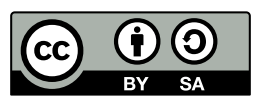
# Load Balacing
# From zero to XDP

Luca Bassi
luca@argoware.com

February 24, 2023



# Outline

# What is load balancing?

Load balancing is the practise of distributing workloads between multiple computers.

It is widely used to distribute network traffic among several servers.

Load balancer, software or hardware-based, determine which server should handle the request based on an algorithm.

Two category of algorithms:

- Static
- Dynamic

# Static load balancing algorithms

Static load balancing algorithms ignore the status of servers and distribute the traffic based on assumptions about the system made beforehand.

# Round-robin DNS

Round-robin DNS is a technique of load balancing.

It distributes the traffic assigning multiple IP to the same DNS record, permuting the sequence with each DNS request.

Pros:

- Easy to deploy and understand.
- Do not require specialized software or hardware.

Cons:

- DNS response are often cached by clients.
- Ignore servers load.
- Assume that all requests are computationally equal.
- Assume all servers have equivalent capacity.

# Server-side load balancers

Another classic approach to load balancing is the deployment of a server with a load balancer software (ex. nginx) listening on the port where clients connect and redirecting the traffic to one of the backend server.

nginx operates as a Layer 7 load balancer.

Reverse proxy implementation in nginx includes load balancing for HTTP, HTTPS, FastCGI, uwsgi, SCGI, memcached and gRPC.

# nginx as a load balancer

Pros:

- More algorithms available: round-robin, least-connected, ip-hash.
- More expressive than Layer 4 load balancers.

Cons:

- Less efficient than Layer 4 load balancers.
- Do not support all protocols.

# LVS and direct routing

Linux Virtual Server is a load balancer to distribute the IP load across a set of real servers.

The real servers appear as one entity to the Internet.

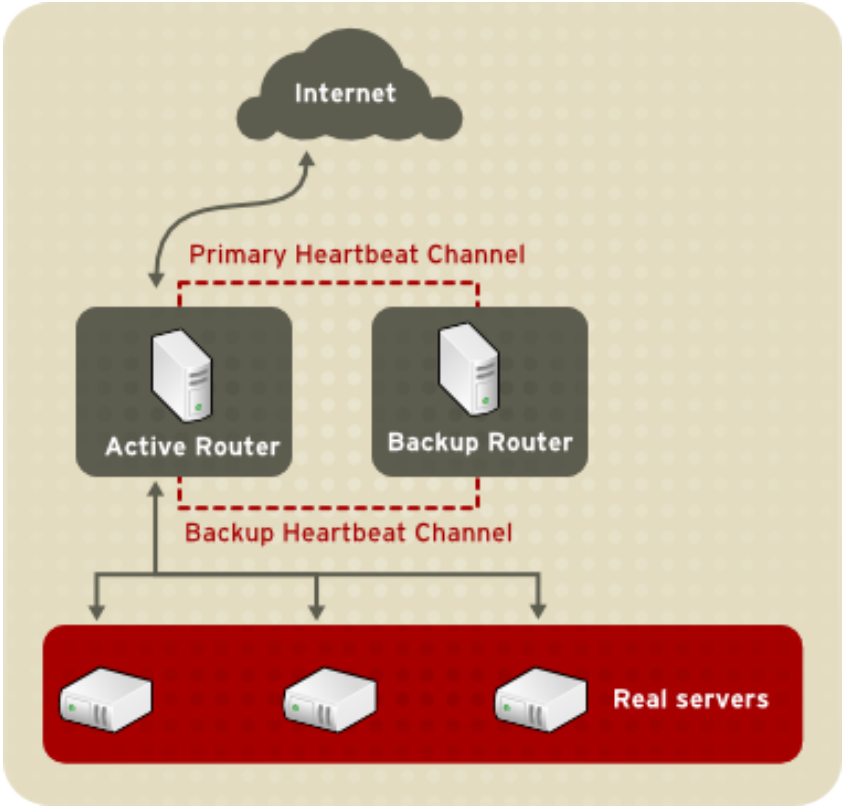



Image by Red Hat / CC BY-SA 3.0

# LVS and direct routing

In the previous basic LVS configuration, all traffic pass back and forth through the LVS router.
We can use direct routing to have outbound traffic served directly from real servers.

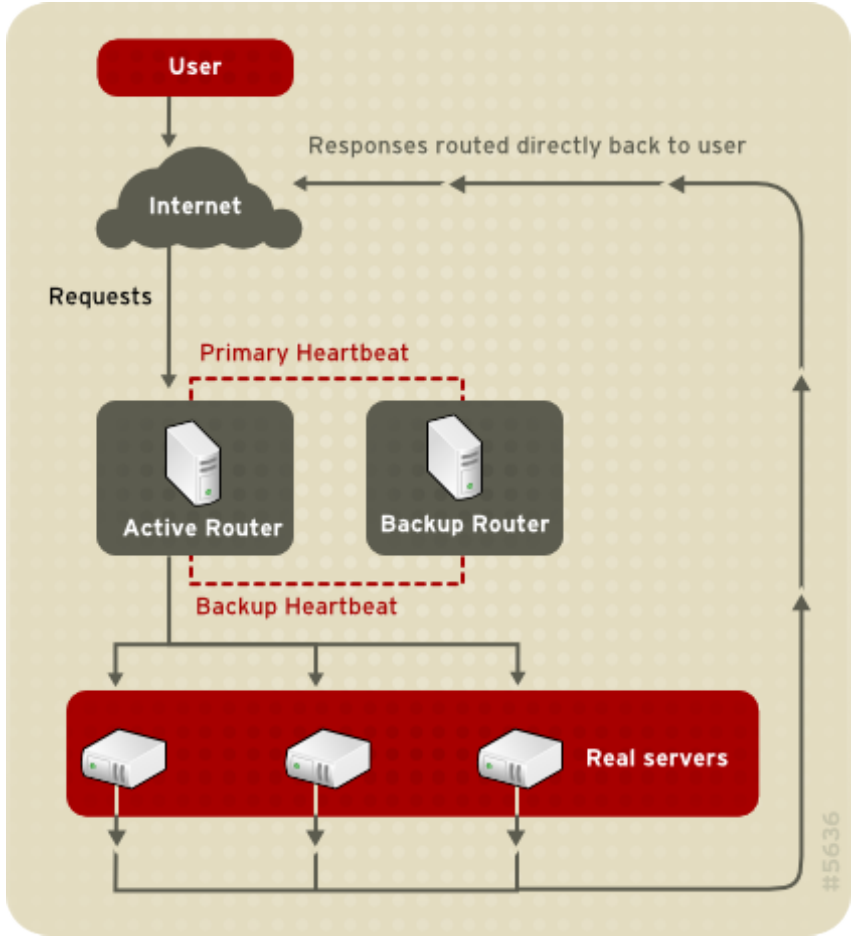



Image by Red Hat / CC BY-SA 3.0

# LVS and direct routing

Pros:

- Outgoing traffic do not pass through the LVS router.
- Several balancing schedulers.

Cons:

- All incoming traffic pass through the LVS router.

# ECMP

Routers have a feature called Equal-Cost Multi-Path (ECMP) routing, which is designed to split traffic destined for a single IP across multiple links of equal cost.

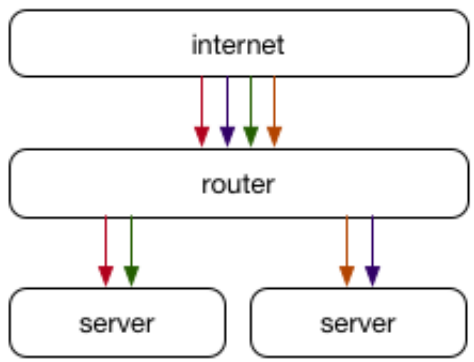
ECMP works by hashing certain components of an incoming packet, such as the source and destination IP addresses and ports.

By using a consistent hash for this, subsequent packets that are part of the same TCP flow will hash to the same path, avoiding out of order packets and maintaining session affinity.

# ECMP

An alternative use of ECMP can come in to play when we want to shard traffic across multiple servers rather than to the same server over multiple paths.

Each server can announce the same IP address with BGP or another similar network protocol, causing connections to be shared across those servers, with the routers blissfully unaware that the connections are being handled in different places, not all ending on the same machine as would traditionally be the case.

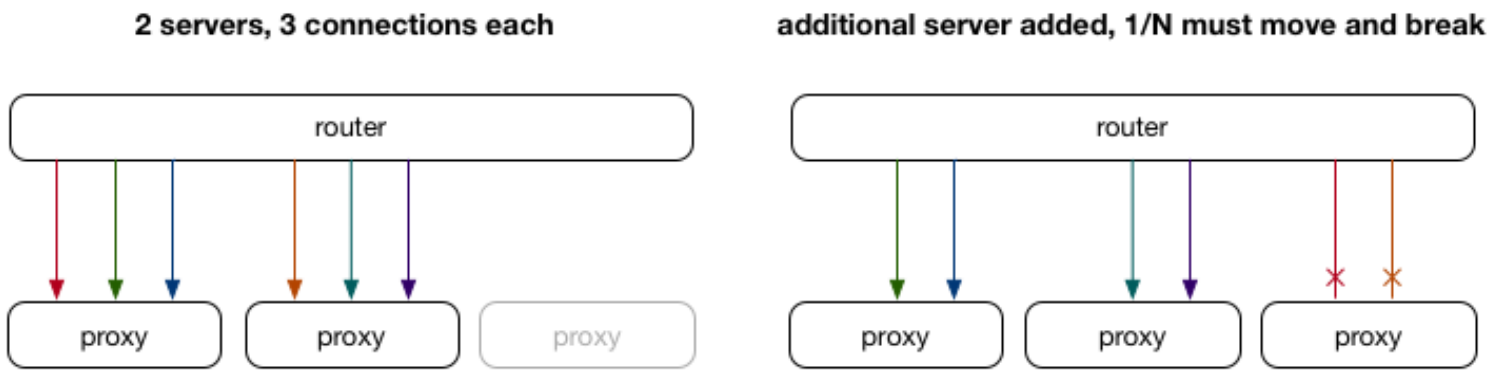



Copyright GitHub

# ECMP

When the set of servers that are announcing the same IP change (or any path or router along the way changes), connections must rebalance to maintain an equal balance of connections on each server.

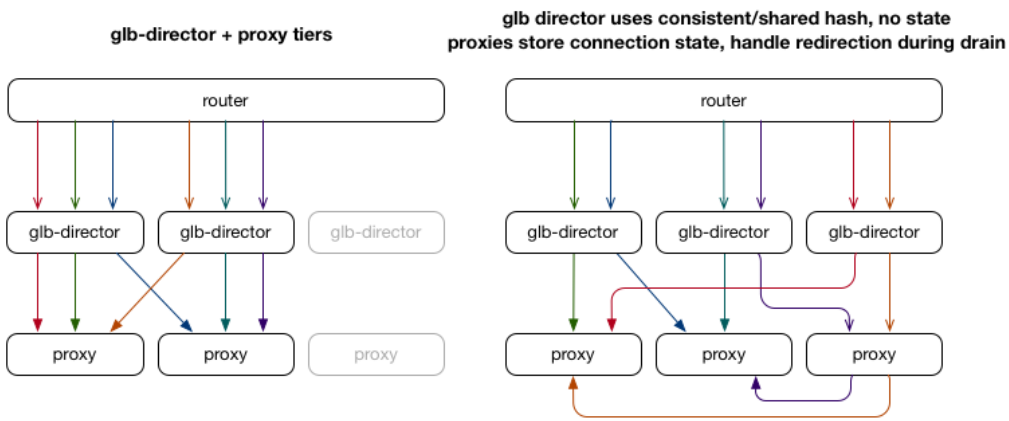
Routers are typically stateless devices, simply making the best decision for each packet without consideration to the connection it is a part of, which means some connections will break in this scenario.



# GitHub: GLB

GLB Director is a Layer 4 load balancer which scales a single IP address across numerous physical machines while attempting to minimize connection disruption during any change in servers.

For each incoming connection, it picks a primary and secondary server that could handle that connection. When a packet arrives at the primary server and isn't valid, it is forwarded to the secondary server.



# GitHub: GLB

GLB uses DPDK, an open source project that allows very fast packet processing from userland by bypassing the Linux kernel.

Pros:

- Very efficient load balancing.
- It's possible to gracefully shutdown servers without disrupting exiting connections.

Cons:

- DPDK is compatible only with some NICs.

# eBPF

eBPF is an evolution of Berkeley Packet Filter.

It permits to run sandboxed programs in the OS kernel, mainly event-driven.

It's used to extend the kernel without recompiling it or use modules.

Safety is provided through static code analysis.

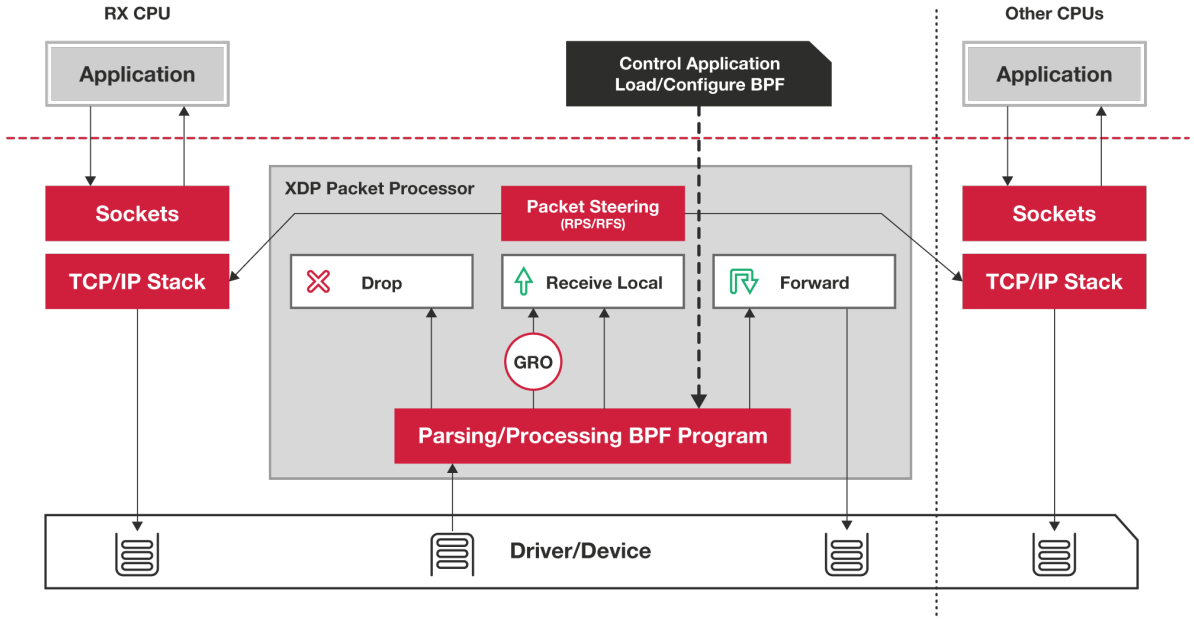
The website states:

> *"Today, eBPF is used extensively to drive a wide variety of use cases: Providing high-performance networking and load-balancing in modern data centers and cloud native environments [...]"*

# XDP

eXpress Data Path provides a high performance, programmable network data path in the Linux kernel.

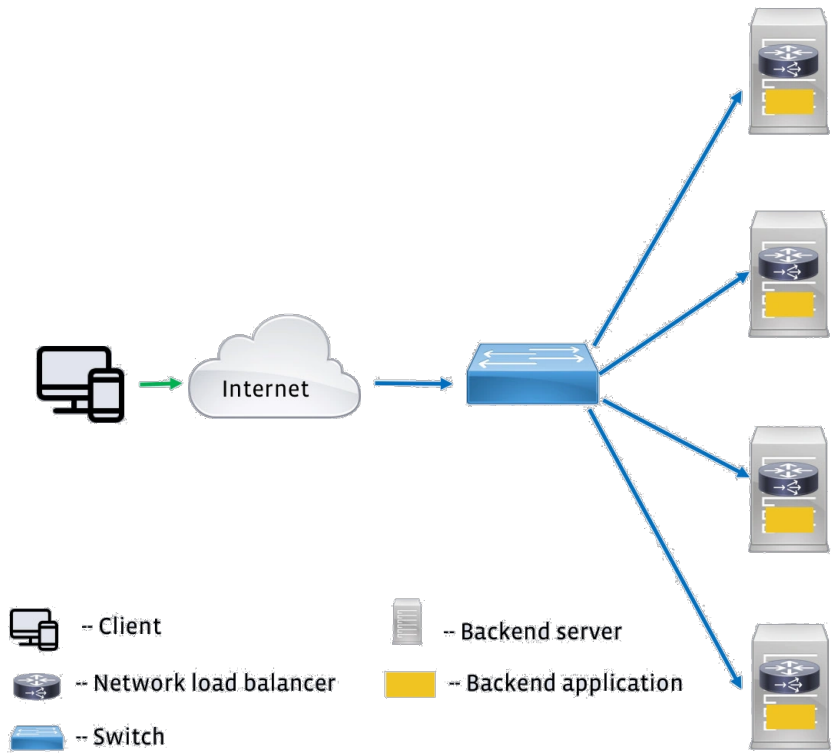
It provides bare metal packet processing at the lowest point in the software stack, which makes it ideal for speed.



# Facebook: Katran

In 2018, Facebook open-sourced Katran, a load balancer that use XDP.

1. ExaBGP announces to the world which VIPs a particular Katran instance is responsible for.
2. Packets destined to a VIP are sent to Katran instances using an ECMP mechanism.
3. Katran selects a backend and forwards the packet to the correct backend server (it uses XDP in combination with a BPF program).



# Facebook: Katran

Pros:

- Very efficient load balancing.
- Katran runs alongside any application without any performance penalties on the same host, in contrast to a full-fledged Kernel bypass solution (such as DPDK).

Cons:

- Katran cannot forward fragmented packets, nor can it fragment them by itself.
- Katran doesn't support packets with IP options set. The maximum packet size cannot exceed 3.5 KB.

# Beamer: Stateless Load-balancing

In 2018, four researchers of the University Politehnica of Bucharest presented Beamer, a stateless data centre load balancer.

The architecture of Beamer is: load balancers run BGP (Quagga) and announce the same VIP to border routers. ECMP at the routers spreads packets across the load balancers, which direct traffic to servers, finally the servers respond directly to clients.

They introduce daisy chaining forwarding between servers.
If server A receives a packet belonging to a connection, for which it does not have an entry in the open connections table, the default behaviour is to reset the connection. If A knew that server B might have state for this connection, it could simply forward all packets it doesn't have state for to B, where they could be processed normally.

# Beamer: Stateless Load-balancing

Pros:

- Stateless thanks to daisy chaining.

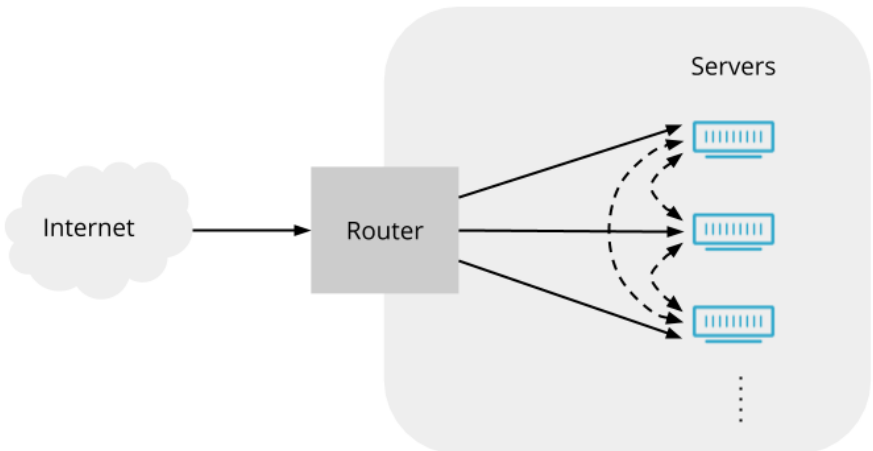
Cons:

- Use kernel modules.
- Requires a kernel patch.

# Cloudflare: Unimog

Unimog is a layer 4 dynamic load balancer: it takes regular measurements of the load on each of our servers and uses a control loop that increases or decreases the number of connections going to each server.

Unimog makes every server into a load balancer. The router can send any packet to any server, and that initial server will forward the packet to the right server for that connection.



# Cloudflare: Unimog

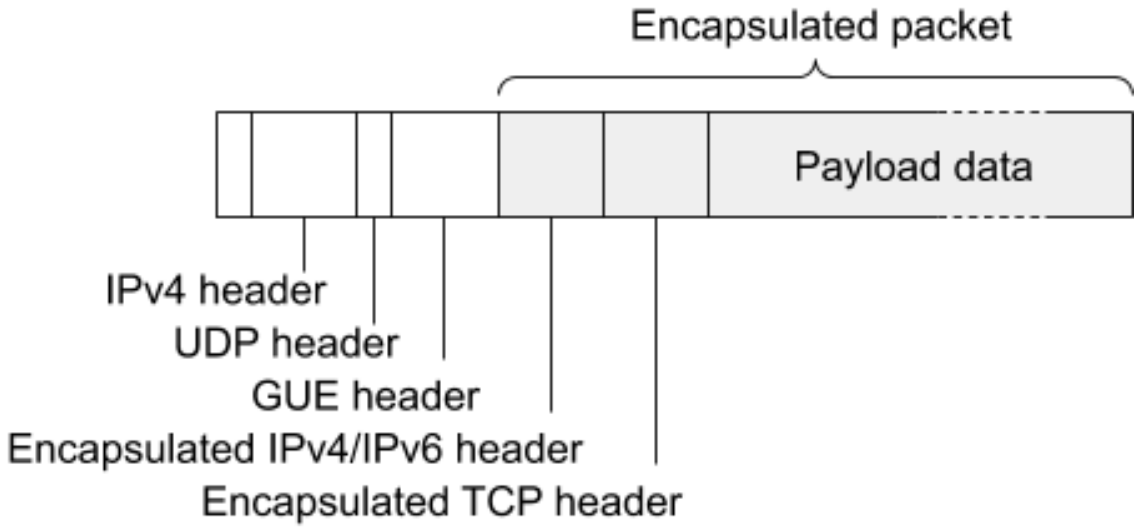
Unimog implements packet forwarding using XDP. XDP allows a program to be attached to a network interface, and the program gets run for every packet that arrives, before it is processed by the kernel's main network stack. The XDP program returns an action code to tell the kernel what to do with the packet:

- PASS: Pass the packet on to the kernel's network stack for normal processing.
- DROP: Drop the packet.
- TX: Transmit the packet back out of the network interface. The XDP program can modify the packet data before transmission.

# Cloudflare: Unimog

Before redirecting packets, they must be encapsulated.

Unimog use GUE (Generic UDP Encapsulation).



# Cloudflare: Unimog

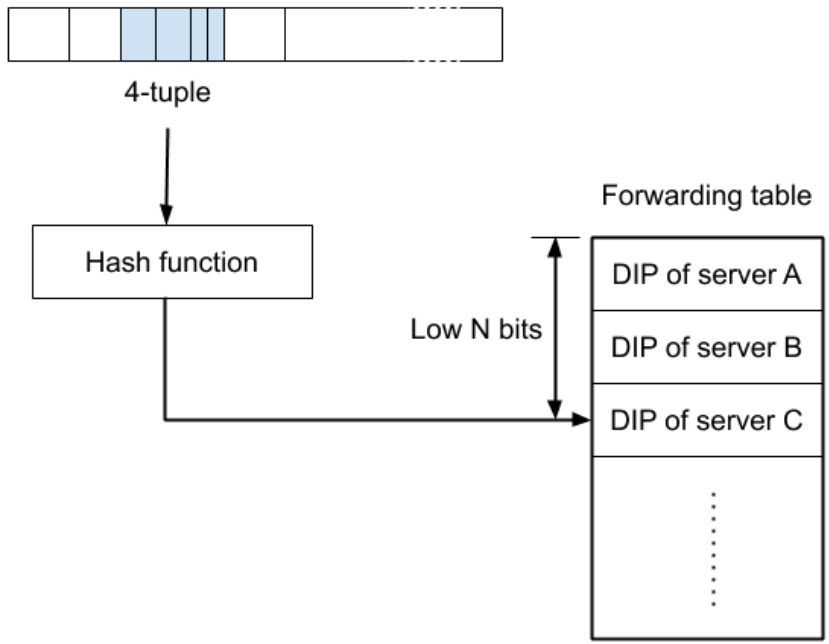
Unimog XDP program processes each packet in the following way:

1. Determine whether the packet is destined for a VIP address. Not all the packets arriving at a server are for VIP addresses. Other packets are passed through for normal handling by the kernel's network stack.
2. Determine the direct IP (DIP) for the server handling the packet's connection.
3. Encapsulate the packet and retransmit it to the DIP.

In step 2, note that all the load balancers must act consistently. L4LBs adopt designs which allow the load balancers to reach consistent forwarding decisions independently. To do this, they rely on hashing schemes.

# Cloudflare: Unimog

A forwarding table consists of an array where each entry contains a DIP specifying the target server for the relevant packets.
The forwarding table is generated by the Unimog control plane and broadcast to the load balancers, so that it has the same contents on all load balancers.
This means it doesn't matter which packets are sent by the router to which servers, and so ECMP re-hashes are a non-issue.



# Cloudflare: Unimog

Updating a forwarding table, and changing the DIPs in some buckets, would break connections that hash to those buckets (because packets on those connections would get forwarded to a different server after the update).

Unimog adopts the "daisy chaining" technique to make changes to the forwarding table without breaking established connections.

Unimog extends the forwarding table so that each bucket has two slots, each containing the DIP for a server. The first slot contains the current DIP, the second one preserves the previous DIP (if any).

# Cloudflare: Unimog

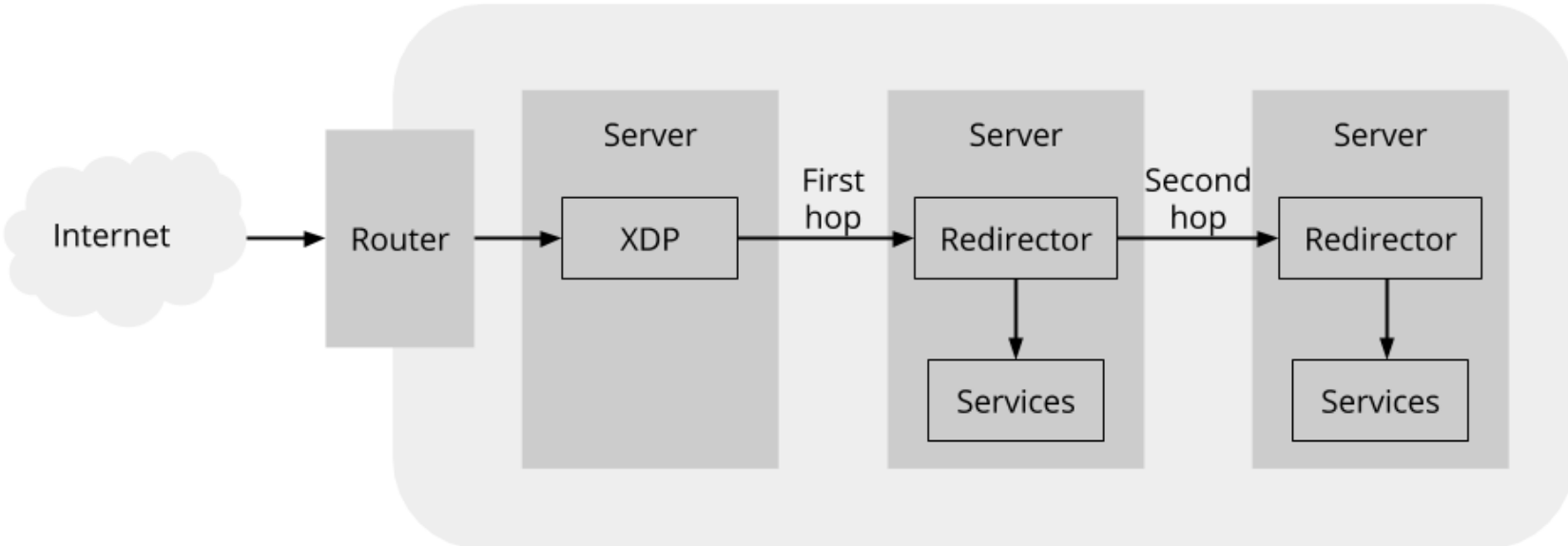
The redirection logic is:

- If the packet is a SYN packet (initiating a new connection), then it is always processed by the first server.
- For other packets, it is processed by the first server if the packet belongs to a connection with a corresponding TCP socket on that server.
- Otherwise, it's forwarded to the second server.

The second DIP is carried in a GUE extension header to avoid a second forwarding table lookup.

The redirector is implemented as a TC (traffic control subsystem) classifier program for easier debugging.

# Cloudflare: Unimog



# Cloudflare: Unimog

A control plane is necessary to generate the forwarding tables.

The control plane takes into account:

- Server information: the set of servers present in the data centre, DIPs, status etc.
- Health: if the server is online and services are running correctly.
- Load: to balance load (e.g. resource utilization).
- IP address information: if not statically configured.

The control plane obtains this data from Consul and Prometheus.

# Cloudflare: Unimog

In case of UDP there is no SYN packets, so the redirection logic is simply:

- A packet is processed by the first server if it belongs to a connection with a corresponding UDP socket on that server.
- Otherwise, it's forwarded to the second server.

Note that for UDP, new flows go to the second server unlike TCP's SYN packets.

# Cloudflare: Unimog

Pros:

- Very efficient load balancing.
- No specialized roles for servers.

Cons:

- Encapsulating a 1500-byte packet results in a 1536-byte packet, so jumbo frames must be enabled on the networks inside the data centre.
- Not open source.

# Links

Direct Routing

DPDK

eBPF

XDP

Extending extended BPF

Introducing the GitHub Load Balancer

A thorough introduction to eBPF

Stateless Datacenter Load-balancing with Beamer

Open-sourcing Katran, a scalable network load balancer

GLB: GitHub's open source load balancer

Unimog - Cloudflare's edge load balancer

BPF as a safer kernel programming environment