

# Load-Balancing Introduction (with examples...)

For AFNOG 2015  
By Frank Kuse

(Rework of slides from Joel Jaeggli and Laban Mwangi)

# Load-Balancing Introduction (with examples...)

For AFNOG 2015  
By Frank Kuse

(Rework of slides from Joel Jaeggli and Laban Mwangi)

# What is Load-balancing

- The act of dividing a workload between  $N > 1$  devices capable for performing a task.
- Multiple contexts in internet services where this concept occurs.
  - DNS
  - MX records
  - Multiple links (L2 trunks, L3 ECMP)
  - Multiple servers

# Goals

- Greater scalability
  - Horizontal scaling. Just add more switches/servers...
- Higher availability
  - Don't care about single device failure. Route around failures automatically!
- Reduced cost
  - Cheaper to use commodity hardware and architecture for failure. Examples: AWS/GCE...

amaze..



# Quick Survey

- L2
  - LACP (Switches)
- L3
  - L3 ECMP (Switches, Routers, OS kernel)
- L4
  - HAProxy (OS userland)
- L4+
  - NGINX (OS userland)
  - HAProxy (OS userland)
  - F5, A10, Netscaler... (Hardware..)

# Examples: L2 – Link aggregation

- Widespread support for **LACP** (Link Aggregation Control Protocol)
- Bond two physical layer 2 channels into one logical one.
  - Resilience against single port/channel failure.
  - L2 Bandwidth scaling
- Balancing and dynamic behaviour is important!

# Examples: L3 - Equal-cost multi-path routing (ECMP)

- Packets are forwarded to the next hop over links having an equal routing cost.
- Stateless mode breaks TCP (PMTU)
  - Different hops may have different MTU settings
  - TCP sensitive to re-ordering
- We need a way to make flows stateful.....



# Examples: L4 - Equal-cost multi-path routing (ECMP) + hashing

- If packets in a TCP session take the same path...
  - Path MTU issues would be fixed
  - Re-ordering would be fixed
- Different TCP sessions can take different paths.
- We need a way to uniquely identify L4 sessions ....
- What attributes do you think would identify a TCP session?

# Flow identification (5-tuple)

- XOR hash of fields to generate a flow id.
- Hash src & dest ip addresses, protocol number from the IP header and ....

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP				ECN				Total Length															
4	32	Identification																Flags				Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

# 5-tuple continued

- ... hash of port numbers.
- How?
  - Example:  $\text{CRC32}(\text{src\_ip}, \text{dst\_ip}, \text{pr\_no}, \text{src\_port}, \text{dst\_port}) \% \text{count of links}$

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset	Reserved 0 0 0			N S	C W R	E C E	U R G	A C K	P R E	R S S	S S Y	F I N	Window Size																		
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																															
...	...	...																															

# What does an L4 load Balancer do?

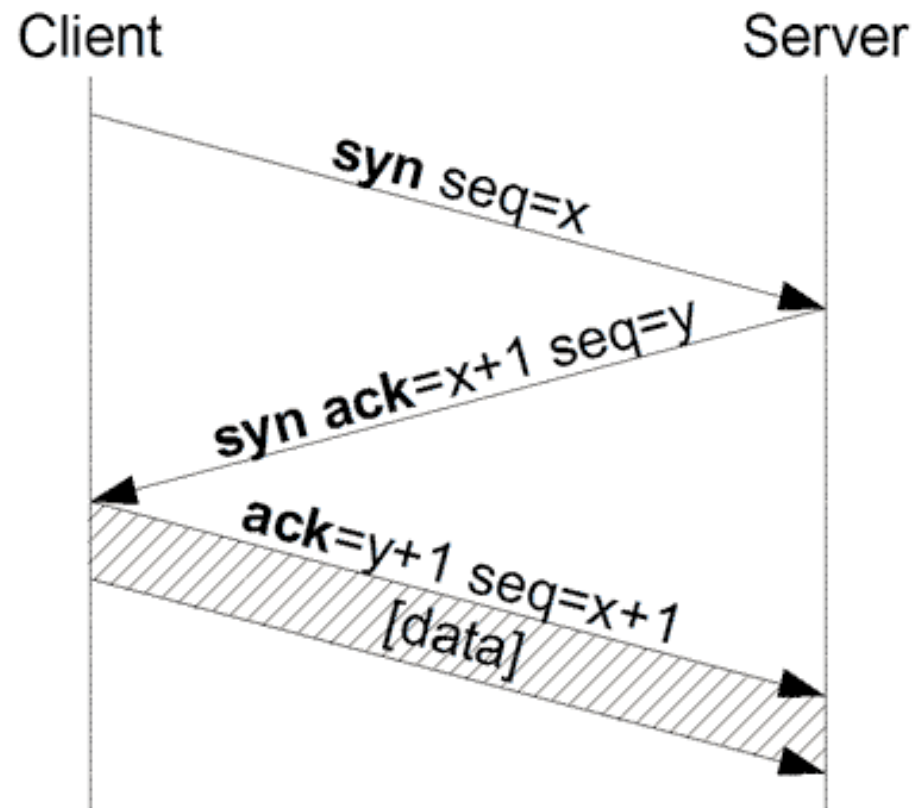
- Looks and the Destination IP and Port to determine which endpoint to send a packet/flow to in a pool of servers.
- Forwards the incoming connection to one pool member on the basis of policy (liveness, load).
- May keep the connection pinned to the particular pool member by tracking the connection.
- But... This breaks scaling!
  - Existing flows won't be remapped dynamically!
  - An LB/server failure would break a session!

# What does an L7 load balancer do?

- An L7 load balancer answers incoming connection requests.
- It understands the protocol being spoken across the connection (e.g. HTTP IMAP FTP etc).
- On the basis of either 5-tuple hash or some higher layer value, (example a URI or a cookie or both) the request is directed to a member of the appropriate pool.
- L7 is another word for proxy or ALG (Application Layer Gateway).

# Isn't L7 going to be slower than L4?

- Probably but not always.
- Importantly there are optimizations that can reduce the expense.
  - TCP syn-cookies
  - Connection pooling
  - Consider 3-way handshake



# Applications - Cont

- Open source
  - Apache mod\_proxy\_balance
  - Squid
  - Haproxy
  - NGNIX
  - LVS

# Applications Commercial

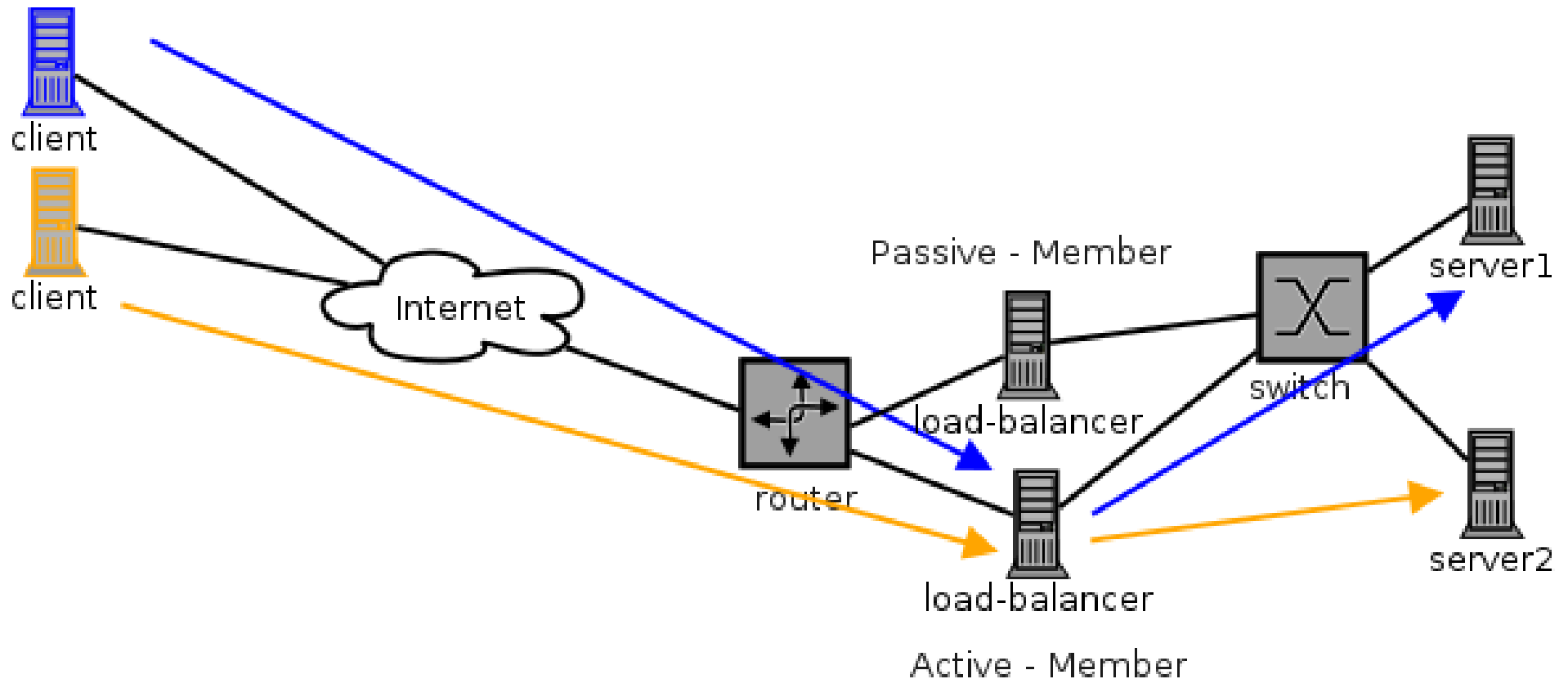
- Commercial
  - F5
  - Netscaler
  - A10
- Benefits of a commercial approach
  - Coordination of supporting elements
    - Routing
    - DNS
    - Complex health checks
    - HA
  - Can have ASIC based acceleration.



# High Availability Approaches

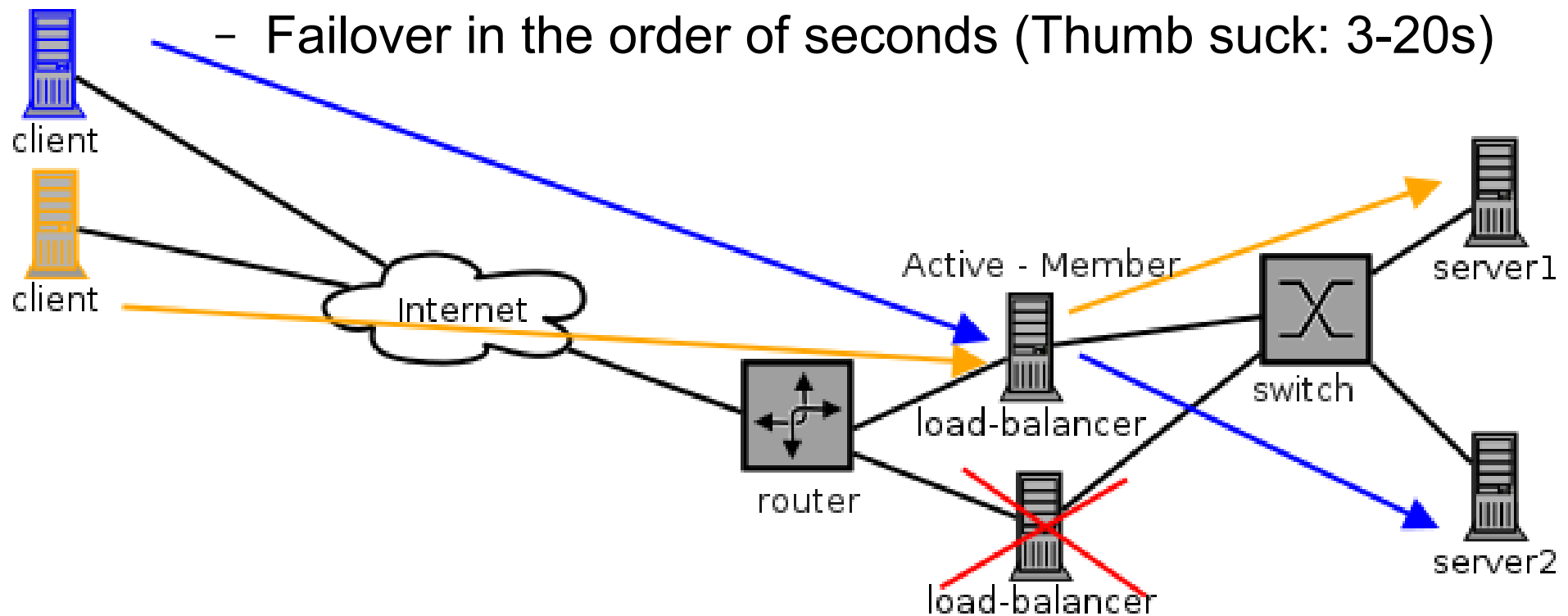
- Active-Passive
  - VRRP
  - State replication
- Active-Active
  - State-replication considerations
- Horizontally scaled
  - GTM – DNS based approach
  - L3ECMP (routed)

# HA – active/passive



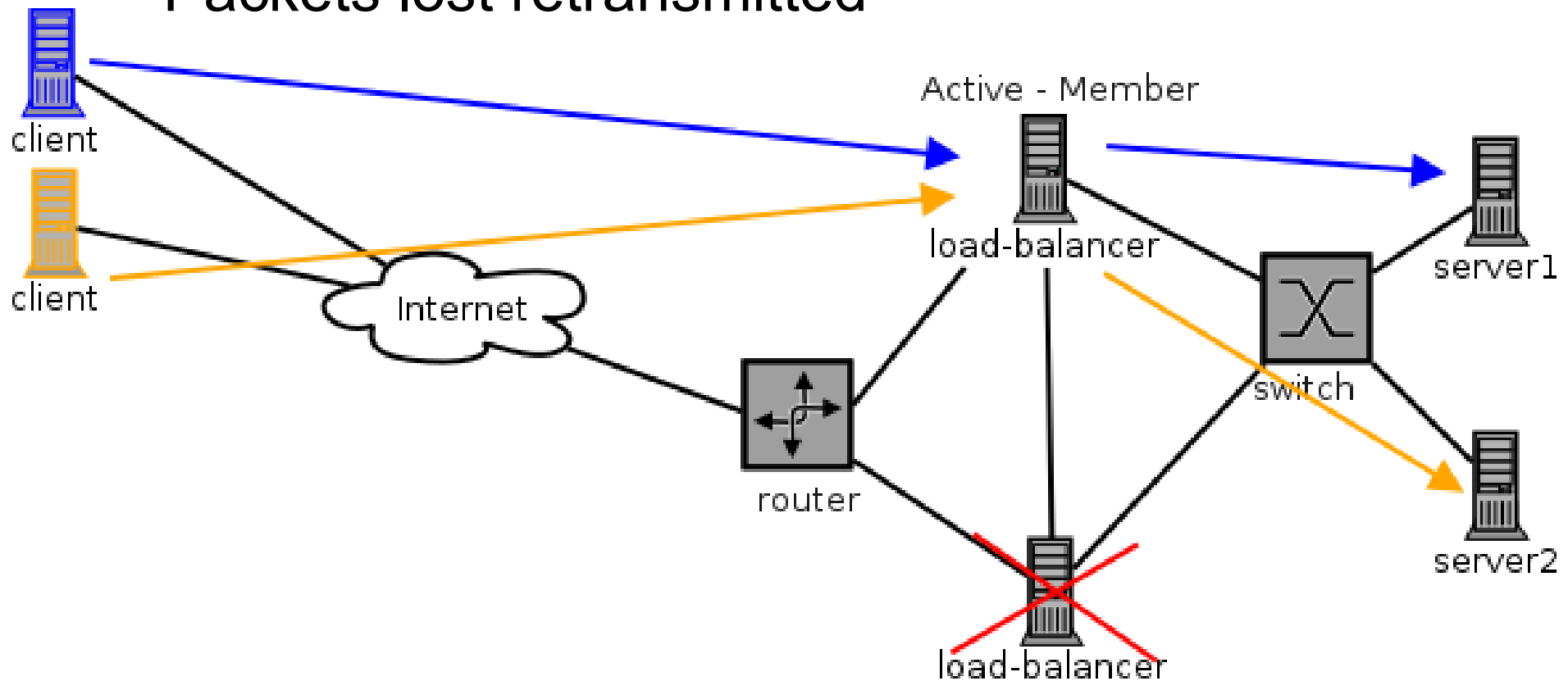
# HA – active/passive - failover

- Connections terminated:
  - Stateless secondary
    - Secondary won't know which server to send packets to
    - TCP sessions will timeout and a new session initiated
    - Failover in the order of seconds (Thumb suck: 3-20s)



# HA – active/passive failover with replication

- Connections work:
  - Secondary knows the hash state
  - Packets lost retransmitted



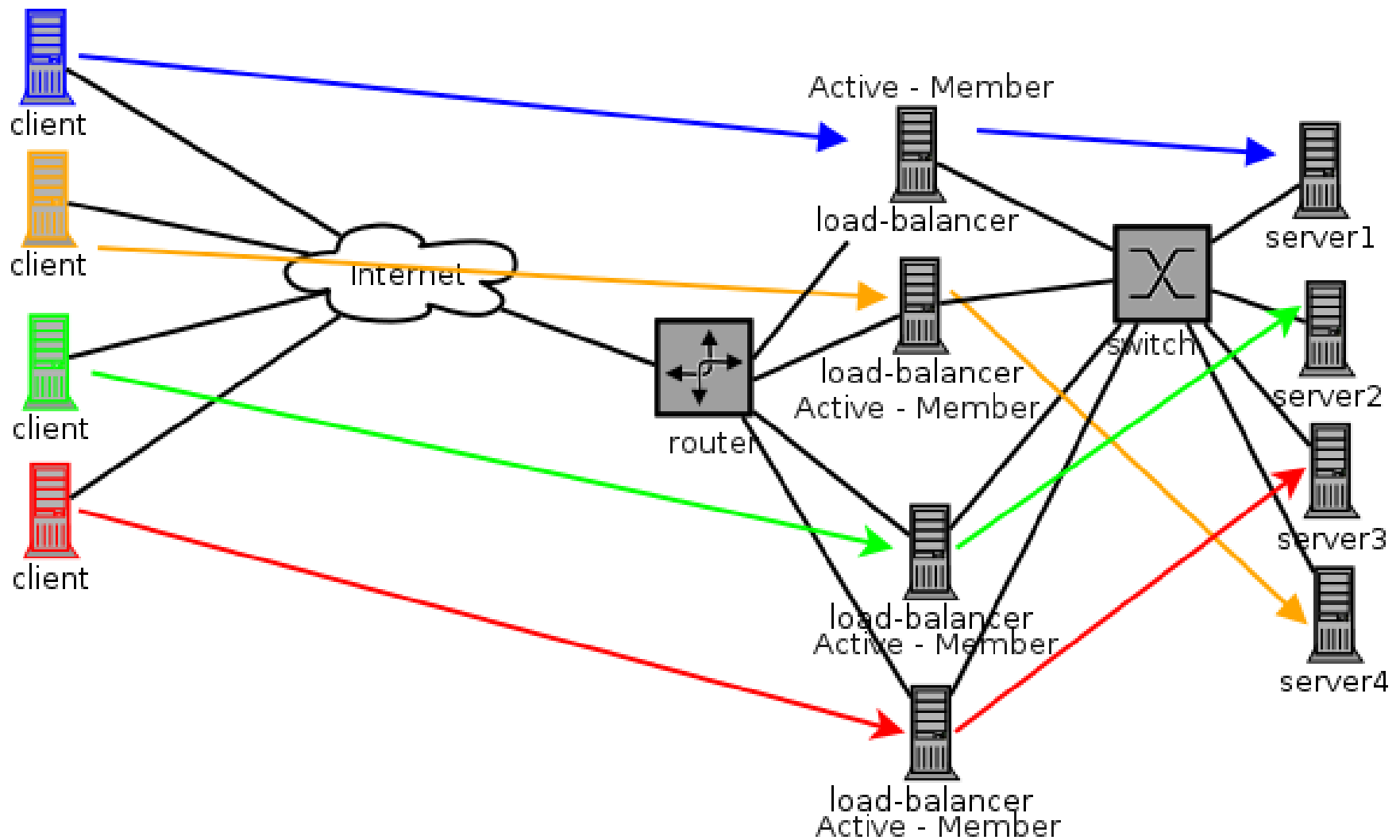
# Active / Passive

- Active-passive failover requires a mechanism
- Could use:
  - VRRP (Virtual Router Redundancy Protocol)
  - CARP (Common Address Redundancy Protocol)
- If failover is not coordinated with load-balancer-health, a failed load-balancer may remain active (coordination problem).
- If state is not replicated between load balancers, failover will not account for existing connections (not a problem for short-lived connections with no affinity)

# Active / Passive Cont

- Affinity can be preserved with a Cookie
- LVS (linux virtual server) can do state-replication (using a kernel module)
- State-replication doesn't help with scaling performance-wise (at all)

# Active/Active



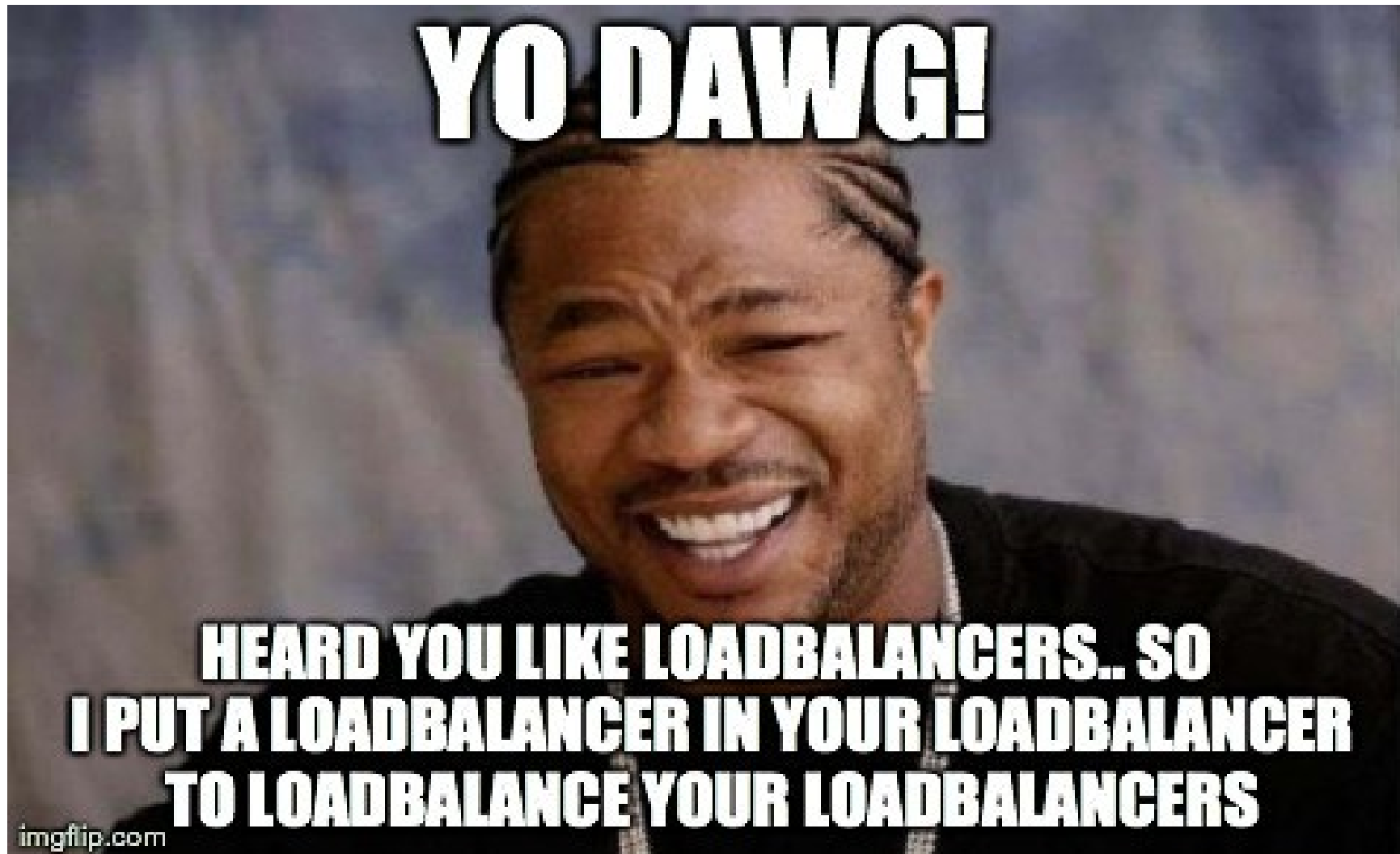
# Active/Active – How?

- Need a mechanism to distribute requests to multiple front end load-balancers. In effect, a load balancer for your load balancers.
- HOW?
  - DNS e.g. each LB has a separate ip address associated with resources it's load-balancing
    - Return one or more resource records either randomly or on some externally instrumented basis.
    - Fail load balancers in or out using health check or manually
  - L2 or L3 stateless plus sticky mechanism.



# Turtles all the way...

- When do we stop?



# Active/Active – Stateful vs Not

- Stateful is typically done by clusters of commercial load-balancers. State replication can be expensive and imperfect.
  - At scale, can be extremely expensive
  - Memory on cluster members and bandwidth/cpu for replication is the limiting factor for state and connections per section.
- Stateless
  - In the DNS case resource records for a failed LB have to time out of caches before that LB stops being used.
  - In the L3-ECMP case a failure will cause some fraction of connections to rehash across other load-balancers anywhere from a quarter to half (they will then be rendered out of state and lost).

# Our Exercise - HAProxy

- We're going to deploy HAProxy to load-balance connections to two http servers.
- HAProxy can do L4 (any TCP) or L7 (HTTP) load balancing
- We're going to do L7, this allows us to access http related features, including for example including a cookie.

# HAProxy vs NGINX

- L4 vs L7
- HAProxy can load balance anything over TCP or do L7.
- NGINX is L7 only (HTTP(s) and IMAP/POP3).
- SSL
  - HAProxy doesn't support (can't only treat as TCP)
  - NGINX does, so cookies for example can be parsed, can be used for SSL offload etc.
- Model
  - HAProxy is threaded, effectively allowing it to engage multiple cpus in the activity.
  - NGINX uses an event driven single threaded model.
  - Both have merit, HAProxy is probably more scalable.

# Goals

- 1) Install and perform a basic configuration of HAProxy.
- 2) Configure two additional webserver instances on alternate ports.
- 3) Demonstrate load-balanced-http connections between them.
- 4) Log X-Forwarded-For.
- 5) Bonus: use a cookie to pin a requesting host to one server or another.
- 6) Bonus: Remove failing servers from LB pool.

# Exercise Details (1)

1) Install HAProxy using commands below

```
sudo apt-get install haproxy
```

2) Get your secondary IP address from your VM using commands below.

```
sudo ip addr ls
```

The detail instruction for getting the address on on the next slide.

# Exercise Details (1)

1) Install HAProxy using commands below

```
sudo apt-get install haproxy
```

2) Get your secondary IP address from your VM using commands below.

```
sudo ip addr ls
```

The detail instruction for getting the address on on the next slide.

# Exercise Details (2)

For the primary server running the Haproxy application needs to make use of the secondary address on the block which can be gotten from commands below.

```
afnog@pc38:~$ sudo ip addr ls
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
1621: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:12:35:00 brd ff:ff:ff:ff:ff:ff
    inet 197.4.11.138/24 brd 197.4.11.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet 197.4.11.188/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 2001:43f8:220:219:5054:5bff:feaa:d69f/64 scope global mngtmpaddr dynamic
        valid_lft 2591996sec preferred_lft 604796sec
    inet6 fe80::5054:5bff:feaa:d69f/64 scope link
        valid_lft forever preferred_lft forever
```



# Exercise Details (3)

The HAProxy configuration file should be configured as below. It can be located on the file system by opening the haproxy.cfg in the /etc/haproxy directory

```
global
#       log /dev/log      local0
#       log /dev/log      local1 notice
log 127.0.0.1    local2
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin
stats timeout 30s
user haproxy
group haproxy
daemon

# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
```

# Exercise Details (4)

```
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:
AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
ssl-default-bind-options no-sslv3
```

defaults

```
log global
mode http
option httplog
option dontlognull
option http-server-close
option forwardfor except 127.0.0.0/8
option redispatch
retries 3
timeout http-request 20
timeout queue 86400
timeout connect 86400
timeout client 86400
timeout server 86400
timeout http-keep-alive 30
timeout check 20
maxconn 50000
```

# Exercise Details (5)

```
frontend LB
  bind 197.4.11.188:80
  reqadd X-Forwarded-Proto:\ http
  default_backend LB

backend LB 197.4.11.188:80
  mode http
  stats enable
  stats hide-version
  stats uri /stats
  stats realm Haproxy\ Statistics
  stats auth afnog:afnog          # Credentials for HAProxy Statistic report page.
  balance roundrobin            # Load balancing will work in round-robin process.
  option httpchk
  option httpclose
  option forwardfor
  cookie LB insert
  server pc36.sse.ws.afnog.org 197.4.11.136:80 cookie pc36.sse.ws.afnog.org check
  server pc37.sse.ws.afnog.org 197.4.11.137:80 cookie pc37.sse.ws.afnog.org check
```

# Exercise Details (6)

Change the configuration of the syslog to help in debugging purposes

```
sudo vi /etc/rsyslog.conf
```

Make sure you remove comment on the following port 514 and module as shown below.

```
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

Restart your syslog service as well as haproxy service as below.

```
sudo service rsyslog restart
```

```
sudo service haproxy restart
```

# Exercise Details (7)

Create an html file for testing purposes with below content inside the web servers that you wish to load balance

For our example, we are going to create an index2.html file with content on next slide.

```
sudo nano /var/www/html/index2.html
```

Repeat the same for all the other web server that you wish to help load balance your apache page.

# Exercise Details (8)

```
<html>
<head>
  <title>Afnog 2015 HAProxy Test Page</title>
</head>

<body>
<!-- Main content -->
<h1>My Afnog HAProxy Test Page</h1>

<p>Welcome to our Afnog HA Proxy test page!

<p> Welcome to this year's Afnog tutorials
We hope you get answers to most of your queries :p.

<address>Made 27 May 2015<br>
  by Frank Kuse .</address>

</body>
</html>
```

# Exercise Details (9)

Try accessing the created content via the link below

<http://pc38.sse.ws.afnog.org/index2.html>

Check the message logs to see which web server is serving the content by running the below command

```
sudo tail -f /var/log/haproxy.log
```

Try accessing the stats page on the haproxy server with the following credentials to see the statistic of your haproxy performance.

[Http://pc38.sse.ws.afnog.org//stats](http://pc38.sse.ws.afnog.org//stats)

Username: afnog

password: afnog

# Bibliography

- HAProxy - <http://haproxy.1wt.eu/>
- NGINX - <http://wiki.nginx.org/Main>
- F5 LTM - <http://www.f5.com/products/big-ip/local-traffic-manager.html>
- A10 Networks - <http://www.a10networks.com/>
- Apache mod\_proxy\_balance -  
[http://httpd.apache.org/docs/2.2/mod/mod\\_proxy\\_balancer.html](http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html)
- Linux virtual server - <http://www.linuxvirtualserver.org/index.html>
- Wikipedia CARP -  
[http://en.wikipedia.org/wiki/Common\\_Address\\_Redundancy\\_Protocol](http://en.wikipedia.org/wiki/Common_Address_Redundancy_Protocol)
- Wikipedia VRRP -  
[http://en.wikipedia.org/wiki/Virtual\\_Router\\_Redundancy\\_Protocol](http://en.wikipedia.org/wiki/Virtual_Router_Redundancy_Protocol)