Load Balancing Methods for DNS Traffic Control

You can define the following load balancing methods:

<table>
<thead>
<tr>
<th>Load Balancing Method</th>
<th>LBDN</th>
<th>Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Available</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Global Availability</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Round Robin</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Ratio: Fixed</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Ratio: Dynamic (Round Trip Delay)</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Ratio: Dynamic (SNMP)</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Topology</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Based on the load balancing method defined for an LBDN, the DNS Traffic Control selects an available pool. Based on the method selected for a pool, it selects an available server.

The following is a description of the load balancing methods with examples for pools or LBDNs:

- **Using the All Available method**, the appliance responds to the query with all the available servers in the DTC pool for the appropriate record type. The responses are returned in the same order in which the servers are listed in the DTC pool, eliminating the unavailable servers. Consider the following example for all available records load balancing method with an LBDN x.abc.com:

  - Pool= Pool_1; load balancing method= all available records; health check= HTTPS 10.10.1.1; Availability = up
  - 2001:db8:a22:a00::29; Availability = up
  - 10.10.2.2; Availability = down
  - 2001:db8:a22:a00::32; Availability = up
  - 10.10.3.3; Availability = up

  In this example, the appliance responds with 10.10.1.1, 10.10.3.3 for each A record query. For each AAAA record query, NIOS responds with 2001:db8:a22:a00::29, 2001:db8:a22:a00::32. The unavailable servers are eliminated. Note that the system considers only the order of the servers in the DTC pool and ignores the weight of available servers.

- **Using the Global Availability method**, the appliance creates the response to the query, so that the clients are directed to the first available pool and server. Consider the following example for global availability load balancing method with an LBDN x.abc.com:

  - Pool= Pool_1; load balancing method= global availability; health check= HTTPS 10.10.1.1
  - 10.10.2.2
  - 10.10.3.3
  - Pool= Pool_2; load balancing method = global availability; health check = HTTPS 10.10.4.4
  - 10.10.5.5
  - 10.10.5.5

  In this example, the appliance always responds with 10.10.1.1 (A record) for all x.abc.com queries assuming that all servers are available. The DNS Traffic Control LBDN determines which pool to use based on the health check response and adjusts the response accordingly. The appliance responds with 10.10.4.4 from Pool_2 if health check for all servers associated with Pool_1 fails.

- **Using the Ratio: Fixed method**, NIOS adjusts the response to the query so that the clients are directed to servers in a pool or among pools. When multiple pools or servers are configured, the appliance uses the weighted round robin method, a load-balancing pattern in which requests are distributed among several pools or servers based on a weight assigned to each pool or server. Note that the system considers the weight of available servers only. Consider the following example for ratio load balancing method with an LBDN x.abc.com, load balancing method = Ratio and two linked pools: Pool_1 with weight = 70 and Pool_2 with weight = 30.

  Pool = Pool_1; load balancing method = ratio; health check = HTTPS 10.10.1.1; Weight = 50
  10.10.2.2; Weight = 2
  10.10.3.3; Weight = 25

  Pool = Pool_2; load balancing method = ratio; health check = HTTPS 10.10.4.4; Weight = 30
  10.10.5.5; Weight = 2
In this example, the appliance responds 70% of the time with a server associated with Pool_1. Within Pool_1, it responds with 10.10.1.1 address 50% of the time.

- Using the **Ratio: Dynamic** method, the appliance weights the DTC servers dynamically based on round trip delay or SNMP health monitor data. You can use one of the following options:
  - **Round trip delay**: Based on the round trip delay from the DTC member that received a client’s DNS request, the system sends clients to the server with the minimal latency time, i.e. the closest one. You need a pre-configured health monitor for this load balancing method.
    
    For example:
    
    - Server A latency = 25 ms
    - Server C latency = 18 ms
    - Server D latency = 50 ms
    
    In this case, the traffic distribution is as follows:
    
    - Server A = 0%
    - Server C = 100%
    - Server D = 0%

  - **SNMP**: Based on data from the SNMP monitor associated to the server, for example, CPU or memory utilization, the system sends clients to the server with the lowest load. For this load balancing method, you need a pre-configured SNMP health monitor with a required metric to be tracked. The metric is set through an object identifier (OID) in the monitor properties. This method supports only OIDs for which the system can return an integer value.
    
    The value of the monitored metric defines how the traffic is directed. By default, the servers with the highest metric values receive the client requests. There may be cases when your selected metric reflects server availability in the opposite way, that is, the lowest metric values indicate available servers. For such cases, you can invert the value of the OID, that is, of the monitored metric, and have the traffic directed to the lowest-rated servers.
    
    You can select to weight servers by either priority or ratio. In case of priority, traffic is directed towards the servers that report the best metric values, other servers being bypassed. In case of ratio, traffic is distributed across all servers based on the values of the monitored metric for each of them. If a health check for a server is failed, the server is excluded from the load balancing.
    
    Consider the following example where the CPU utilization metric is used for server monitoring:
    
    - Server A CPU utilization = 90%
    - Server C CPU utilization = 50%
    - Server D CPU utilization = 10%
    
    With normal dynamic weights, the distribution is as follows:
    
    - Server A 60% (calculated as 90 / (10 + 50 +90) = 90/150 = 0.6)
    - Server C 33% (calculated as 50 / (10 + 50 +90) = 50/150 ~ 0.33)
    - Server D 7% (calculated as 10 / (10 + 50 +90) = 10/150 ~ 0.066)
    
    This means that the most loaded server will receive most requests than the less loaded one. For this case, the metric should be inverted to reflect server availability appropriately:
    
    - Server A 8% (calculated as 1/90 / (1/90 + 1/50 + 1/10) = 0.011/0.131 ~ 0.08)
    - Server C 15% (calculated as /50 / (1/90 + 1/50 + 1/10) = 0.02/0.131 ~ 0.15)
    - Server D 77% (calculated as 1/10 / (1/90 + 1/50 + 1/10) = 0.1/0.131 ~ 0.77)

  **NOTE**: You can see traffic distribution percentage across members in pools and servers based on selected load balancing methods in the visualization panel. For information, see Visualization for DNS Traffic Control Objects on page 1087.

- Using the **Round Robin** method, the appliance returns servers sequentially and cyclically. Consider the following example for round robin load balancing method with an LBDN x.abc.com:
    
    - Pool = Pool_1; load balancing method = Round Robin; health check = HTTPS 10.10.1.1;
    - 10.10.2.2;
    - 10.10.3.3;

    In this example, NIOS responds with a server associated with Pool_1. Within Pool_1, it responds sequentially:
    
    Time 1: Response = 10.10.1.1
    Time 2: Response = 10.10.2.2
    Time 3: Response = 10.10.3.3
    Time 4: Response = 10.10.1.1
    Time 5: Response = 10.10.2.2
    Time 6: Response = 10.10.3.3

- Using the **Topology** method, the appliance applies a predefined geographic mapping method and evaluates user-defined geography, subnet, or extensible attribute rules sequentially. Geographical locations for the geography rules are provided through an external topology database. The appliance supports the MaxMind GeoIP2 City or Country database and the MaxMind GeoLite2 City or Country database. For more information, see the following section, Configuring Topology Rules and Rulesets.