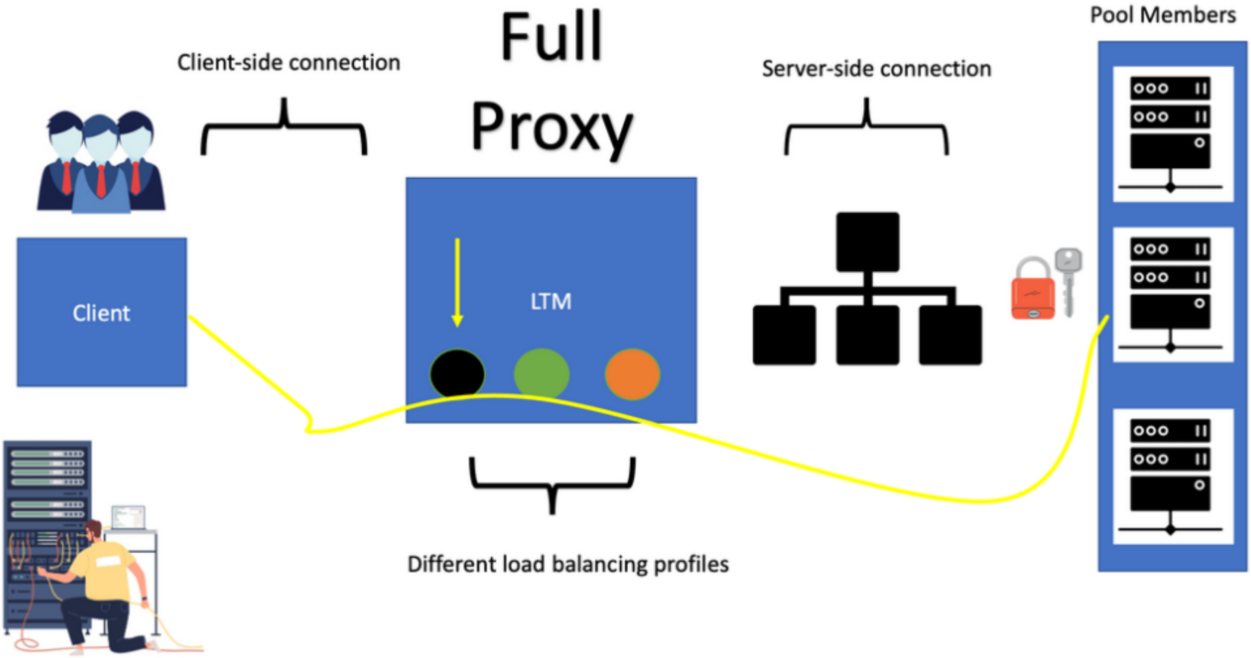
NI THE VISUAL AGE



Load Balancing and Scale-Out Architectures

February 26, 2015 by Matt Conran with no comment Blog requests. This helps to improve overall system performance and reliability. Load balancers can balance traffic between multiple web servers, application servers, and databases.

They can also be used to balance traffic between different geographic locations. Load balancers are typically configured to use round-robin, least connection, or source-IP affinity algorithms to determine how to distribute traffic. They can also be configured to use health checks to ensure that only healthy servers receive traffic. By distributing the load across multiple servers, the load balancer helps reduce the risk of server failure and improve overall system performance.

- **Load Balancers and the OSI Model**

Load balancers operate at different **Open Systems Interconnection (OSI) Layers** from one data center to another; joint operation is between Layer 4 and Layer 7. The load balance function becomes the virtual representation of the application. Internal applications are represented by a virtual IP address (VIP). VIP acts as a front-end serving external clients' requests. Data centers hosts-unique applications with different requirements. Therefore load balancing and scalability will vary depending on what applications are housed.

For example, every application is unique concerning the number of sockets, TCP connections (short-lived or long-lived), idle time-out, and activities in each session regarding packets per second. Therefore, understanding the application structure and protocols is one of the most critical elements in determining how to scale load balancer and design an effective load-balancing solution. **Let us examine Direct Server Return as the first example.**

Preliminary Information: Useful Links to Relevant Content

Before you proceed, you may find the following post helpful:

1. Auto Scaling Observability

6. GTM Load Balancer

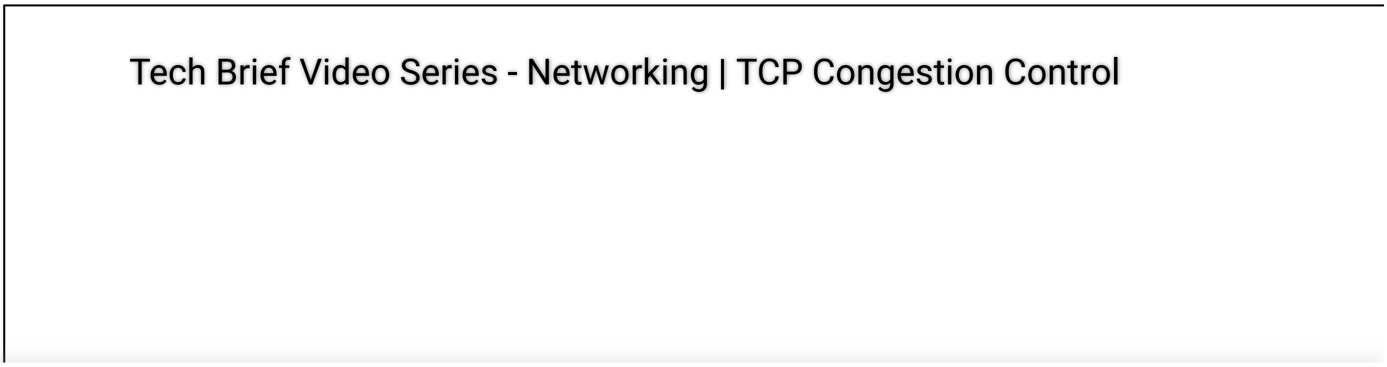
Key Load Balancer Scaling Discussion Points:

Load Balancers

1. Introduction to load balancer scaling and what is involved.
2. Highlighting the details of how to scale load balancer.
3. Critical points on load balancer scalability: Up and Out.
4. Technical details on the different load balancer scaling types: Network and Application.
5. Technical details on Layer 2 and 3 load balancing.

- **A key point: Video on TCP congestion control**

In this video tutorial, we will address the concept of TCP Congestion control. The discrepancy and uneven bandwidth allocation for flow boil down to the natural behavior of how TCP reacts and interacts with insufficient packet buffers and the resulting packet drops.



Tech Brief Video Series - Networking | TCP Congestion Control

A Key Point: Knowledge Check

- **A key point: Back to basics with load balancers and load balancing**

How is this like load balancing in the computing world? It all comes down to having finite resources and attempting to make the best potential use of them. For example, you may have the goal of making your websites fast; to do that, you must route your requests to the machines best capable of handling them. To get around this, you need more resources.

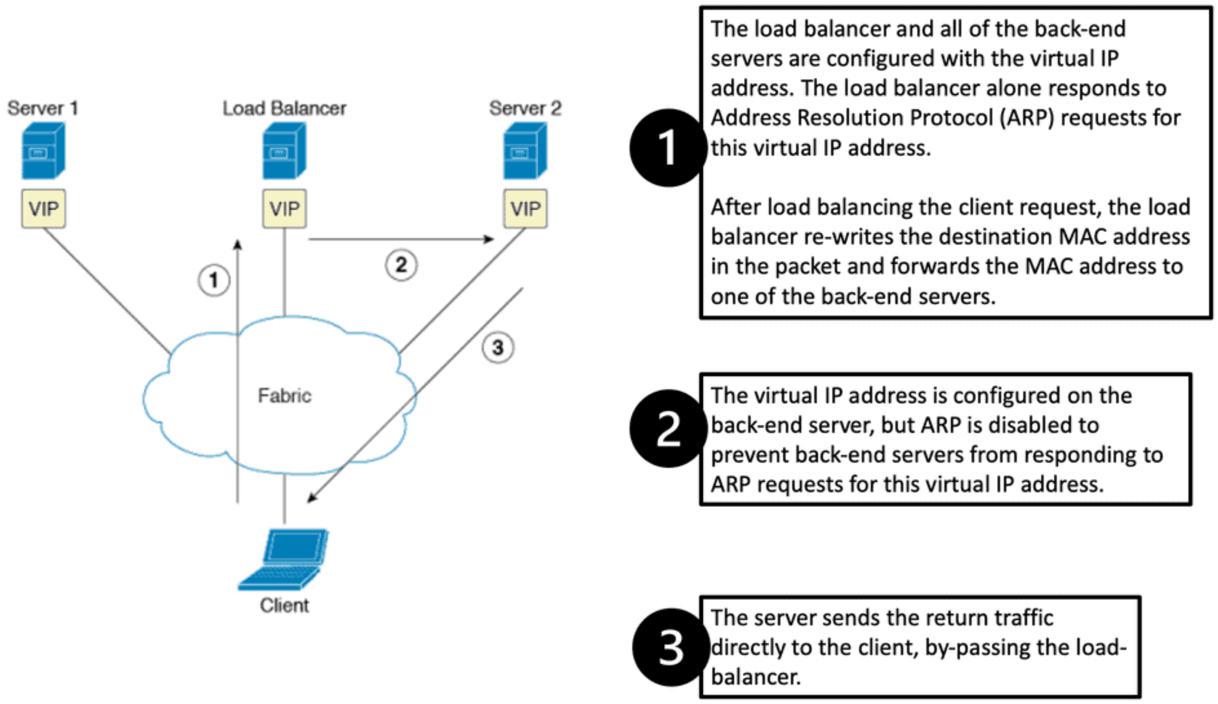
For example, you can buy a giant machine to replace your current server, known as **scale-up** and pricey, or another small device that works alongside your existing server, known as **scale-out**. As noted, the biggest challenge in load balancing is trying to make many resources appear as one. So we can have load balancing with **DNS, content delivery networks, and HTTP load balancing**. We also need to load balance our database and network connections.

- **Example: Direct Server Return.**

Direct server return (DSR) is an advanced networking technology that allows servers to send data directly to a client computer without going through an intermediary. This provides a more efficient and secure data transmission between the two, leading to faster speeds and better security.
DSR is also known as loopback, direct routing, or reverse path forwarding. It is essential in various applications, such as online gaming, streaming video, voice-over-IP (VoIP) services, and virtual private networks (VPNs).

server can be UDP, bypassing the load balancer. For this scenario, the load-balancing method of Direct Server Return is a viable option.

DSR is an excellent choice for high-speed, secure data transmission applications. It can also be used to help reduce latency and improve reliability. For example, DSR can help reduce lag and improve game performance in online gaming.



1 The load balancer and all of the back-end servers are configured with the virtual IP address. The load balancer alone responds to Address Resolution Protocol (ARP) requests for this virtual IP address.

After load balancing the client request, the load balancer re-writes the destination MAC address in the packet and forwards the MAC address to one of the back-end servers.

2 The virtual IP address is configured on the back-end server, but ARP is disabled to prevent back-end servers from responding to ARP requests for this virtual IP address.

3 The server sends the return traffic directly to the client, by-passing the load-balancer.

Diagram: Direct Server Return (DRS). Source Cisco.

How to scale load balancer

This post will first address the different load balancer scalability options that **consist of scale-up and then scale-out**. The scale-out is generally the path of scaling load balancers we see today, mainly as the traffic load, control, and data plane, are spread across VMs or containers that are easy to spin up and down, commonly seen for absorbing DDoS attacks. We will then discuss how to scale load balancer and the scalability options in the **application and at a network load balancing** level. We will finally address the different design options for load balancing, such as user **session persistence**, **destination-only NAT**, and **persistent HTTP sessions**.

In the diagram below, we see the following.

- **Virtual IP address:** A virtual IP address is an IP address that is used to virtualize a computer's identity on a local area network (LAN). The network address translation (NAT) form allows multiple devices to share a public IP address.
- **Load Balancer Function:** The load balancer is configured to receive client requests and then route the request to the most appropriate server based on a defined algorithm.

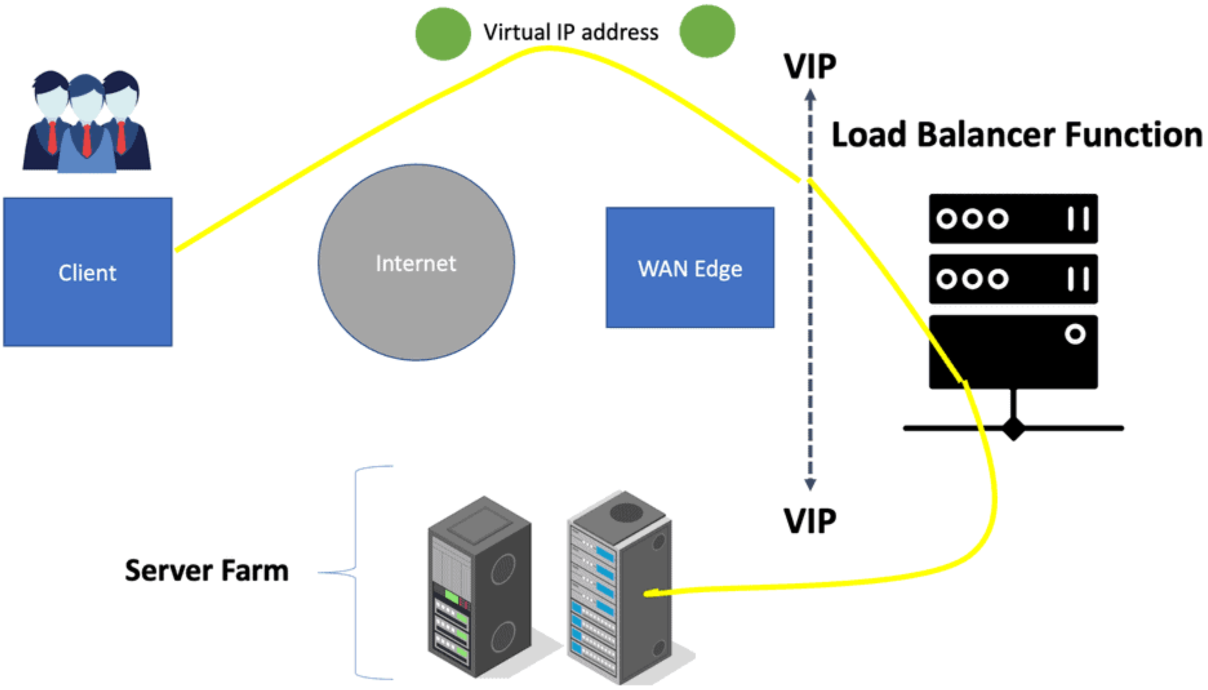


Diagram: How to scale load balancer and load balancer functions.

The primary benefit of load balancer scaling is that it provides scalability. Scalability is the ability of a networking device or application to handle organic and planned network growth. Scalability is the main advantage of load balancing, and in terms of application capacity, it increases the number of concurrent requests data centers can support. **So, in summary, load balancing is the ability to distribute incoming workloads to multiple end stations based on an algorithm.**

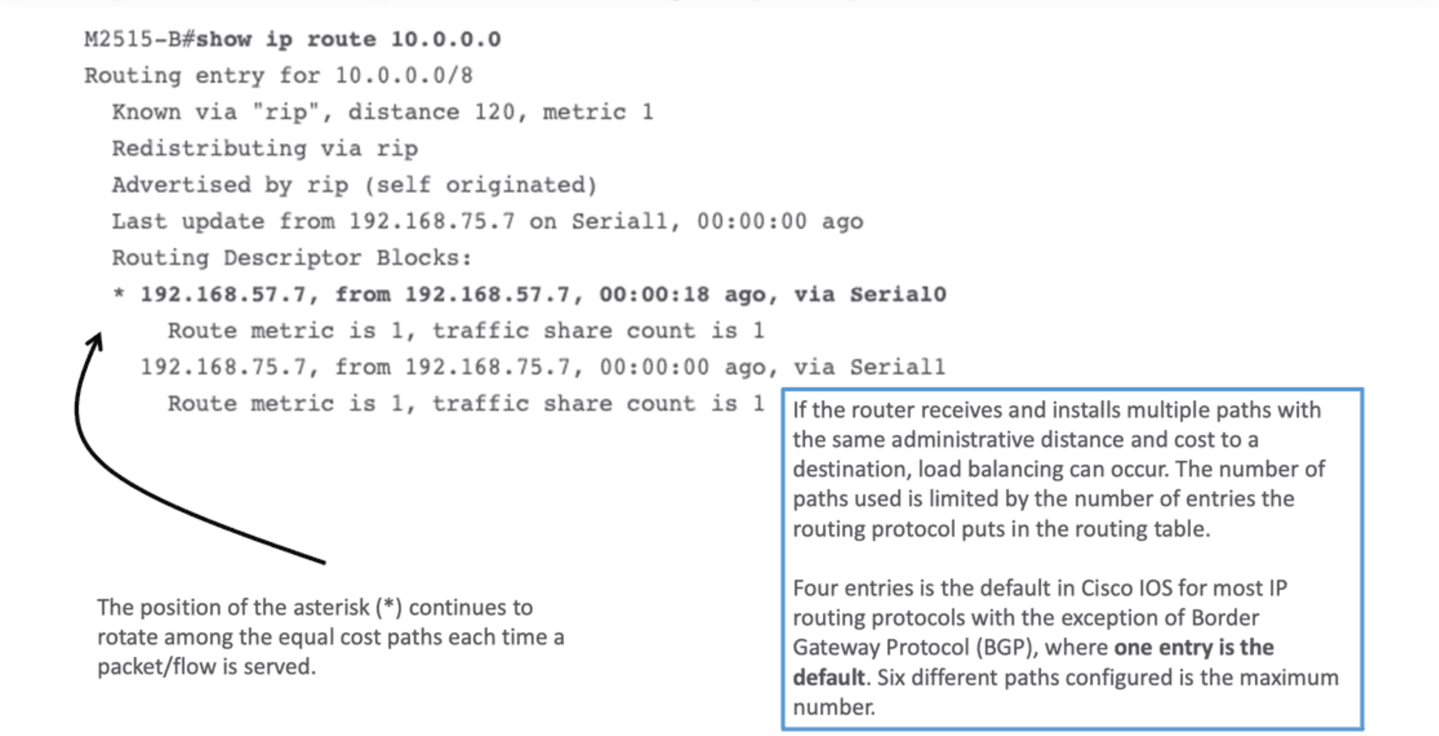
Load balancers also provide several additional features. For example, they can be configured to detect and remove unhealthy servers from the pool of available servers. They also provide **SSL encryption**, which can help to protect sensitive data being passed

Load Balancing Method 1	Round Robin Load Balancing
Load Balancing Method 2	Weighted Round Robin Load Balancing
Load Balancing Method 3	URL Hash Load Balancing
Load Balancing Method 4	Least Connection Method
Load Balancing Method 5	Weighted Least Connection Method
Load Balancing Method 6	Least Response Time Method

Load Balancing with Routers

Load Balancing is not limited to load balancer devices. Routers perform load balancing, too, with routing. Across all Cisco IOS® router platforms, load balancing is a standard feature. The router automatically activates this feature when multiple routes to a destination are in the routing table. Routing Information Protocol (RIP), RIPv2, Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), and Interior Gateway Routing Protocol (IGRP) are standard routing protocols, or they are derived from static routing and packet forwarding protocols. When forwarding packets, it allows a router to use multiple paths.

- For process-switching — load balancing is on a per-packet basis, and the asterisk (*) points to the interface over which the next packet is sent.
- For fast-switching — load balancing is on a per-destination basis, and the asterisk (*) points to the interface over which the next destination-based flow is sent.



```
M2515-B#show ip route 10.0.0.0
Routing entry for 10.0.0.0/8
  Known via "rip", distance 120, metric 1
  Redistributing via rip
  Advertised by rip (self originated)
  Last update from 192.168.75.7 on Serial1, 00:00:00 ago
  Routing Descriptor Blocks:
  * 192.168.57.7, from 192.168.57.7, 00:00:18 ago, via Serial0
      Route metric is 1, traffic share count is 1
    192.168.75.7, from 192.168.75.7, 00:00:00 ago, via Serial1
      Route metric is 1, traffic share count is 1
```

The position of the asterisk (*) continues to rotate among the equal cost paths each time a packet/flow is served.

If the router receives and installs multiple paths with the same administrative distance and cost to a destination, load balancing can occur. The number of paths used is limited by the number of entries the routing protocol puts in the routing table.

Four entries is the default in Cisco IOS for most IP routing protocols with the exception of Border Gateway Protocol (BGP), where **one entry is the default**. Six different paths configured is the maximum number.

Diagram: IOS Load Balancing. Source Cisco.

Load Balancer Scalability

Scaling load balancers with Scale-Up or Scale-Out

a) ***Scale-up* –** Expand linear by buying more considerable servers, adding CPU and memory, etc. Scale-up is usually done on transaction database servers as these servers are difficult to scale out. Scaling up is a simple approach but the most expensive and nonlinear. Old applications were upgraded by scaling up (vertical scaling)—a rigid approach that is not elastic. In a virtualized environment, applications are scaled linearly in a scale-out fashion.

b) ***Scale-out*** – Add more parallel servers, i.e., scaling linearly. Scaling out is easier on web servers; add additional web servers as needed. Netflix is an example of a company that designs by scale-out. It spins up Virtual Machines (VM) on-demand due to daily changes in network load. Scaling out is elastic and requires a load-balancing component. It is an

- **Additional information: Scale up load balancing**

A load balancer scale-up is the process of increasing the capacity of a load balancer by adding more computing resources. This can increase the system's scalability or provide redundancy in case of system failure. The primary goal of scaling up a load balancer is to ensure the system can handle the increased workload without compromising performance.

Scaling up a load balancer involves adding more hardware and software resources, such as CPUs, RAM, and hard disks. These resources will enable the system to process requests more quickly and efficiently. When scaling up a load balancer, consider its architecture and the types of requests it will handle. Different types of requests require different amounts of computing resources. For example, if the load balancer handles high-volume requests, it is essential to ensure that the system has enough CPUs and RAM to handle the requests.

Considering the network topology when scaling up a load balancer is also essential. The network topology defines how the load balancer will communicate with other systems, such as web servers and databases. If the network topology is not configured correctly, the system may be unable to handle the increased load. Finally, monitoring the system after scaling up a load balancer is essential. This will ensure that the system performs as expected and that the increased capacity is used effectively. Monitoring the system can also help detect potential issues or performance bottlenecks.

By scaling up a load balancer, organizations can increase the scalability and redundancy of their system. However, it is important to consider the architecture, types of requests, network topology, and monitoring when scaling up a load balancer. This will ensure the system can handle the increased workload without compromising performance.

- **Additional information: Scale-out load balancing**

Scaling out a load balancer adds additional load balancers to evenly distribute incoming requests across multiple nodes. The process of scaling out a load balancer can be achieved in a variety of ways. Organizations can use virtualization or cloud-based solutions to add additional load balancers to their existing systems. Some organizations

Finally, organizations should constantly monitor the load balancer's performance to ensure the system runs optimally. This can be done by tracking the load-balancing performance, analyzing the response time of requests, and providing that the system can handle unexpected spikes in traffic.

Load Balancer Scalability: The Operations

The virtual IP address and load balancing control plane

Outside is a VIP, and inside, a pool of servers exists. A load balancer scaling device is configured for rules associating outside IP and port numbers with an inside pool of servers. Clients only know the outside IP address through, for example, DNS replies. The load-balancing control plane monitors the servers' health and determines which can accept requests.

The client sends a TCP SYN packet, which the load balancer device intercepts. The load balancer carry's out a load-balancing algorithm and sends it to the best-server destination. To get the request to the server, you can use either **Tunnelling, NAT, or two TCP sessions**. In some cases, the load balancer will have to rewrite the content. Whatever the case, the load balancer has to create a session to know that this client is associated with a particular inside server.

Local and global load balancing

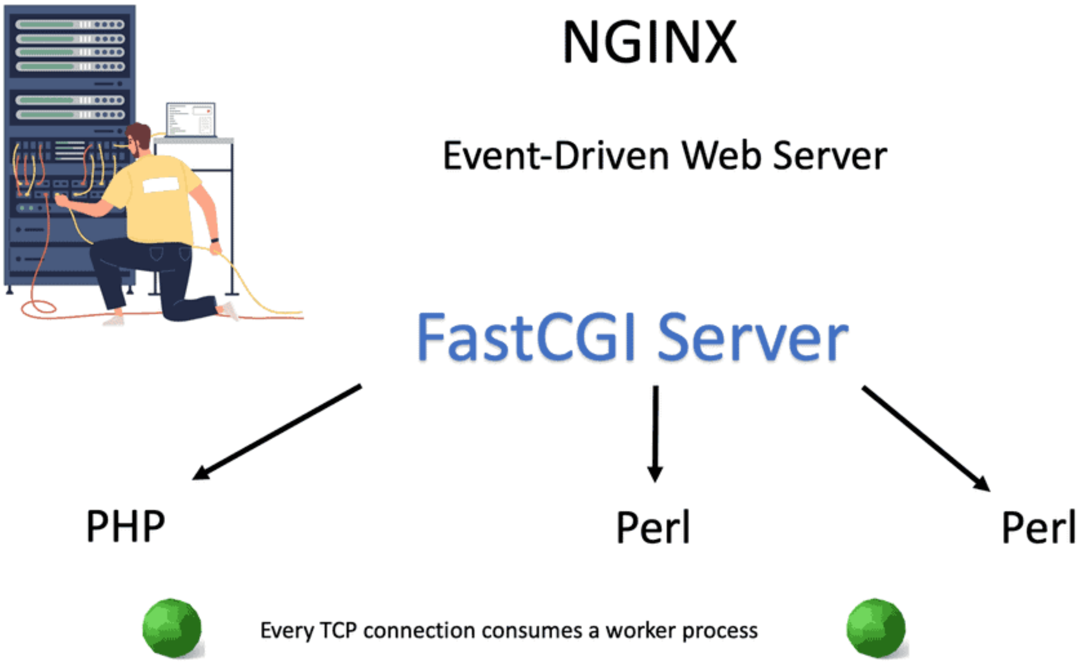
Local server selection occurs within the data center based on server load and application response times. Any application that uses TCP or UDP protocols can be load balanced. Whereas local load balancing determines the best device within a data center, global load balancing chooses the best data center to service client requests. Global load balancing is supported through redirection based on DNS and HTTP. HTTP mechanism provides better control, while DNS is fast and scalable. Both local and global appliances work hand-in-hand; the local device feeds information to the global device, enabling it to make better load-balancing decisions.

where applications are coded correctly, making it possible to configure load balancing in the application. Designers can use open-source tools with DNS or another method to track flows between tiers of the application stack.

Network-Level Load Balancer Scalability: Network-level load balancing includes DNS round-robin, Anycast, and Layer 4 – Layer 7 load balancers. Web browser clients do not usually have built-in application layer redundancy, which pushes designers to look at the network layer for load-balancing services. If applications were designed correctly, load balancing would not be a network-layer function.

Application-level load balancing

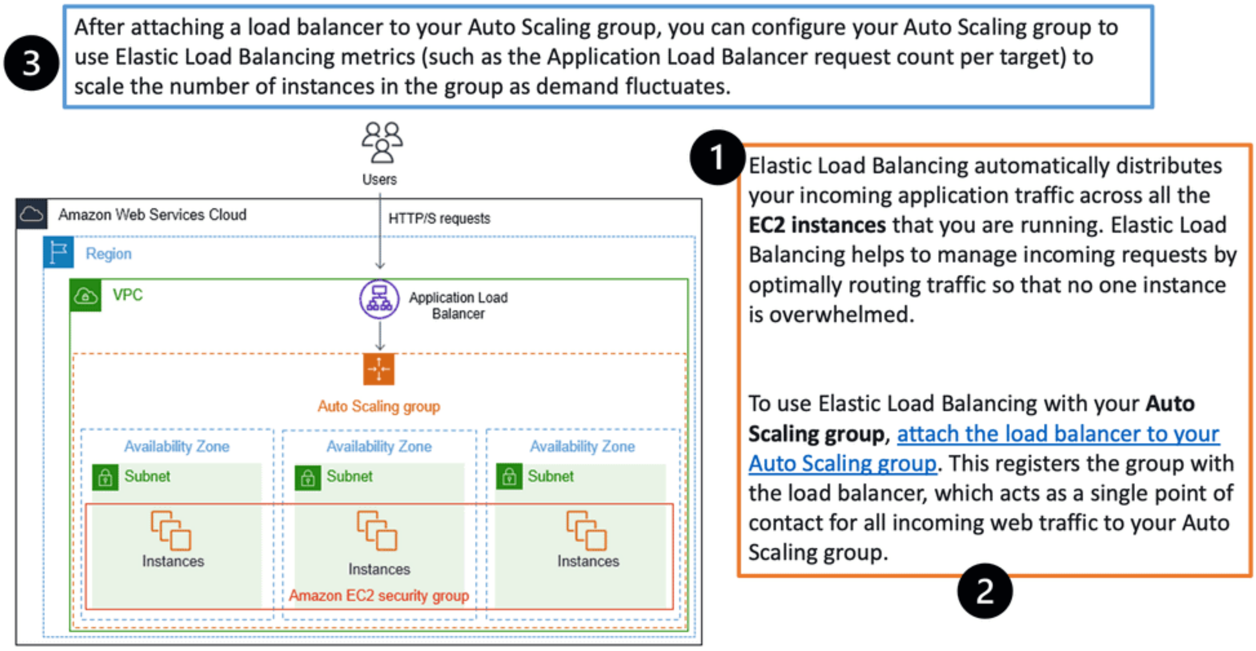
The application-level load balancer scaling is about what we can do inside the application for load-balancing services. The first thing you can do is scale-up – add a more-worker process. Clients issue requests that block some significant worker processes, and that resource is tied to TCP sessions. If your application requires session persistence (long-lived TCP sessions), you block worker processes even if the client is not sending data. The solution is FastCGI or changing the webserver to Nginx.



requests. Nginx does not use threads and can easily have 100,000 connections. With Apache, you lose 50% of the performance, and adding CPU doesn't help. With around 80,000 connections, you will experience severe performance problems no matter how many CPUs you add. Nginx is by far a better solution if you expect a lot of simultaneous connections.

Example: Load Balancing with Auto Scaling groups on AWS.

The following looks at an example of load balancing in AWS. Registering your Auto Scaling group with an Elastic Load Balancing load balancer helps you set up a load-balanced application. Elastic Load Balancing works with Amazon EC2 Auto Scaling to distribute incoming traffic across your healthy Amazon EC2 instances. This increases the scalability and availability of your application. In addition, you can enable Elastic Load Balancing within multiple Availability Zones to increase the fault tolerance of your applications. Elastic Load Balancing supports different types of load balancers. A recommended load balancer is the Application Load Balancer.



3 After attaching a load balancer to your Auto Scaling group, you can configure your Auto Scaling group to use Elastic Load Balancing metrics (such as the Application Load Balancer request count per target) to scale the number of instances in the group as demand fluctuates.

1 Elastic Load Balancing automatically distributes your incoming application traffic across all the **EC2 instances** that you are running. Elastic Load Balancing helps to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed.

To use Elastic Load Balancing with your **Auto Scaling group,** attach the load balancer to your Auto Scaling group. This registers the group with the load balancer, which acts as a single point of contact for all incoming web traffic to your Auto Scaling group.

2

First, try to solve the load balancer scaling in the application. When you cannot load balance solely using applications, turn to the network for load-balancing services.

DNS round-robin load balancing

The most accessible type of network-level load balancing is DNS round robin. DNS server that keeps track of application server availability. The DNS control plane distributes user traffic over multiple servers in a round-robin fashion. However, it does come with caveats:

1. DNS does not know server health.
2. DNS caching problems.
3. No measures are available to prevent DoS attacks against servers.

Clients ask for the IP of the web server, and the DNS server replies with an IP address using some random order. It works well if the application uses DNS. However, some applications use hard-coded IP addresses; you can't rely on DNS-based load balancing in these scenarios. DNS load balancing also requires **low TTL times**, so the client will often ask the servers. Generally, DNS-based load balancing works well but not with web browsers. Why? DNS pinning.

DNS pinning

This is because there have been so many attacks on web browsers, and browsers now implement a security feature-called DNS pinning. DNS pinning is a method whereby you get the server's IP address, and even though the TTL has expired, you ignore the DNS TTL and continue to use the URL. It prevents people from spoofing DNS records and is usually built-in to browsers. DNS load balancing is perfect if the application uses DNS and listens to DNS TTL times. But unfortunately, web browsers are not in that category.

IP Anycast load balancing

IP Anycast provides geographic server load balancing. The idea is to use the same IP address on multiple POPs. Routing in the core will choose the closest POP, routing the client to the nearest POP. All servers have the same IP address configured on loopback.

Best for UDP traffic

As requests come in, the router will load balance based on 5-tuple. Do not load the balance on destination addresses /ports, as it's always the same. It is usually done on the source client's the IP address/port number. The process takes the 5-tuple and creates a hash value, which creates independent paths based on that value. This works well for UDP traffic and how root servers work. Good for DNS server load balancing. It works well for UDP as every request from the client is independent. TCP does not work like this, as **TCP has sessions**. It recommended not to use Anycast load balancing for TCP traffic. You need an actual load balancer if you want to load-balance TCP traffic. This could be a software package, Open Source (HAproxy), or a dedicated appliance.

Scaling load balancers at Layer 2

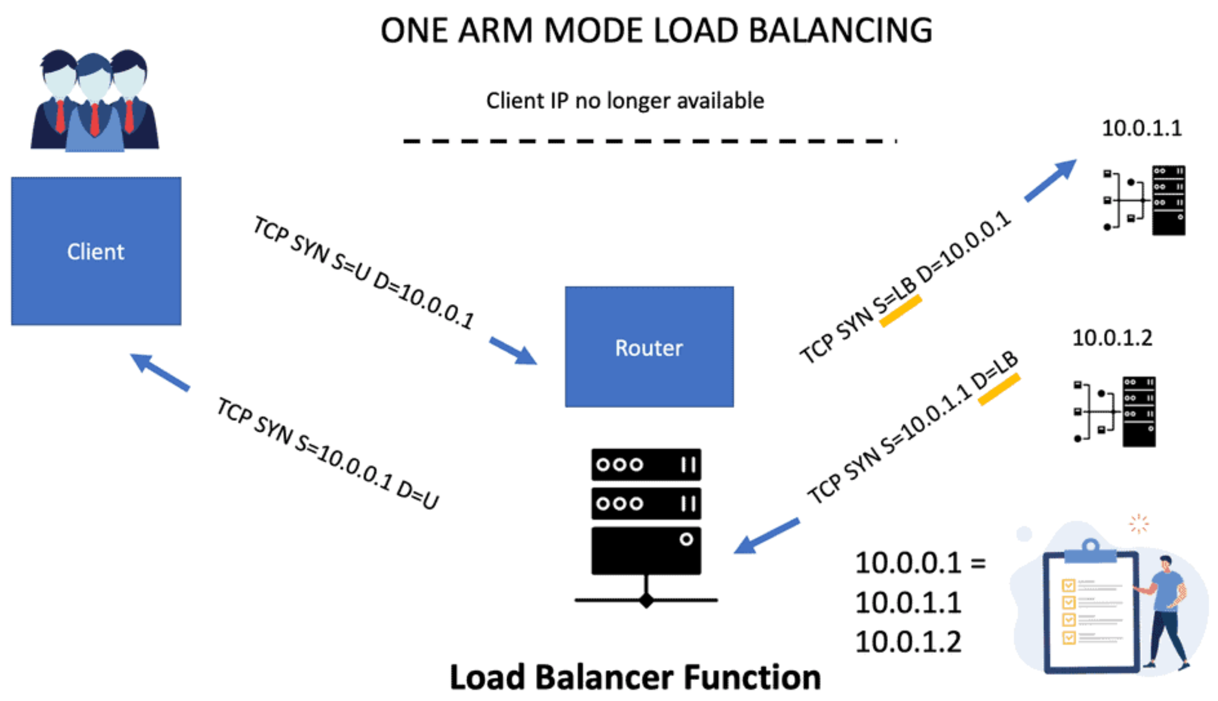
Layer 2 designs refer to the load balancer in *bridged mode.* As a result, all load-balanced and non-load-balanced traffic to and from the servers goes through the load-balancing device. The device bridges two VLANs together in the same IP subnet. Essentially, the load balancer acts as a crossover cable, merging two VLANs. The critical point is that the client and server sides are in the same subnet. As a result, layer 2 implementations are much more accessible than layer 3 implementations, as there are no changes to IP addresses, netmasks, and default gateway settings on servers. But with a bridged design, be careful about introducing loops and implementing spanning tree protocol (STP).

Scaling load balancers at Layer 3

With layer 3 designs, the load-balancing device acts in *routed mode.* Therefore, all load-balanced and non-load-balanced traffic to and from the server goes through the load-balancing device. The device routes between two different VLANs that are in two different subnets. The critical point and significant difference between layer 3 and layer 2 designs are client-side VLANs and server-side VLANs in different subnets. Therefore, the VLANs are not merged, and the load-balancing device routes between VLANs. Layer 3 designs may be more complex to implement but will eventually be more scalable in the long run.

One-armed mode refers to a load-balancing device, not in the forwarding path. The critical point is that the load balancer resides on its subnet and has no direct connectivity with server-side VLAN. A vital advantage of this model is that only load-balanced traffic goes through the device. **Server-initiated traffic bypasses the load balancer**. Changes both source and destination IP address. The load balancer terminates outside TCP sessions and initiates new inside TCP sessions. When the client connection comes in, you take the source IP and port number, put them in connection tables, and associate it with the load balancer's TCP port number and IP.

As everything comes from the load balance IP address, the servers can no longer see the original client. On the right-hand side of the diagram below, the source and destination traffic flow on the server side is the load balancer. The VIP addresses 10.0.0.1, and that is what the client connects to.



The use of X-forwarder-for HTTP header

To indicate to the server which the original client is, we use – **X-forwarder-for HTTP header**. The client's IP address is replaced with the load balancer IP address. The load balancer can insert the X-Forwarders-for HTTP header where they copy the original IP address of the client into the extra HTTP header – "*X-forward-for header*." Apache has a standard that copies the value of this header into the standard CGI variable so all the

Scaling load balancers with Direct Server Return

Direct Server Return is when the same IP address is configured on all hosts. The same IP is configured on the loopback interface, not the LAN interface. The LAN IP address is only used for ARP, so the load balancer would send ARP requests only for the LAN IP address, rewrite the MAC header (not TCP or HTTP alterations), and send the unmodified IP packet to the selected server.

The server sends the reply to the client **and does not involve the load balancer**. Requires layer 2 connectivity between the load balancer and servers as load balancing is done on the MAC address (example: Linux Virtual Server). Also, a tunneling method that uses Layer 3 between the load balancer and servers is available.

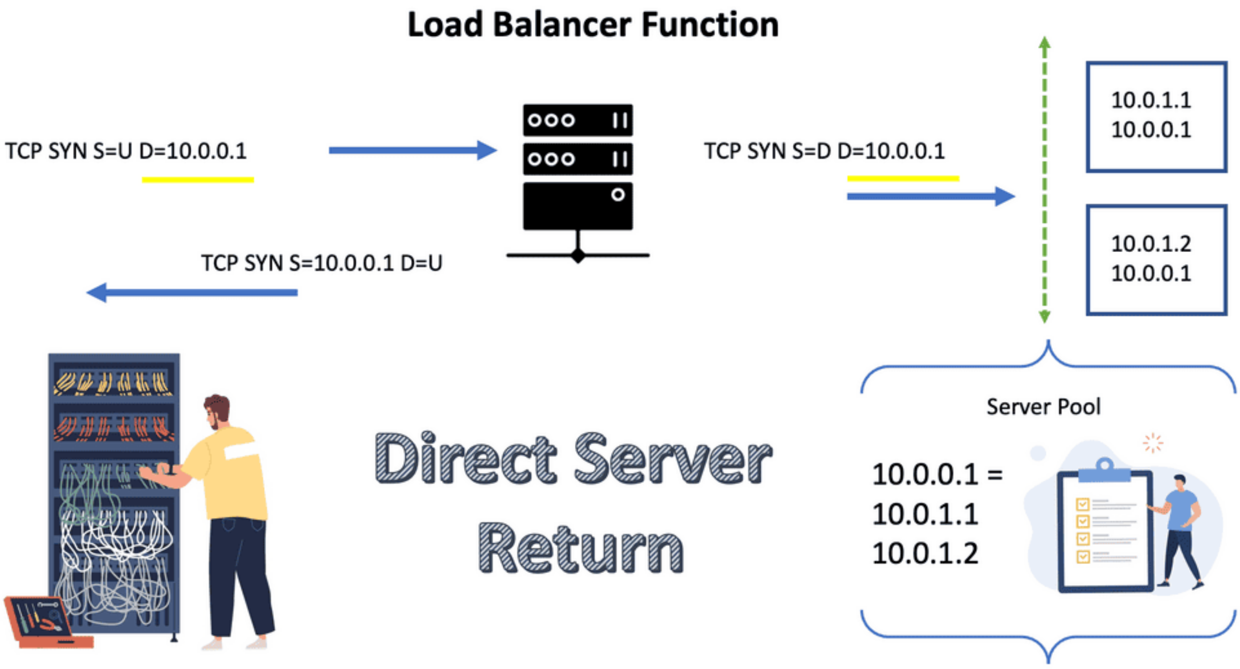


Diagram: Direct Server Return.

- A key point: MTU issues

If you do not have layer 2 connectivity, you can use tunnels, but be aware of MTU issues. Make sure the Maximum Segment Size (MSS) on the server is reduced, so you do not for TCP, the source in the reply is always copied from the destination IP address in the original TCP SYN request.

Scaling load balancers with Microsoft network load balancing

Microsoft load balancing is the ability to implement load balancing without load balancers. Instead, create a cluster IP address for the server and then use the *flooding behavior* to send it to all servers. Clients send a packet to the shared cluster IP address associated with a client's MAC address. This cluster MAC does not exist anywhere. When the request arrives at the last Layer 3 switch, it sends an ARP request "*Who has this IP address*"?.

ARP request arrives at all the servers. So, when the client packet arrives, it sends to the bogus MAC address of the cluster, and because the MAC address has never been associated with any source, all the traffic is flooded from the Layer 2 switch to the servers. The performance of the Layer 2 switch falls massively as unicast flooding is done in software.

The use of Multicast

Microsoft then changed this to use Multicast. This does not work, and packets are dropped as an illegal source MAC to use a multicast MAC address. Cisco routers drop ARP packets with the source MAC address as multicast. Overcome this by configuring static ARP entries. Microsoft also implements IGMP to reduce flooding.

Load Balancing Options

User session persistence (Stickiness)

The load balancer must keep all sessions state, even for inactive sessions. Session

The client establishes an HTTP session with the webserver and logs in. After login, the HTTPS session from the same client should land on the same web server to which the client first logged in using the initial HTTP request. The following are ways load balancers can determine who the source client is.

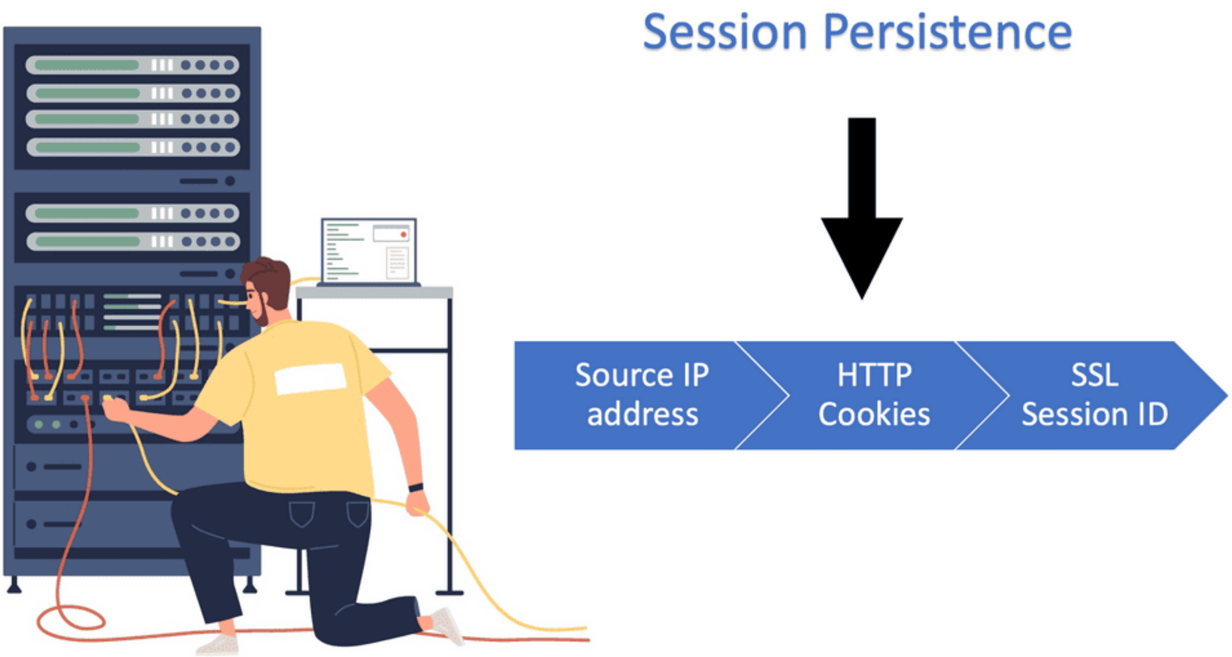


Diagram: Scaling load balancers and session persistence.

- **Source IP address –** > Problem may arise with large-scale NAT designs.
- **Extra HTTP cookies** – > May require the load balancer to take ownership of the TCP session.
- **SSL session ID** -> Will retain session persistence even if the client is roaming and the client's IP address changes.

Data path programming

F5 uses scripts that act on packets triggering the load-balancing mechanism. You can select the server, manipulate HTTP headers or even manipulate content. For example, the

Persistent HTTP sessions

The client has a long-lived HTTP session to eliminate one RTT and congestion window problem; then, we have a short-lived session from the load balancer to the server. SPDY is a next-generation HTTP with multiple HTTP sessions over one TCP session. This is useful in high-latency environments such as mobile devices. F5 has a SPDY-to-HTTP gateway.

Destination-only NAT

Rewrites the destination IP address to a destination IP of the actual server and then forwards the packet. The reply packet has to hit the load balancer as the load balancer has to replace the server's source IP with the load balancer's source IP. The client IP does not change, so the server is talking directly with the client. This allows the server to do address-based access control or GEO location based on the source address.

- **A quick summary: How to scale load balancer**

This post first addressed the different load balancer scalability options that consist of scale-up and then scale-out. The scale-out is generally the path of scaling load balancers we see today. It is less expensive and easier to perform. We then discussed how to scale the load balancer and the load balancer scalability options in the application and at a network level load balancing.

We also discussed the different design types of load balancing, such as user session persistence, destination-only NAT, and persistent HTTP sessions. There were several videos included that could provide more details on scaling load balancers. So when you ask yourself how to scale load balancer, the first step is to examine the application. Can this be solved in the application, or do we need to push this to the network layer? Both have their pros and cons.

Matt Conran has more than 24 years of networking and security industry with entrepreneurial start-ups, government organizations, and others. He now focuses on public speaking, authoring content, consulting, and creating Elearning courses.

Comments are closed.

Recent Blogs

DMVPN

Computer Networking

eBOOK – SASE Capabilities

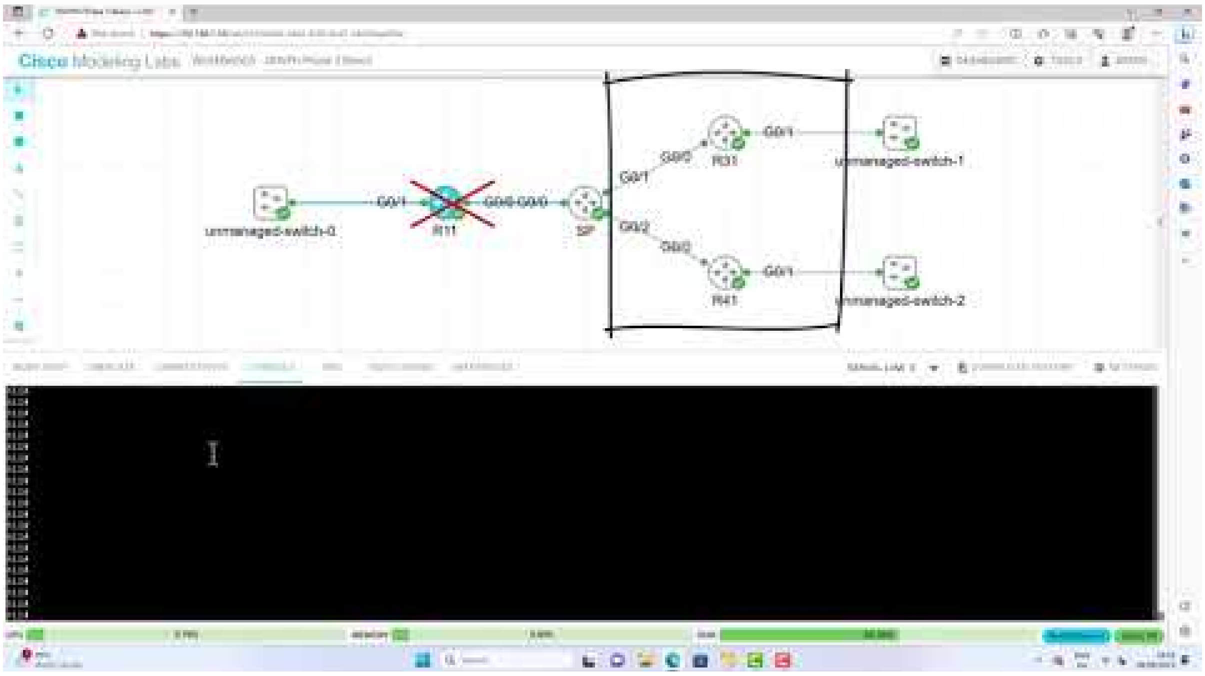
Recent Pluralsight

CCT Routing and Switching: Networking Foundations

The IT Ops Sessions: Observability and Cisco Thousand Eyes

CyberArk Fundamentals

Recent Youtube



Recent Publications

SD-WAN management means more than reviewing logs and parsing events

Secure SD-WAN: The security vendors and their SD-WAN offerings

AI and 5G: Entering a new world of data

Subscribe to Blog via Email

Subscribe

➞ Make an enquiry

Home Blogs News Publications Pluralsight Videos

Talent Garden Dublin, Claremont Avenue, Glasnevin, Dublin 11.

© Conran Insight Ltd

00353 87 2806033 | matt@conran-insight.com