3 (*N* = 3)	*N* = 4	
Request-4	Service-HTTP-1 (*N* = 4)	*N* = 5	
Request-5	Service-HTTP-3 (*N* = 4)	*N* = 5	
Request-6	Service-HTTP-1 (*N* = 5)	*N* = 6	Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have the same *N* values.
Request-7	Service-HTTP-2 (*N* = 5)	*N* = 6	
Request-8	Service-HTTP-3 (*N* = 5)	*N* = 6	

Note: If you enable the RTSP NAT option on the vserver, the NetScaler uses the number of data and control packets to calculate the number of packets for RTSP services. For more information about RTSP NAT option, see "Managing RTSP Connections," on page 140. RTSP is not supported in NetScaler 9.1 nCore.

Using Weights with the Least Packets Method

The NetScaler also performs load balancing by using the number of packets and weights if different weights are assigned to the services. The NetScaler selects the service by using the value (N_w) of the following expression:

$N_w = (N) * (10000 / weight)$

The following example shows how a NetScaler selects a service for load balancing by using the least packets method when weights are assigned on the services.

Example:

In the preceding example, suppose Service-HTTP-1 is assigned a weight of 2, Service-HTTP-2 is assigned a weight of 3, and Service-HTTP-3 is assigned a weight of 4.

The NetScaler delivers the requests as follows:

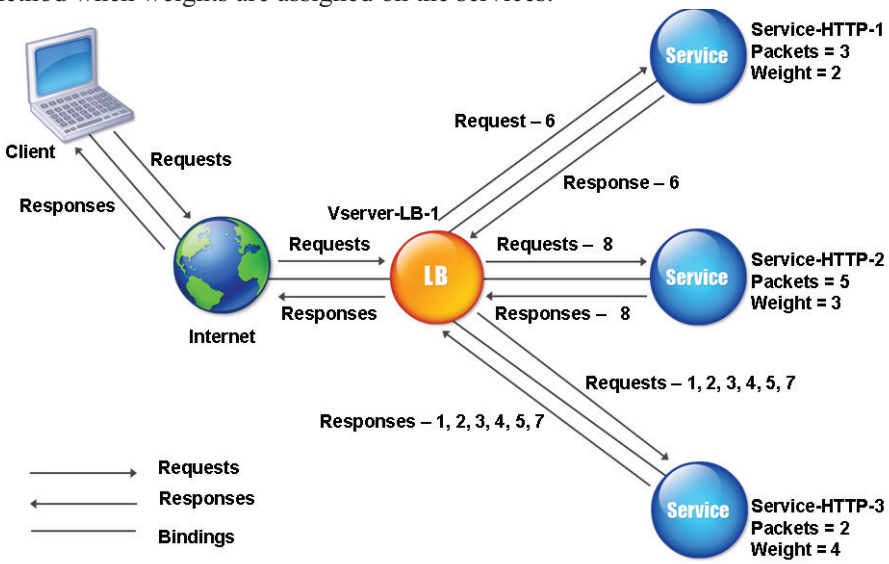
- Service-HTTP-3 receives the first second, third, fourth, and fifth requests because the service has the least N_w value.
- Service-HTTP-1 receives the sixth request because the service has the least N_w value.
- Service-HTTP-3 receives the seventh request because the service has the least N_w value.
- Service-HTTP-2 receives the eighth request because the service has the least N_w value.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

Examples of Least Bandwidth Method: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-3 (N_w = 5000)	N_w = 5000	Service-HTTP-3 has the least N_w value.
Request-2	Service-HTTP-3 (N_w = 5000)	N_w = 7500	
Request-3	Service-HTTP-3 (N_w = 7500)	N_w = 10000	
Request-4	Service-HTTP-3 (N_w = 10000)	N_w = 12500	
Request-5	Service-HTTP-3 (N_w = 12500)	N_w = 15000	
Request-6	Service-HTTP-1 (N_w = 15000)	N_w = 20000	Service-HTTP-1 and Service-HTTP-3 have the same N_w value.
Request-7	Service-HTTP-3 (N_w = 15000)	N_w = 17500	
Request-8	Service-HTTP-2 (N_w = 16666.67)	N_w = 20000	Service-HTTP-2 has the least N_w value.

The following diagram illustrates how the NetScaler uses the least packets method when weights are assigned on the services.



Least packets mechanism when weights are assigned

To configure the least packets method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Least Packets**.

Configuring the Token Method

When the NetScaler is configured to use the token method, it selects a service based on the value of a token extracted from the client request. You can configure the location and size of the token. For subsequent requests with the same token, the NetScaler chooses the same server that handled the initial request.

This method is content aware. This means that it operates differently for the TCP, HTTP, and HTTPS service types. For HTTP or HTTPS services, the token is found in the HTTP headers or the URL or the BODY using the configured expressions.

Rule Parameter

Parameter	Specifies
Rule (`rule`)	String value. The string can be an existing rule name, or it can be an inline expression with a maximum of 256 characters. The default string value is none. The maximum length of the string value is 14999.

Expressions can be classic or advanced. Advanced expressions do not need to have rules. For more information about expressions, see the *Citrix NetScaler Policy Configuration and Reference Guide*.

For example, consider you are load balancing a set of servers that contain Web content, and you want to configure the NetScaler to search for the token inside a URL query in each request. The NetScaler searches the token inside the URL query after matching the string token.

For HTTP services, the NetScaler searches for the configured token in the first 24 kilobytes (KB) of the TCP payload. For non-HTTP (TCP, SSL, and SSL_TCP) services, the NetScaler searches for the configured token in the first 16 packets if the total size of the 16 packets is less than 24 KB. But, if the total size of the 16 packets is greater than 24 KB, the NetScaler searches for the token in the first 24 KB of payload.

To configure the location and size of the token, use the parameters as described in the following table.

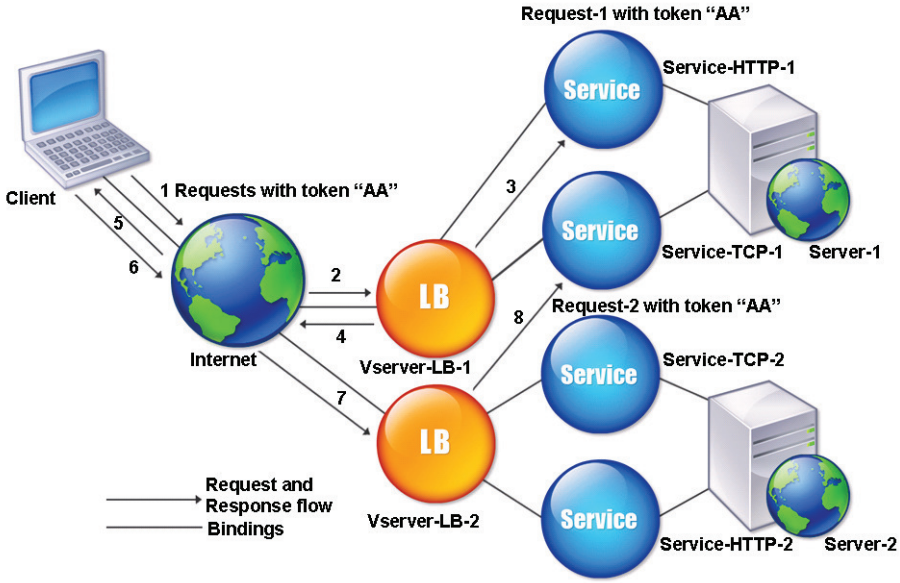
Token Location and Size Parameters

Parameters	Specifies
Data Length (`dataLength`)	Length of the token in bytes. This parameter is applicable to TCP virtual servers when the Token method is selected. The minimum value is 0, and the maximum value is 100.
Data Offset (`dataOffset`)	Offset of the data to be taken as a token. This parameter is applicable to the TCP type virtual servers when the Token load balancing method is used and must be within the first 24 KB of the client TCP data. The minimum value is 0, and the maximum value is 25400.

You can use this load balancing method across vservers of different types to make sure that requests presenting the same token are directed to the services on the same servers, regardless of the protocol used.

For example, consider server-1 has two services, Service-HTTP-1 and Service-TCP-1, and server-2 has two services, Service-HTTP-2 and Service-TCP-2. The TCP services are bound to Vserver-LB-2, and the HTTP services are bound to Vserver-LB-1.

A request sent to Vserver-LB-1 with the token “AA” selects the service Service-HTTP-1 (bound to server-1) to process the request. A different request sent to Vserver-LB-2 with the same token “AA” directs this request to the service Service-TCP-1, as shown in the following diagram.



Working of token method

To configure the token method, perform the steps described in the section “Changing the Load Balancing Algorithm,” on page 55. Under **LB Method**, select **Token**. You must configure a rule to configure a token.

To configure a rule using the configuration utility

1. In the navigation pane, expand **Load Balancing**, and then click **Virtual Servers**.
2. In the details pane, a vserver for which you want to configure a rule (for example, **Vserver-LB-1**), and then click **Open**.
3. In the **Configure Virtual Server (Load Balancing)** dialog box, click the **Method and Persistence** tab and under **LB Method**, select **Token**.
4. Click **Configure** next to the **Rule** text box.
5. In the **Create Expression** dialog box, select **Classic Syntax** or **Advanced Syntax**.
6. Under **Expression**, click **Add**.

7. In the **Add Expression** dialog box, enter an expression. For more information about expressions, see the *Citrix NetScaler Policy Configuration and Reference Guide*.

   For example, if you are configuring a classic expression, you can select an **Expression Type** of **General**, a **Flow Type** of **REQ**, a **Protocol** of **HTTP**, a **Qualifier** of **URLQUERY,** an **Operator** of **CONTAINS**, and in the **Value** text box, type **AA**.

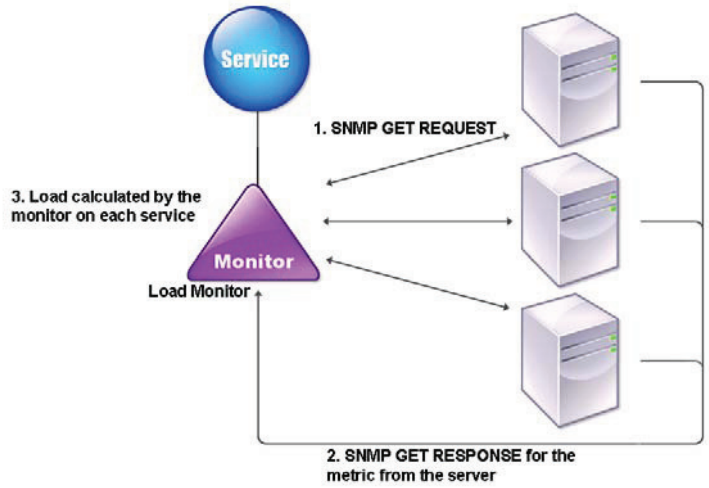
8. Click **OK** and click **Close**.

9. In the **Create Expression** dialog box, click **Create**. The expression you created appears in the **Rule** text box.

10. Click **OK**.

Configuring the Custom Load Method

Generally, the NetScaler selects a service that is not handling any active transaction. If the services are handling active transactions, the NetScaler selects a service based on its load. A special type of monitor, known as a load monitor, calculates the load on each service in the network. The load monitors do not mark the state of a service; however, they take the service out of the load balancing decision. Custom load balancing is performed on server parameters such as CPU usage, memory, and response time. For more information about load monitors, see "Configuring Load Monitors," on page 206. The following diagram illustrates how a load monitor operates.



Working of load monitors

The load monitor uses an a Simple Network Management Protocol (SNMP) probe to calculate the load on the service. To accomplish this, the monitor sends an SNMP GET request to the server. This request contains one or more object IDs (OID). The server then sends back an SNMP GET response with metrics corresponding to the SNMP OIDs. The monitor calculates the load on each service based on the metrics in the response message and parameters such as pre-configured threshold and weight as described later in the chapter.

The load monitor calculates the load on a service using the following parameters:

- Metrics values retrieved through SNMP probes that exist as tables in the NetScaler.
- Threshold value set for each metric.
- Weight assigned to each metric.

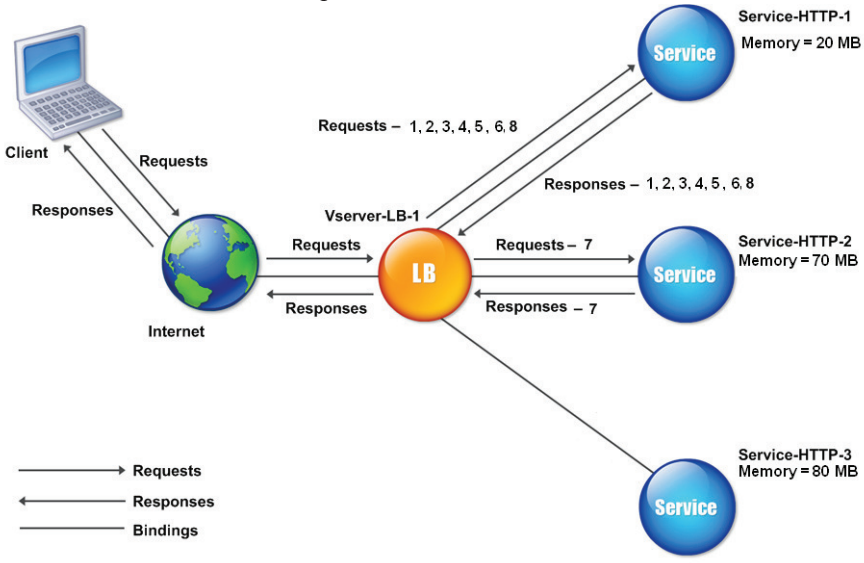
The following example shows how the NetScaler selects a service for load balancing by using the custom load method.

Example:

Consider three services, Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3.

- Service-HTTP-1 is using 20 megabytes (MB) of memory.
- Service-HTTP-2 is using 70 MB of memory.
- Service-HTTP-3 is using 80 MB of memory.

The servers can export metrics such as CPU and memory usage. The load monitor sends an SNMP GET request containing the OIDs 1.3.6.1.4.1.5951.4.1.1.41.1.5, 1.3.6.1.4.1.5951.4.1.1.41.1.4, and 1.3.6.1.4.1.5951.4.1.1.41.1.3 to the servers. The three services respond to the request. The NetScaler compares the exported metrics to select Service-HTTP-1 because it has more memory for processing requests. The following diagram illustrates how the NetScaler uses the custom load method and forwards requests to the three services.



Custom load mechanism

The NetScaler selects the service by using the load (*N*). If each request uses 10 MB memory, the NetScaler delivers the requests as follows:

- Service-HTTP-1 receives the first, second, third, fourth, and fifth requests because the service has the least *N* value.
- Now, Service-HTTP-1 and Service-HTTP-2 have same load and the NetScaler selects the service in round robin manner. Therefore, Service-HTTP-2 receives the sixth request and Service-HTTP-1 receives the seventh request.
- Now, Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have same load and the NetScaler selects the service in round robin manner. Therefore, Service-HTTP-1 receives the eighth request.

The manner in which a service receives requests based on the *N* value is summarized in the following table.

Custom Load Balancing Method: N

Request received	**Service selected**	Current ***N*** ***(Number of active transaction)*** value	**Remarks**
Request-1	Service-HTTP-1 (*N* = 20)	*N* = 30	Service-HTTP-3 has the least *N* value.
Request-2	Service-HTTP-1 (*N* = 30)	*N* = 40	
Request-3	Service-HTTP-1 (*N* = 40)	*N* = 50	
Request-4	Service-HTTP-1 (*N* = 50)	*N* = 60	
Request-5	Service-HTTP-1 (*N* = 60)	*N* = 70	
Request-6	Service-HTTP-1 (*N* = 70)	*N* = 80	Service-HTTP-2 and Service-HTTP-3 have the same *N* values.
Request-7	Service-HTTP-2 (*N* = 70)	*N* = 80	
Request-8	Service-HTTP-1 (*N* = 80)	*N* = 90	Service-HTTP-1, Service-HTTP-2, and Service-HTTP-3 have the same *N* values.

Using Weights with the Custom Load Method

The NetScaler also performs load balancing by using the load on services, and weights if different weights are assigned to the services, weights are also used for load balancing. The NetScaler selects the service by using the value (N_w) of the following expression:

$N_w = (N) * (10000 / weight)$

The following example shows how the NetScaler selects a service for load balancing by using the custom load method when weights are assigned on the services.

Example:

In the preceding example, suppose Service-HTTP-1 is assigned a weight of 4, Service-HTTP-2 is assigned a weight of 3, and Service-HTTP-3 is assigned a weight of 2. If each request uses 10 MB memory, the NetScaler delivers the requests as follows:

- Service-HTTP-1 receives the first, second, third, fourth, fifth, sixth, seventh, and eighth requests because the service has the least N_w value.
- Service-HTTP-2 receives the ninth request because the service has the least N_w value.

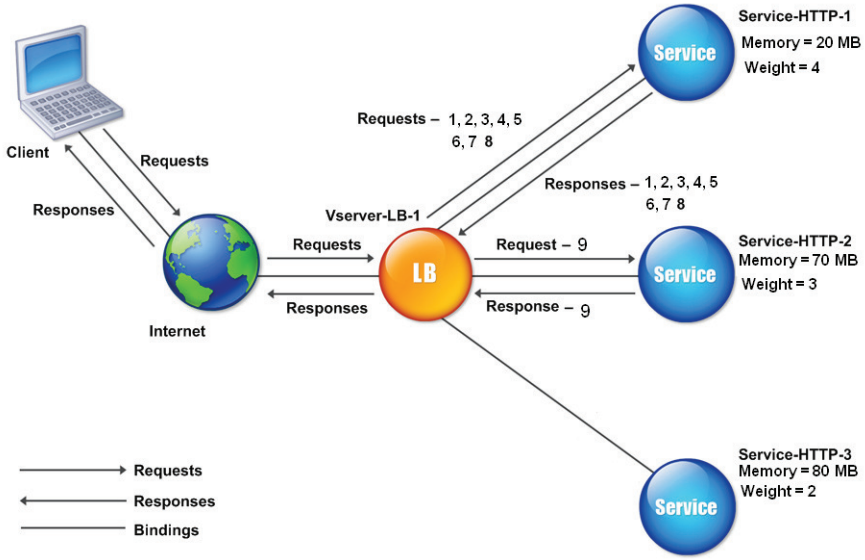
Service-HTTP-3 has the highest N_w value and is not considered for load balancing.

The manner in which a service receives requests based on the N_w value is summarized in the following table.

Custom Load Balancing Method: N_w

Request received	**Service selected**	**Current N_w *(Number of active transactions) * (10000 / weight)* value**	**Remarks**
Request-1	Service-HTTP-1 (N_w = 50000)	N_w = 75000	Service-HTTP-1 has the least N_w value.
Request-2	Service-HTTP-1 (N_w = 5000)	N_w = 100000	
Request-3	Service-HTTP-1 (N_w = 15000)	N_w = 125000	
Request-4	Service-HTTP-1 (N_w = 20000)	N_w = 150000	
Request-5	Service-HTTP-1 (N_w = 23333.34)	N_w = 175000	
Request-6	Service-HTTP-1 (N_w = 25000)	N_w = 200000	
Request-7	Service-HTTP-1 (N_w = 23333.34)	N_w = 225000	
Request-8	Service-HTTP-1 (N_w = 25000)	N_w = 250000	
Request-9	Service-HTTP-2 (N_w = 233333.34)	N_w = 266666.67	Service-HTTP-2 has the least N_w value.
Service-HTTP-1 is selected for load balancing when it completes the active transactions or when the N_w value of other services (Service-HTTP-2 and Service-HTTP-3) is equal to 400000.			

The following diagram illustrates how the NetScaler uses the custom load method when weights are assigned on the services.



Custom Load mechanism when weights are assigned

To configure the custom load method, perform the steps described in the section "Changing the Load Balancing Algorithm," on page 55. Under **LB Method**, select **Custom Load**.

Configuring Persistent Connections Between Clients and Servers

The NetScaler initially selects a server by using a load balancing method. With persistence configured, enabling the NetScaler to send any subsequent client requests to the selected server, the server can access state information for that client.

If persistence is configured, it overrides the load balancing methods once the server has been selected. If the configured persistence applies to a service that is down, the NetScaler uses the load balancing methods to select a new service, and the new service becomes persistent for subsequent requests from the client. If the state selected service is out of service, it continues to serve the outstanding requests, but it does not allow new requests or connections. After the shutdown period elapses, no new requests or connections are directed to the service and the existing connections are closed.