RTB HOUSE

13.05.2024

Load balancing

Jarosław Rzeszótko
RTB HOUSE

What is load balancing?

Process of distributing traffic among many servers capable of handling it.

Two purposes:

- Scaling
- Availability

What is load balancing?

Process of distributing traffic among many servers capable of handling it.

Two slightly different application contexts:

- Load balancing incoming traffic from end users to client-facing app servers (traffic comes from a web browser)
- Load balancing traffic between app servers and other services inside the datacenter(traffic comes from a Java/Python/Go/C++/... programmable client)

DNS Round Robin load balancing

Put multiple IP addresses in DNS A records for the domain, e.g. xyz.com

DNS server will permute the list of IP addresses each time it is queried, returning the addresses in different order

DNS Round Robin load balancing

What will a browser do
when attempting to connect xyz.com?

- It will get the list of IPs from a DNS server and try to connect to the first IP
- As long as all servers in the list of IPs are up you get some load distribution among the servers

- If one of the servers is down, browser will wait until connection timeouts before trying next IP -> failover is slow
- No easy way to quickly remove a server from the pool because of DNS caching etc.
- Not great for load balancing client traffic

RTB HOUSE

DNS Round Robin load balancing

What will a programmatic client do when connecting to xyz.com?

For example a Python program like this:

```
import requests
requests.get(“https://www.xyz.com”)
```

Aside: layers in the software stack

Where can functionality like DNS resolution reside?
import requests requests.get("https://www.xyz.com")

Python interpreter could contain full code necessary to do DNS resolution,
but it could also do a ??? or use a function from the ??? library

Tip: what are the Python interpreter, Java VM and many other language interpreters and VMs all written in?

DNS Round Robin load balancing

What will a programmatic client do when connecting to xyz.com?

- It will call getaddrinfo() from the C standard library (typically glibc) which will sort the IP list by network “nearness”
- Unless you manually program resolving the domain name to full IP list and pick an IP at random, your program will always use the same, nearest IP

- Makes DNS round robin poor also for inside-DC load balancing, unless you sure you can control the DNS resolution code
- https://daniel.haxx.se/blog/2012/01/03/getaddrinfo-with-round-robin-dns-and-happy-eyeballs/

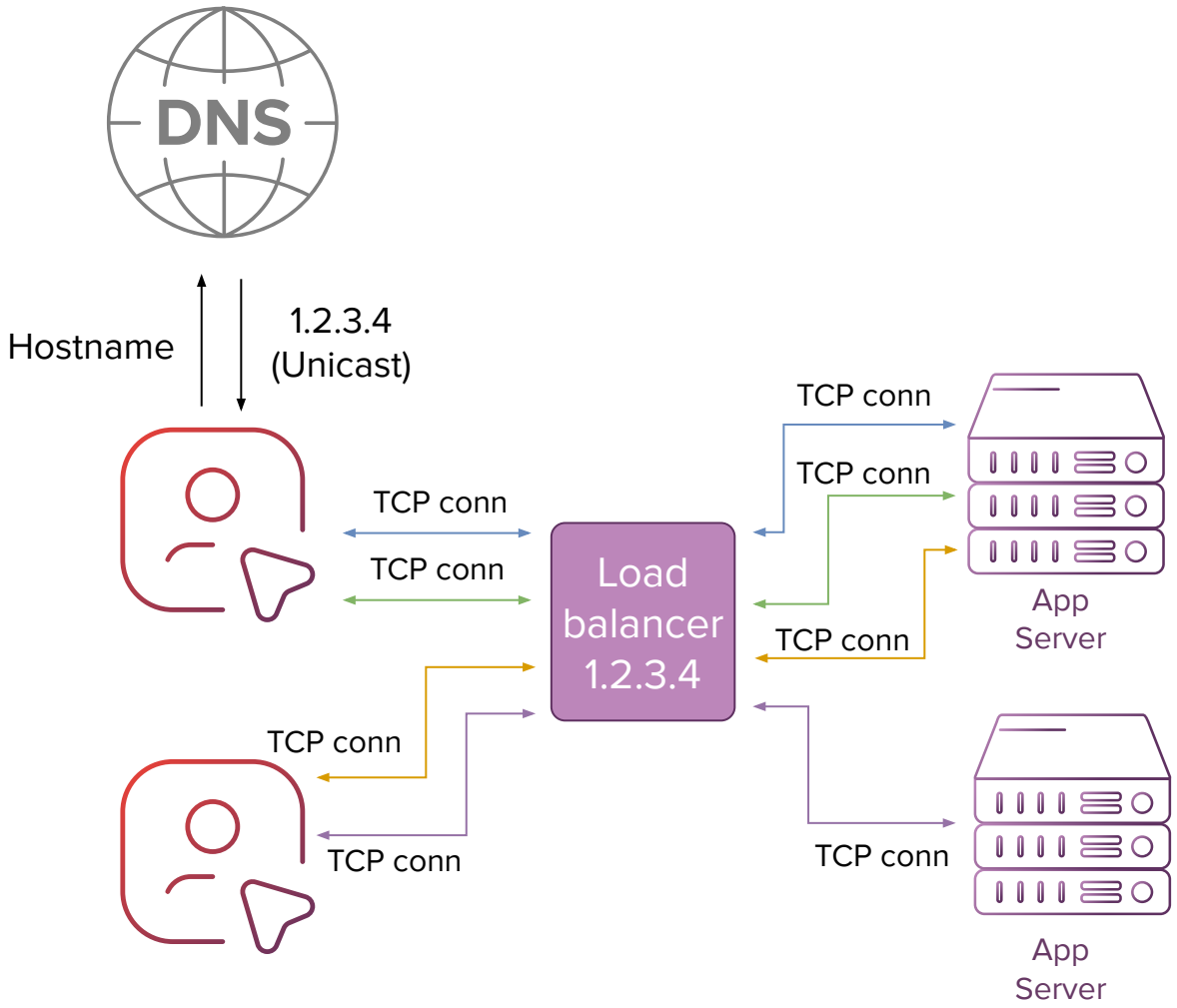
RTB HOUSE

Dedicated load balancers

Hardware load balancers

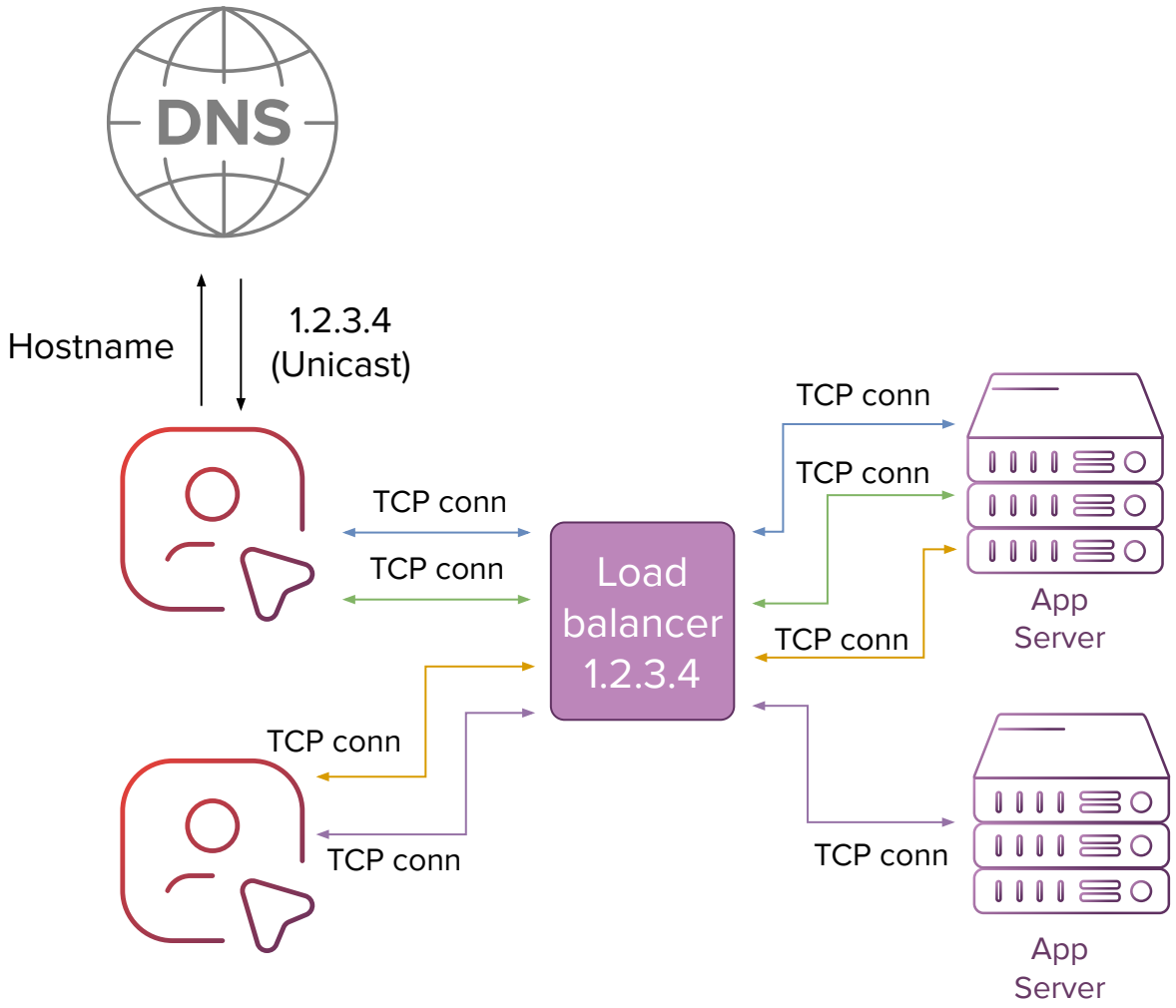
Dedicated load balancers

- Software load balancers - application running on one or more servers
- Can health check application servers and stop sending traffic to unhealthy servers
- Can keep requests from the same client sticky to the same application server
- ... many features
- If running on just one server, single point of failure has moved from the app server to the load balancer - how to deal with this?



Layer 4 load balancing - “proxy”

- Clients establish TCP connections to the LB
- LB establishes TCP connections to backends
- Load balancing algorithm operates at connection establishment time



Layer 4 load balancing - “proxy”

- LB copies data from client connection to associated server connection
- In a plain TCP proxy, the proxy does not parse the TCP stream contents

Layer 4 load balancing - “proxy”

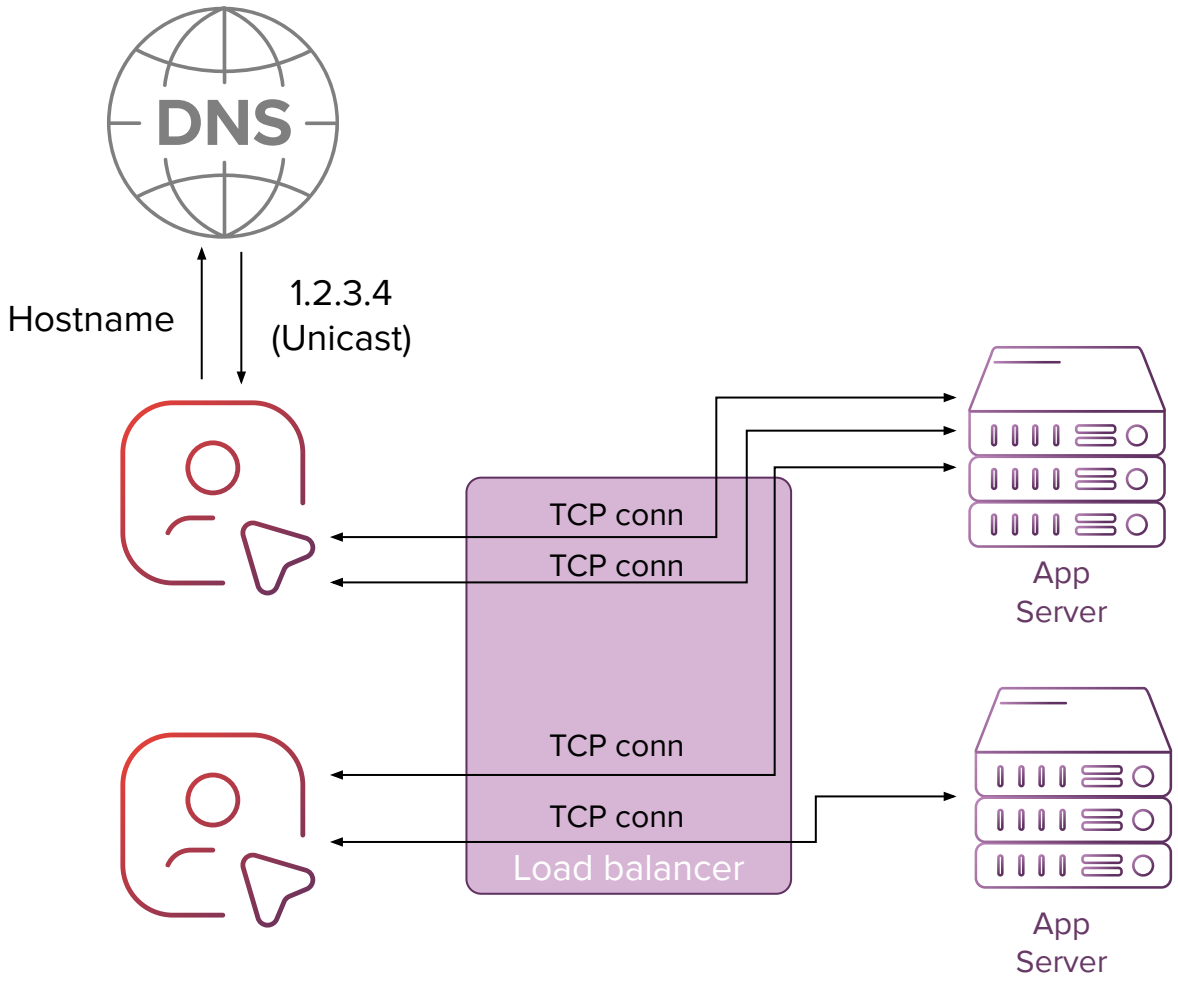
Examples:

AWS: Network Load Balancer (Elastic Load Balancing)
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Google Cloud: TCP Proxy
https://cloud.google.com/load-balancing/docs/tcp

HAProxy (one of possible load balancing modes)
https://www.haproxy.org/

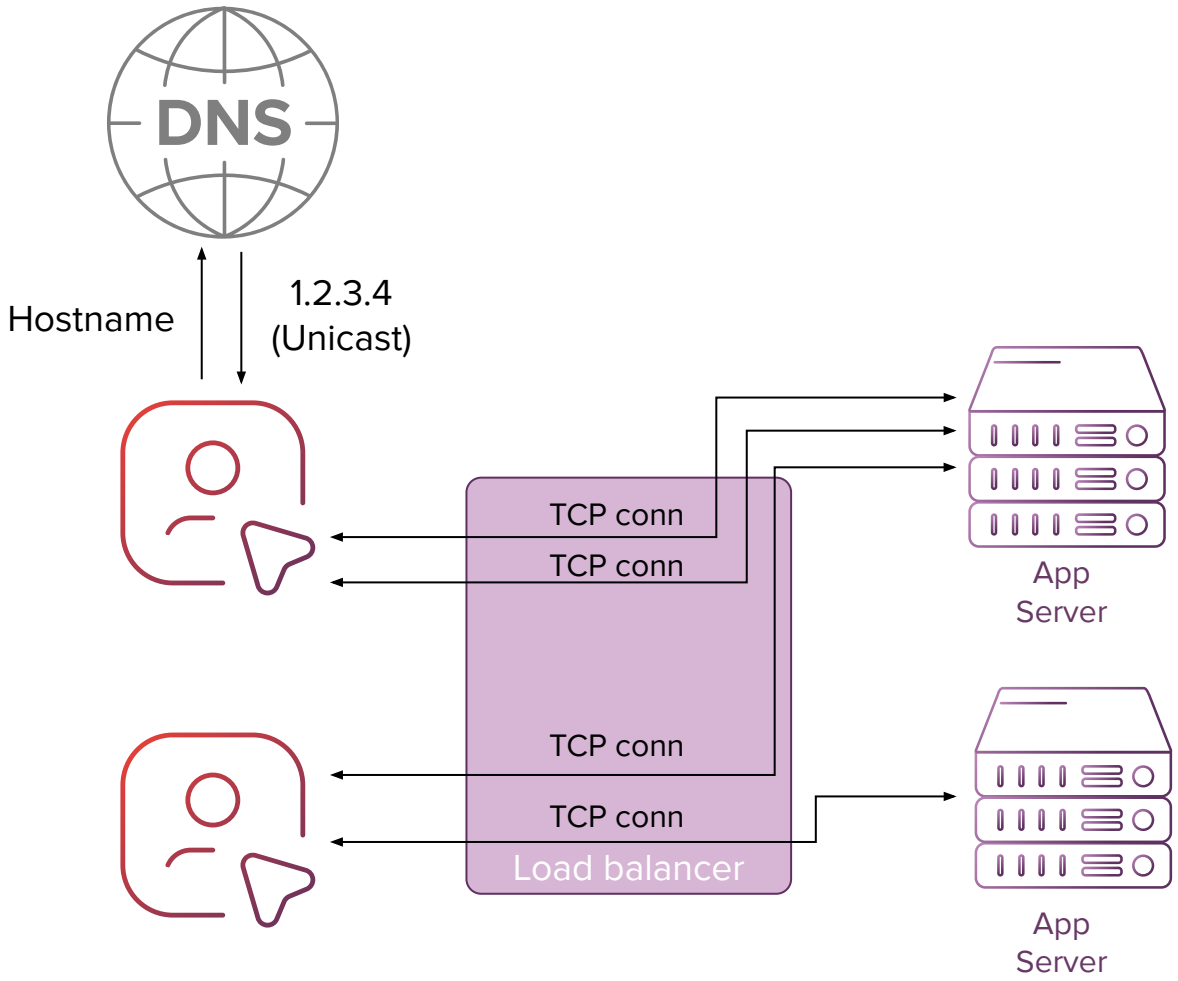
Nginx (one of possible load balancing modes)
https://nginx.org/en/docs/



Layer 4 load balancing – “pass-through”

Variant A:

- 1.2.3.4 is the IP of the load balancer
- Load balancer performs NAT, rewriting the destination IP to a selected app server IP
- Needs to keep a table of client<->server associations



Layer 4 load balancing – “pass-through”

Variant B:

- 1.2.3.4 is an anycast IP, used by all the app servers (on loopback interface)
- Load balancer must be in charge of routing 1.2.3.4
- Load balancer maps (client IP, client port, server IP, server port, TCP/UDP) to specific app server via a hash function

Layer 4 load balancing - “pass-through”

Examples:

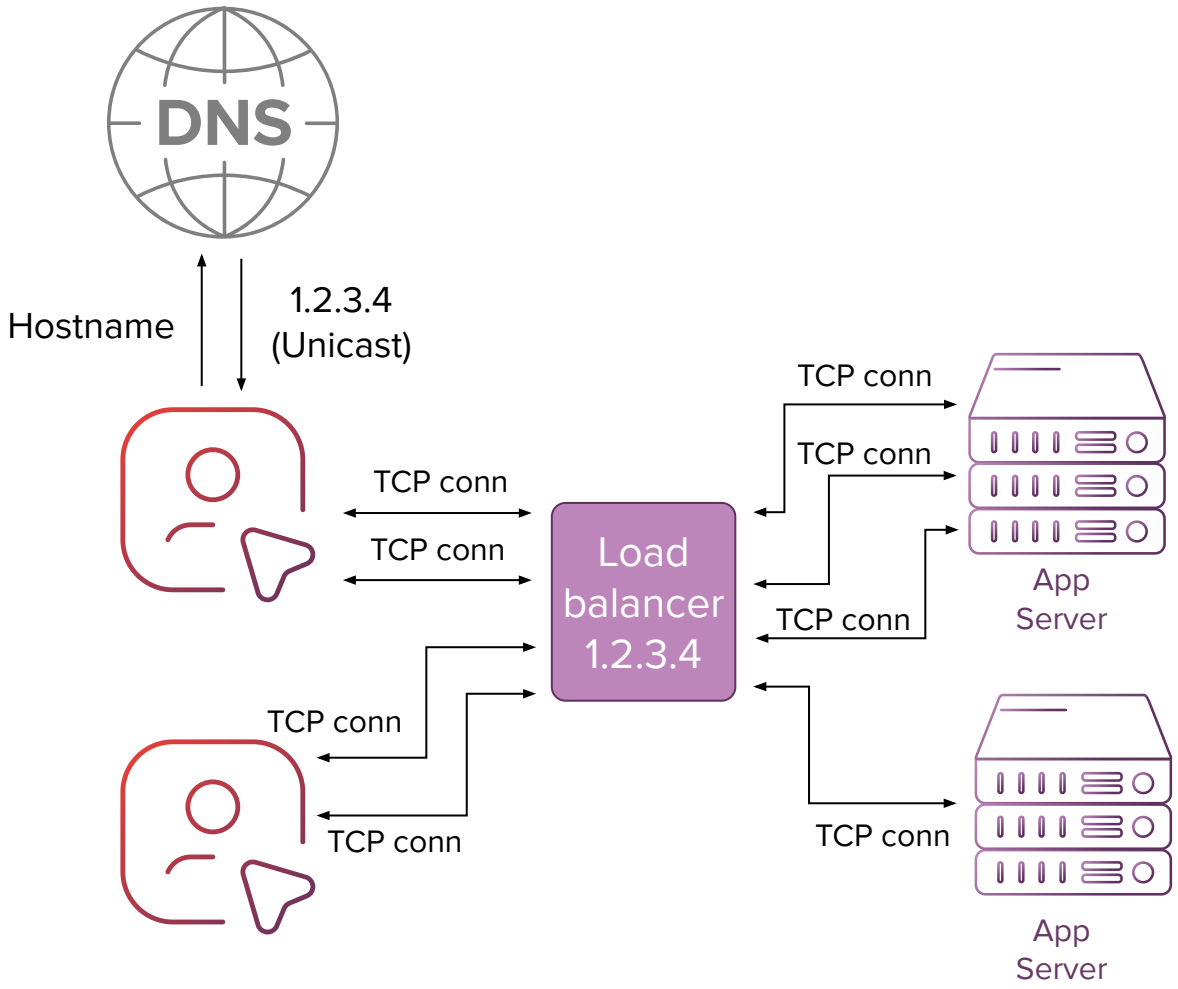
AWS: Gateway Load Balancer (Elastic Load Balancing)
https://docs.aws.amazon.com/elasticloadbalancing/latest/gateway/introduction.html

Google Cloud: External TCP/UDP Network Load Balancing
https://cloud.google.com/load-balancing/docs/network

Google Cloud: Internal TCP/UDP Network Load Balancing
https://cloud.google.com/load-balancing/docs/internal

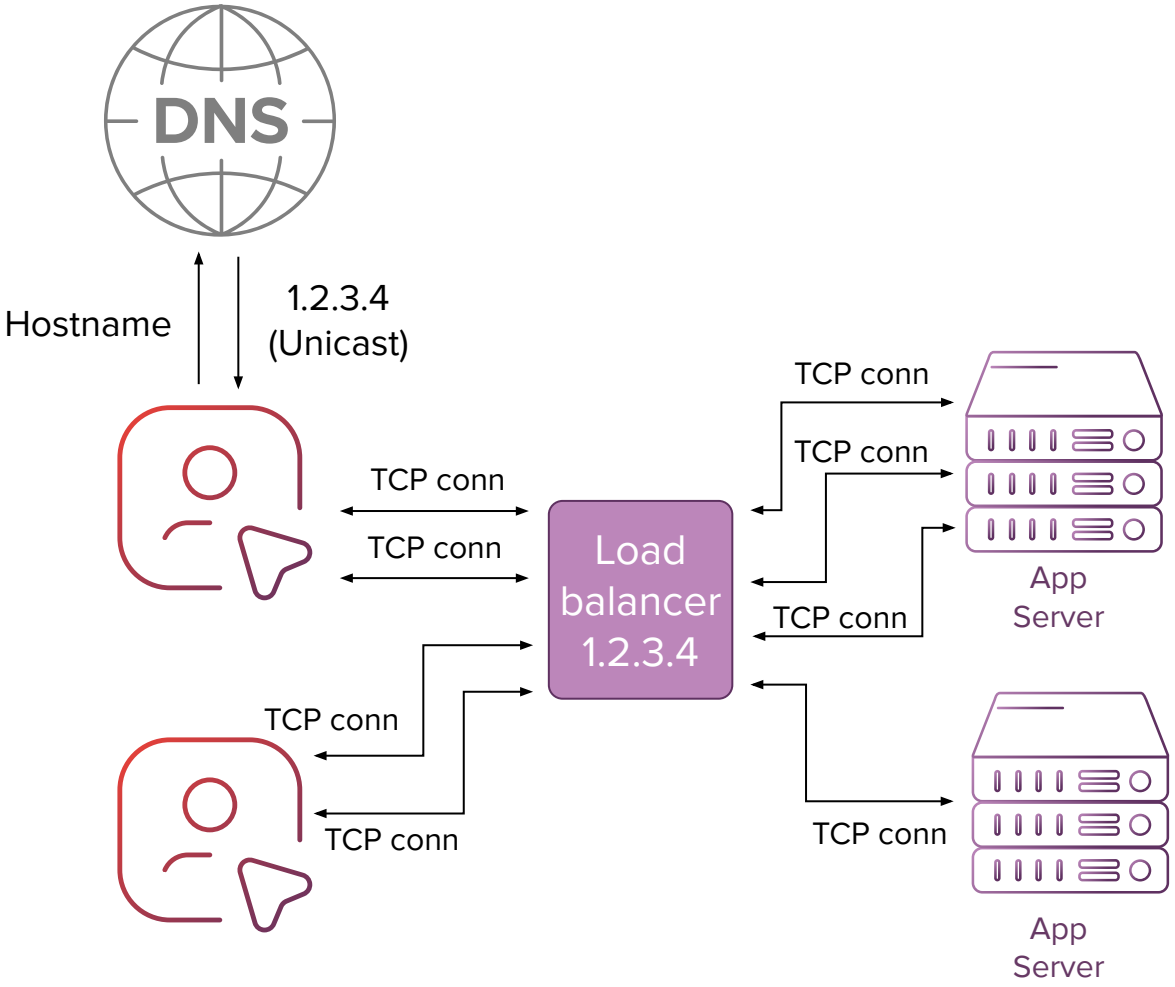
Kubernetes kube-proxy load balancing:
https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/

Linux Virtual Server:
http://www.linuxvirtualserver.org/



Layer 7 load balancing – “reverse proxy”

- Clients establish TCP connections to the LB
- LB establishes TCP connections to app server
- No permanent association between:
  client<->LB conn.
  &
  LB<->app-serer conn.



Layer 7 load balancing – “reverse proxy”

- Load balancing algorithm operates when a (part of) request arrives over the client<->LB TCP connection
- LB parses the L7 protocol operating over TCP stream,
  can make decisions based on HTTP request method, path etc.

Layer 7 load balancing - “reverse proxy”

Examples:

AWS: Application Load Balancer (Elastic Load Balancing)
https://aws.amazon.com/elasticloadbalancing/application-load-balancer/?nc=sn&loc=2&dn=2

Google Cloud: External HTTP(S) Load Balancing
https://cloud.google.com/load-balancing/docs/https

Google Cloud: Internal HTTP(S) Load Balancing
https://cloud.google.com/load-balancing/docs/l7-internal

HAProxy (one of possible load balancing modes)
https://www.haproxy.org/

Nginx (one of possible load balancing modes)
https://nginx.org/en/docs/

Layer 7 load balancing - “reverse proxy”

Examples:

Envoy
https://www.envoyproxy.io/

Traefik
https://traefik.io/

RTB HOUSE

Server side layer 7 load balancing

How to select a server for the incoming request?
Things to consider:

Requests can take varying amount of time to process

Servers can differ in performance

Server side layer 7 load balancing

Weighted round robin algorithm:

Servers “take turns”
handling requests

Server side layer 7 load balancing

Weighted least connections algorithm:

Select the server that has the least number of active connections

Least number of active connections == least requests “in progress”

A request that takes longer to process will also longer contribute to the number of active connections

Server side layer 7 load balancing

Consistent hashing algorithm:

Hash the client IP onto
one of the servers

Server side layer 7 load balancing

Features: TLS termination

Often the layer 7 proxy terminates TLS: connections from client to proxy are encrypted HTTP/1.1 or HTTP/2.0 connections, connections from proxy to backends are unencrypted HTTP/1.1 connections

TLS handling is complicated and might require shared state, layer 7 proxies are typically better at handling it than application servers and there are fewer proxies than application servers

Server side layer 7 load balancing

Features: HTTP routing

Since a layer 7 proxy parses HTTP contents
it can decide which server to use based
on request method, request path,
headers, client IP etc.

RTB HOUSE

Server side layer 7 load balancing

Features: rate limiting

Layer 7 proxies can rate limit connections/s or requests/s to protect from DoS attacks or to provide user quotas etc.

RTB HOUSE

Server side layer 7 load balancing

Features: health checking

We do not want to send requests to backends that will not be able to service client requests correctly

Two (not exclusive) ways to healthcheck: **active** and **passive**

RTB HOUSE

Server side layer 7 load balancing

Active health checking:

Send HTTP request every X seconds

Depending on the response, server is marked healthy or unhealthy

Server side layer 7 load balancing

Passive health checking:

On the TCP level: when enough connection attempt fails, consider the server unhealthy, stop directing traffic to it

On HTTP level: when enough HTTP requests fail, consider the server unhealthy

Server gets healthy again when active health check passes

Server side layer 7 load balancing

Features: service discovery

Need to have a list of available application servers

Naive solution: just have a list of IP addresses in configuration file

Problem: hard to add/remove programmatically from configuration file

Problem: configuration file reload often requires proxy restart (which terminates connections), frequent restarts might destabilize the DC

Server side layer 7 load balancing

Features: service discovery

Example of a better solution:
have the proxy resolve a DNS name to get a list of servers

DNS supports SRV records:
`_service._proto.name. ttl IN SRV priority` **`weight port target`**

Proxy can periodically poll DNS, refresh the SRV records and update the server list without terminating client connections

Platform administrators can add/remove/reconfigure servers by doing DNS updates

Server side layer 7 load balancing

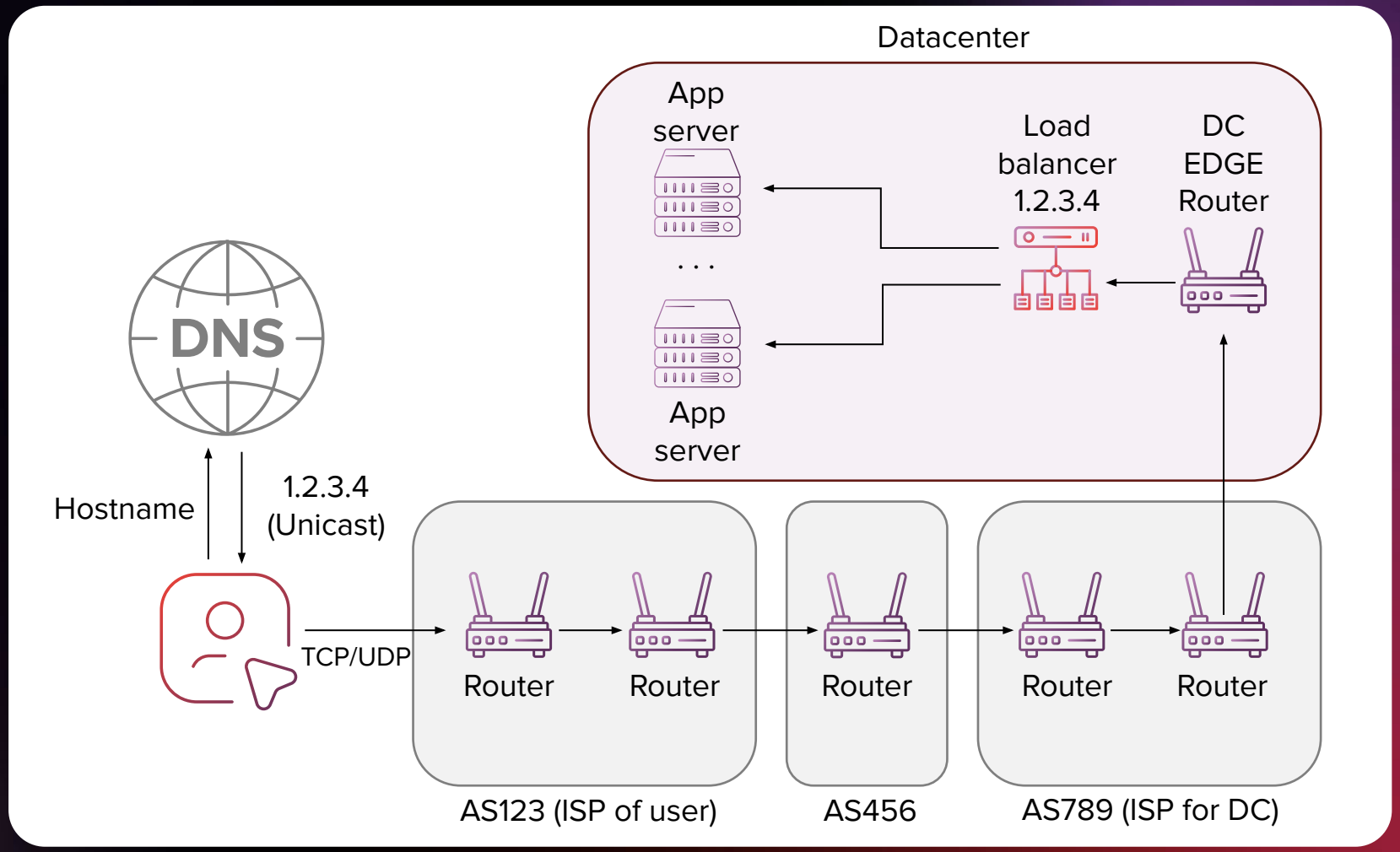
Having a single layer 7 load balancer has two major shortcomings:

Performance ceiling: eventually a single load balancer will saturate and will not be able to serve any more traffic

Single point of failure: if the load balancer fails, the service will become unavailable

RTB HOUSE

Server side layer 7 load balancing



IP Anycast to the rescue

IP protocol provides four adressing modes:

Unicast	Broadcast	Multicast	**Anycast**
delivers a message to a single specific node	delivers a message to all nodes in the network using a one-to-all association	delivers a message to a group of nodes that have expressed interest in receiving the message	delivers a message to any one out of a group of nodes, typically the one nearest to the source

RTB HOUSE

IP Anycast to the rescue

Anycast can be implemented via Border Gateway Protocol (BGP)

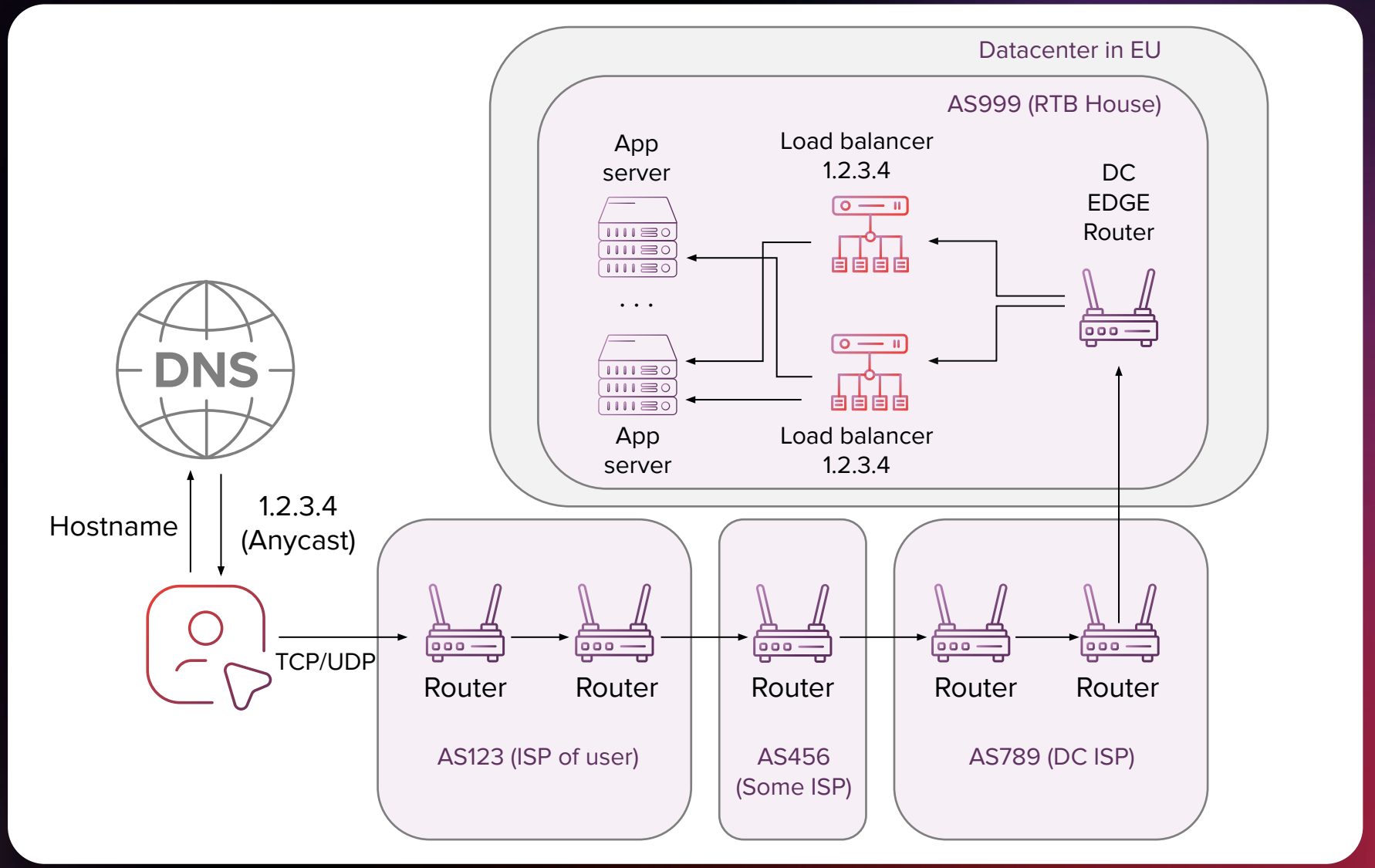
Multiple hosts are given the same unicast IP address and different routes to the address are announced through BGP.

Routers consider these to be alternative routes to the same destination, even though they are actually routes to different destinations with the same address.

As usual, routers select a route by whatever distance metric is in use (the least cost, least congested, shortest). Selecting a route in this setup amounts to selecting a destination.

RTB HOUSE

Anycast-based inside-DC load balancing



RTB HOUSE

Anycast-based inside-DC load balancing

Edge router supports

ECMP: Equal Cost Multipath

When multiple routes are available for an IP, hash different TCP/UDP flows to different available routes

Flows are identified by the four tuple:
(source IP, source port, destination IP, destination port)

Global load balancing

Problem:

For two given geographic locations
of client and server,
Round trip time has a physical lower limit
given by speed of light

Global load balancing

```
Speed of light in vacuum:           300 000 000 m/s
Speed of light in optical fiber:    200 000 000 m/s
Distance from Warsaw to Amsterdam:    1 100 000 m

1 100 000 m / 200 000 000 m/s = 0.0055 s = 5.5 ms

2 * 5.5 ms = 11 ms best-case round trip
```

RTB HOUSE

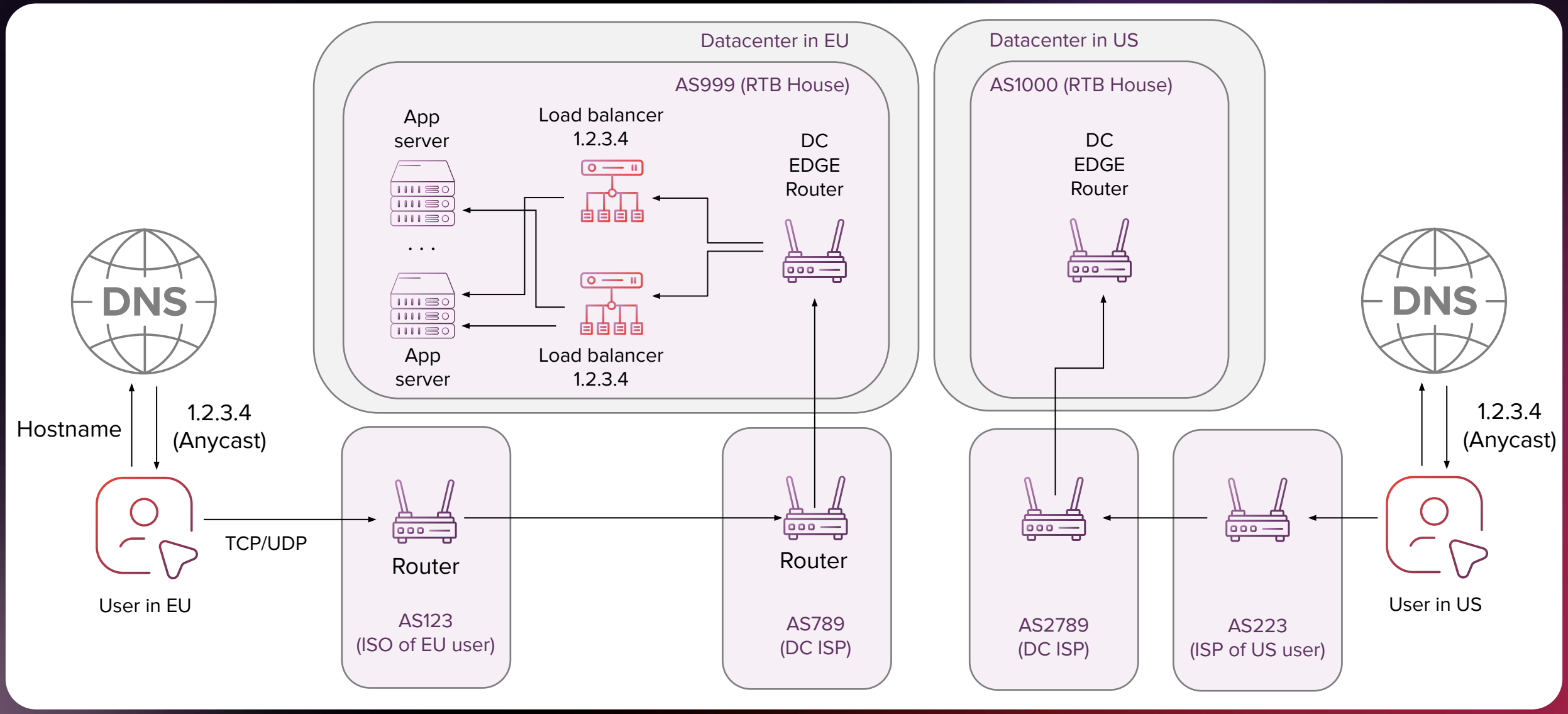
https://tools.bunny.net/latency-test?query=creativecdn.com

Global load balancing

Need to place servers within
reasonable geographic distance
to users

RTB HOUSE

Anycast-based inside-DC load balancing



Anycast-based global load balancing

Available as a cloud service:

AWS: Global Accelerator
https://aws.amazon.com/global-accelerator/

Google Cloud: Global external HTTP(S) load balancer
https://cloud.google.com/load-balancing/docs/https/

RTB HOUSE

Thank you.

Jarosław Rzeszótko