Keeping the Balance
Load balancing Demystified

Murali Suriar
linux.conf.au 2019
Content co-written with Laura Nolan
Murali Suriar

Lapsed computer scientist, turned network engineer, turned network SRE, turned storage SRE.

Some years at Google, with some sailing in the middle.

Laura Nolan

Software engineer, SRE, network SRE.

Worked on Google’s edge network.

Also some pre-Google experience in the ‘real world’.
Why talk about loadbalancing?

- LB failures are often dropped requests
- It's always in your serving path
- Huge impact on the performance and resiliency of your application
  - For better or for worse
Edge routers advertise 203.0.113.0/24 to the Internet via BGP.

DNS

superbowls.com -> 203.0.113.20
<table>
<thead>
<tr>
<th>Load distribution</th>
<th>Distributing load across multiple pieces of infrastructure</th>
<th>Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Partial</td>
</tr>
</tbody>
</table>
Edge routers advertise 203.0.113.0/24 to the Internet via BGP.
Edge routers advertise 203.0.113.0/24 to the Internet via BGP.

Cached: Superbowls.com → 203.0.113.21

Superbowls.com → 203.0.113.20

DNS
Aside: TTL tradeoffs
DNS TTL tradeoffs

- Long TTLs:
  - Many of your users will not see any change you make for a long period of time

- Very short TTLs:
  - Higher load on DNS infrastructure
  - Clients have to query DNS more often - adds latency
  - If DNS experiences any unavailability, a higher proportion of your users will be affected
  - Many clients will ignore very short TTLs anyway
Back to our story
<table>
<thead>
<tr>
<th>Load distribution</th>
<th>Distributing load across multiple pieces of infrastructure</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Poor</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Delayed</td>
</tr>
</tbody>
</table>
Superbowlsw.com \(\rightarrow 203.0.113.200\)

Edge routers advertise 203.0.113.0/24 to the Internet via BGP

192.168.0.20

192.168.0.21
Source address
Source port
Destination address
Destination port
Protocol

Hash of 5-tuple

Selected backend
Superbowl.com → 203.0.113.200

Edge routers advertise 203.0.113.0/24 to the Internet via BGP

203.0.113.200

192.168.0.20

192.168.0.21
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load distribution</td>
<td>Distributing load across multiple pieces of infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Edge routers advertise 203.0.113.0/24 to the Internet via BGP.
Aside: network load balancing
Stateless network load distribution
Stateless network load distribution

- Availability
Stateless network load distribution

- Availability
Stateless network load distribution

- Availability
Stateless network load distribution
Stateless network load distribution

- Capacity
Stateless network load distribution

- Capacity
Network Load Balancing

[Diagram showing load balancing between 203.0.113.200 and 192.168.0.20, 192.168.0.21]
Network Load Balancing - Proxy

- Inbound and outbound traffic through load balancer.
- Requires state in load balancer
- LB backends can be anywhere in your network.

[Diagram showing load balancer connecting to backends 192.168.0.20 and 203.0.113.200]
Network Load Balancing - DSR

- Direct service, er return
- Inbound path through load balancer
- Outbound path direct, bypassing load balancer
Network Load Balancing - L2DSR

- Load balancer and all backends on the same (layer 2, Ethernet) network.
- Service VIP is still .200.
Network Load Balancing - L3DSR

- Load balancer and all backends on the different networks.
- Service VIP is still .200.
Network Load Balancing - L3DSR

- Internet → loadbalancer (black)
  - Src IP: <user public IP>
  - Dst IP 203.0.113.200 (VIP)
- (MAC addresses not relevant this time)
Network Load Balancing - L3DSR

- Loadbalancer → backend (red)
  - Src IP: <load balancer private IP>
  - Dst IP: 192.168.2.20
  - <Encap header> (GRE/IP-IP)
  - Src IP: <user public IP>
  - Dst IP 203.0.113.200 (VIP)
- Request IP header preserved.
- Backends need to be able to decapsulate.
- Careful about MTU!
Network Load Balancing - L2DSR

- Loadbalancer → backend (blue)
  - Src IP 203.0.113.200 (VIP)
  - Dst IP: <user public IP>
Back to our story
Anycast

- It’s not loadbalancing.
- What is it?
  - Same address, multiple locations.
  - Network decides where to route each packet.
  - No concept of balancing; still just load distribution
- Caveats
  - Monitoring is hard
  - Capacity planning is hard
  - Cascading failure is easy.
- See Murali’s previous talk at SRECon EMEA 2017
Geo-aware DNS

superbowls.com -> 203.0.113.200, 198.51.100.200
Aside: the perils of DNS geo loadbalancing
Problems with geographic balancing

- Internet addressing scheme wasn’t designed to support this
- Blocks of addresses move
- Recursive resolution: the source IP that your DNS sees may not be close to the end user
- Inevitably involves a lot of messing about configuring exceptions or cleaning data - toil
EDNS0 extension: client subnet

- Extends DNS with information about the network that originated a query
- Also lets the authoritative nameserver specify the network that the response is intended for
- Implemented by OpenDNS and Google Public DNS
Back to our story
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load distribution</td>
<td>Distributing load across multiple pieces of infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Yes</td>
</tr>
<tr>
<td>Geo awareness</td>
<td>Systems serve from the best location for users (less latency)</td>
<td>Yes</td>
</tr>
<tr>
<td>Load aware</td>
<td>Can balance lightweight and heavyweight loads effectively</td>
<td>No</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Answer</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Load distribution</td>
<td>Distributing load across multiple pieces of infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Yes</td>
</tr>
<tr>
<td>Geo awareness</td>
<td>Systems serve from the best location for users (less latency)</td>
<td>Yes</td>
</tr>
<tr>
<td>Load aware</td>
<td>Can balance lightweight and heavyweight loads effectively</td>
<td>No</td>
</tr>
<tr>
<td>Content-based</td>
<td>Can perform load distribution based on the content of the request (e.g. cookies)</td>
<td>No</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Status</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Load distribution</td>
<td>Distributing load across multiple pieces of infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Yes</td>
</tr>
<tr>
<td>Geo awareness</td>
<td>Systems serve from the best location for users (less latency)</td>
<td>Yes</td>
</tr>
<tr>
<td>Load aware</td>
<td>Can balance lightweight and heavyweight loads effectively</td>
<td>No</td>
</tr>
<tr>
<td>Content-based</td>
<td>Can perform load distribution based on the content of the request (e.g. cookies)</td>
<td>No</td>
</tr>
<tr>
<td>Policy enforcement</td>
<td>Point to apply DDoS protection, rate limiting and load-shedding</td>
<td>No</td>
</tr>
</tbody>
</table>
Layer 7 load balancing

- AKA application loadbalancing, or a reverse proxy
- Terminates the connection from the user, make requests to one or more backend servers, and then returns responses to the user
- Understands the structure of the request -> only kind of balancers that can distribute load based on a cookie, or a parameter or similar
Edge routers advertise 203.0.113.0/24 to the Internet via BGP.
Layer 7 load balancing - scalability

- Resources will be held on the LBs for the duration of user requests
- A L7 balancer crashing will be seen by users
  - L4 can often fail transparently
- L7 balancers can retry a request that failed on one of its backends
- Will add more latency to a request than L4 balancers
Layer 7 load balancing - reliability

• Can be load aware
• Rate limiting and load shedding
• Line of defence against application-layer DoS attacks
• Produces much better telemetry than a L4 balancer can
Aside: the cloud
Loadbalancing algorithms

- Balancing in a single pool of backends
  - Stateless hashing
  - Round robin
  - Least-loaded, shortest queue and similar
  - Weighted round robin
  - Probation
  - Choice of 2

- Multiple pools of backends
  - Priority/failover
  - Nearest by location
Clients

Requests

Servers

Server addresses

Load reports

Lookaside loadbalancer
Service Mesh

- Infrastructure layer for service to service communication
- Linkerd, Envoy, Istio, Conduit
- Goal of a service mesh is to make service communication a first-class citizen
  - Service discovery
  - Configurable routing policies
  - Authentication and authorization
  - Monitoring and management of service to service communications, distributed tracing, fault injection etc
  - Consistent point to apply policies on retrying, deadlines etc
Service A
Sidecar

Service B
Sidecar

Control plane

Config data, telemetry etc
Microservices as backends

Service A
Sidecar

Service B
Sidecar

Control plane

Webservice front-ends
The big idea: consistency
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load distribution</td>
<td>Distributing load across multiple pieces of infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>High Availability</td>
<td>Avoid using unhealthy infrastructure</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Allow operators to shift load manually or via configured policies</td>
<td>Yes</td>
</tr>
<tr>
<td>Geo awareness</td>
<td>Systems serve from the best location for users (less latency)</td>
<td>Yes</td>
</tr>
<tr>
<td>Load aware</td>
<td>Can balance lightweight and heavyweight loads effectively</td>
<td>Yes</td>
</tr>
<tr>
<td>Content-based</td>
<td>Can perform load distribution based on the content of the request (e.g. cookies)</td>
<td>Yes</td>
</tr>
<tr>
<td>Policy enforcement</td>
<td>Point to apply DDoS protection, rate limiting and load-shedding</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Takeaways

● What do you want from your systems?
  ○ More capacity? Higher availability? Higher utilisation?
  ○ Finer grained control?
  ○ More instrumentation and monitoring?

● What constraints do you have?
  ○ Do you trust your clients?
  ○ Do you control your whole stack?
Links

- Google’s maglev paper
- Facebook Katran
- HAProxy
- ucarp
- Google SRE Book loadbalancing chapter
- EDNS0 client subnet RFC
- Summary of Facebook’s Billion User Loadbalancing talk
- Google’s GFS and Bigtable papers
- gRPC load balancing
- Istio, Linkerd
  - Monzo talk on using Linkerd + Kubernetes in production
● Loadbalancing has evolved hugely in the last decade.
● What do you want from your systems?
  ○ More capacity? Higher availability? Higher utilisation?
  ○ Finer grained control? More instrumentation and monitoring?
● What constraints do you have?
  ○ Do you trust your clients?
  ○ Do you control all layers of your stack?

See the talk slides for more.