# High Availability Low Dollar Load Balancing

Simon Karpen

System Architect, VoiceThread

skarpen@voicethread.com

Via Karpen Internet Systems

skarpen@karpeninternet.com

# Overview

- What is Load Balancing
- Why load balance
- What services should you load balance
- What are some common load balancing topologies
- What are some open source load balancing technologies
- How would we build a HA configuration out of these technologies
- How do I IPv6 enable IPv4 services with a single command line on a dual-stack machine

# What is Load Balancing

- Split traffic across two or more servers
- Many different techniques and topologies
- Layer 4 or layer 7
- Useful for most TCP services
- Divides traffic using a variety of algorithms (WLC, RR, etc)

# Why Load Balance

- Improve performance
- Improve redundancy
- More cost effective scaling
  - 4-socket machines cost 4x as much as 2-socket
- More cost effective redundancy
  - n+1 or n+2 instead of 2n
- SSL Acceleration
- Security / IPS / Choke Point

# Which Services

- Without built-in failover
- More than one infrastructure unit of performance
- Good: web services, application services
- Probably not: DNS, inbound SMTP
- Examples: virtually any web site you visit!
- Stickiness – understand your services

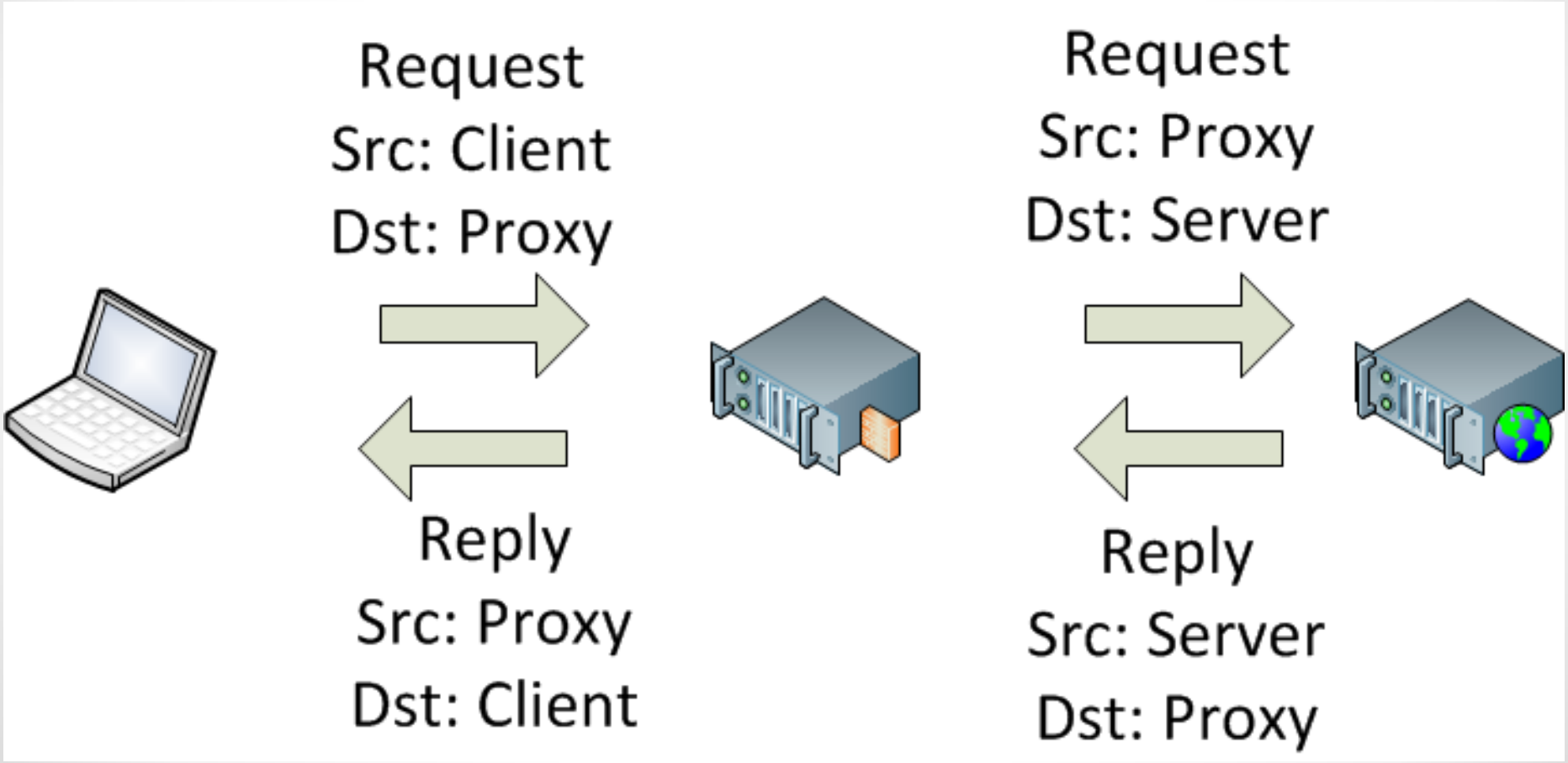
# Background - OSI Model

- Layer 1: Physical (cable, electrical)
- Layer 2: Datalink (example: Ethernet)
- Layer 3: Network (example: IP)
- Layer 4: Transport (example: TCP)
- Layer 5: Session
- Layer 6: Presentation
- Layer 7: Application (example: HTTP)

# Topologies

- Application Proxy
- Half-NAT
- Full-NAT
- Direct Server Return

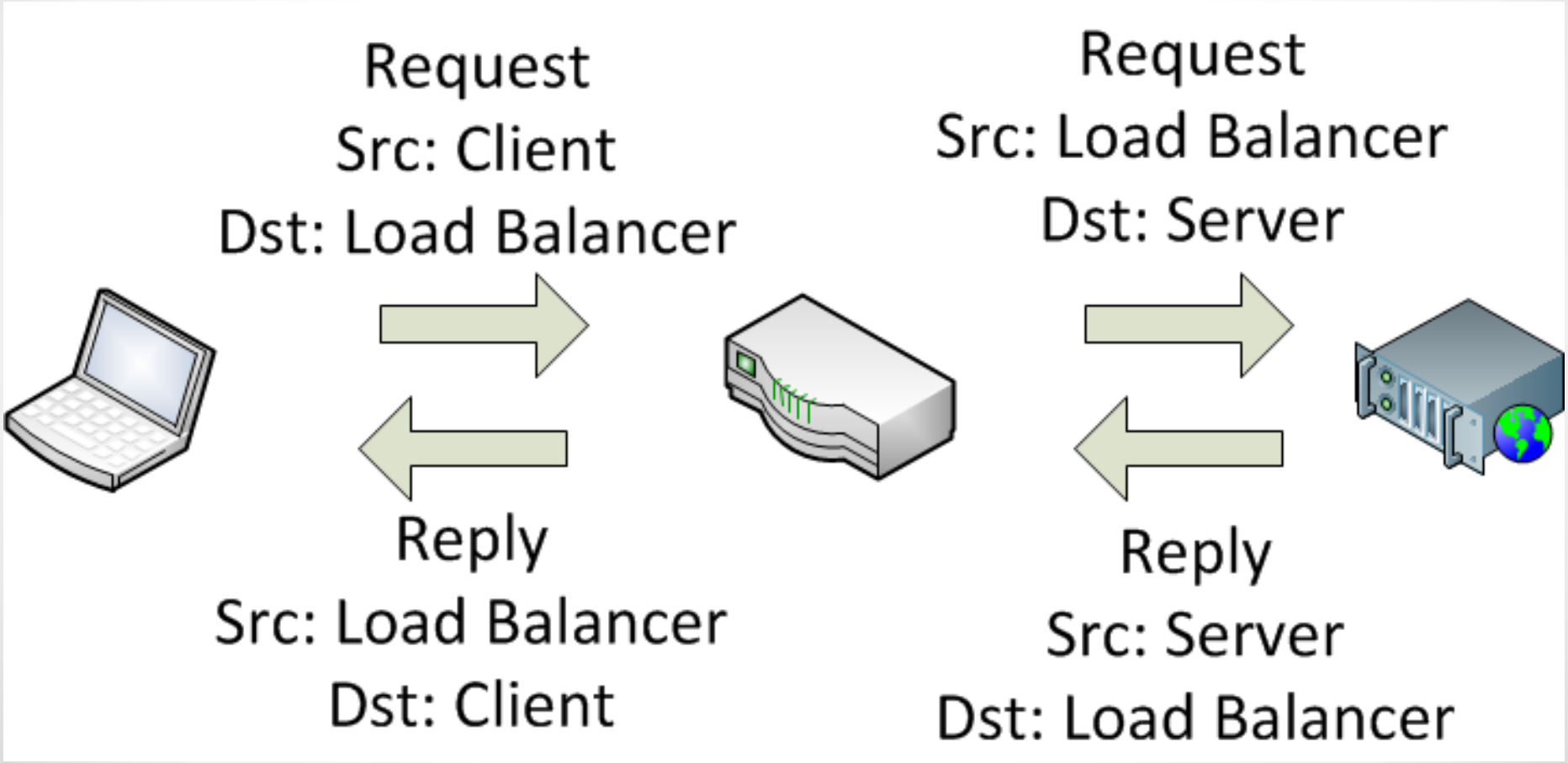
# Application Proxy



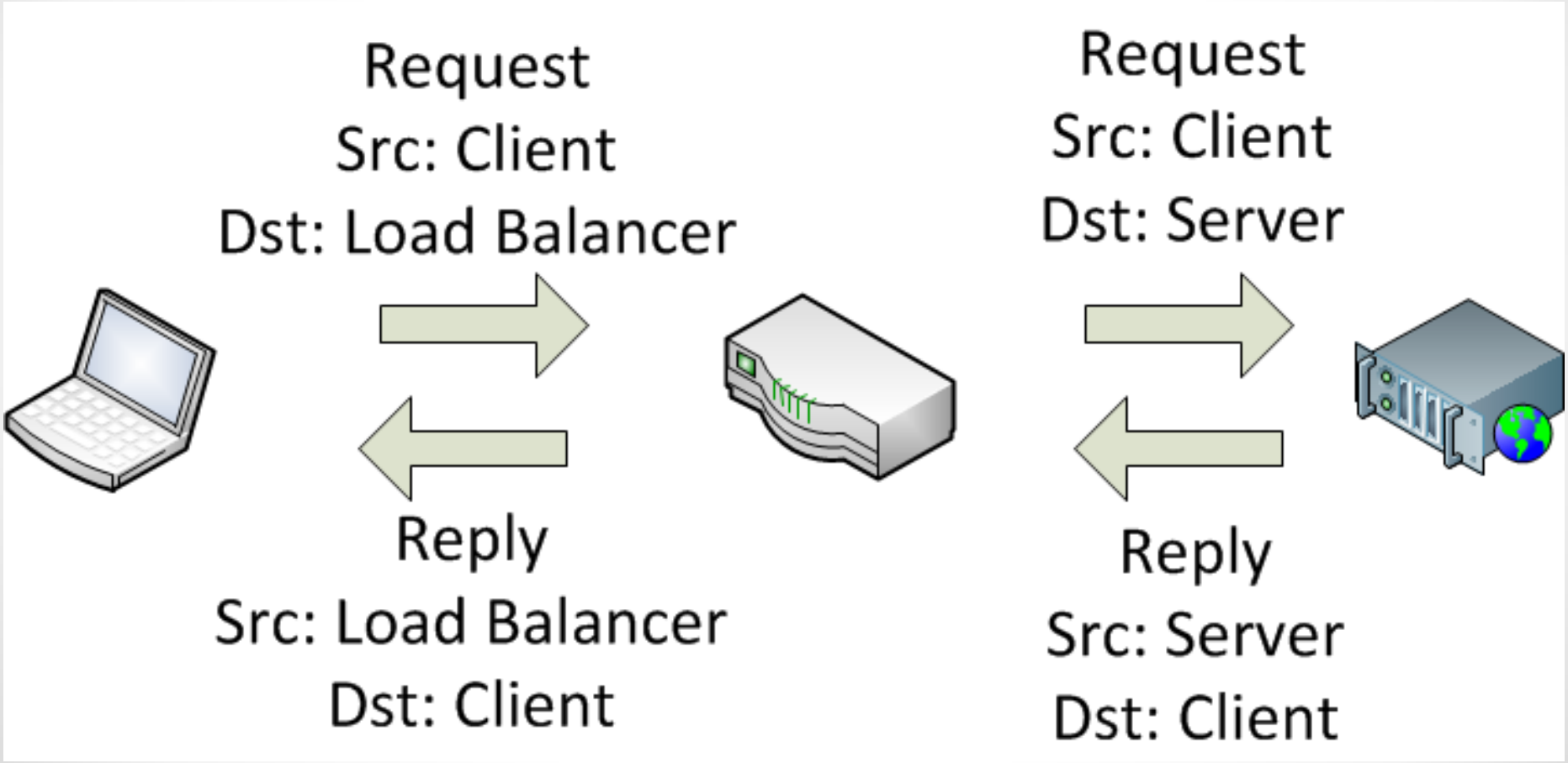
# Application Proxy

- Positives
  - Simplest to setup
  - Minimal platform dependencies
  - Minimal changes to other infrastructure
  - 100% Userspace
- Negatives
  - Limited total performance
  - Hides end user IPs from applications

# Full NAT


Request
Src: Client
Dst: Load Balancer
Request
Src: Load Balancer
Dst: Server
Reply
Src: Load Balancer
Dst: Client
Reply
Src: Server
Dst: Load Balancer

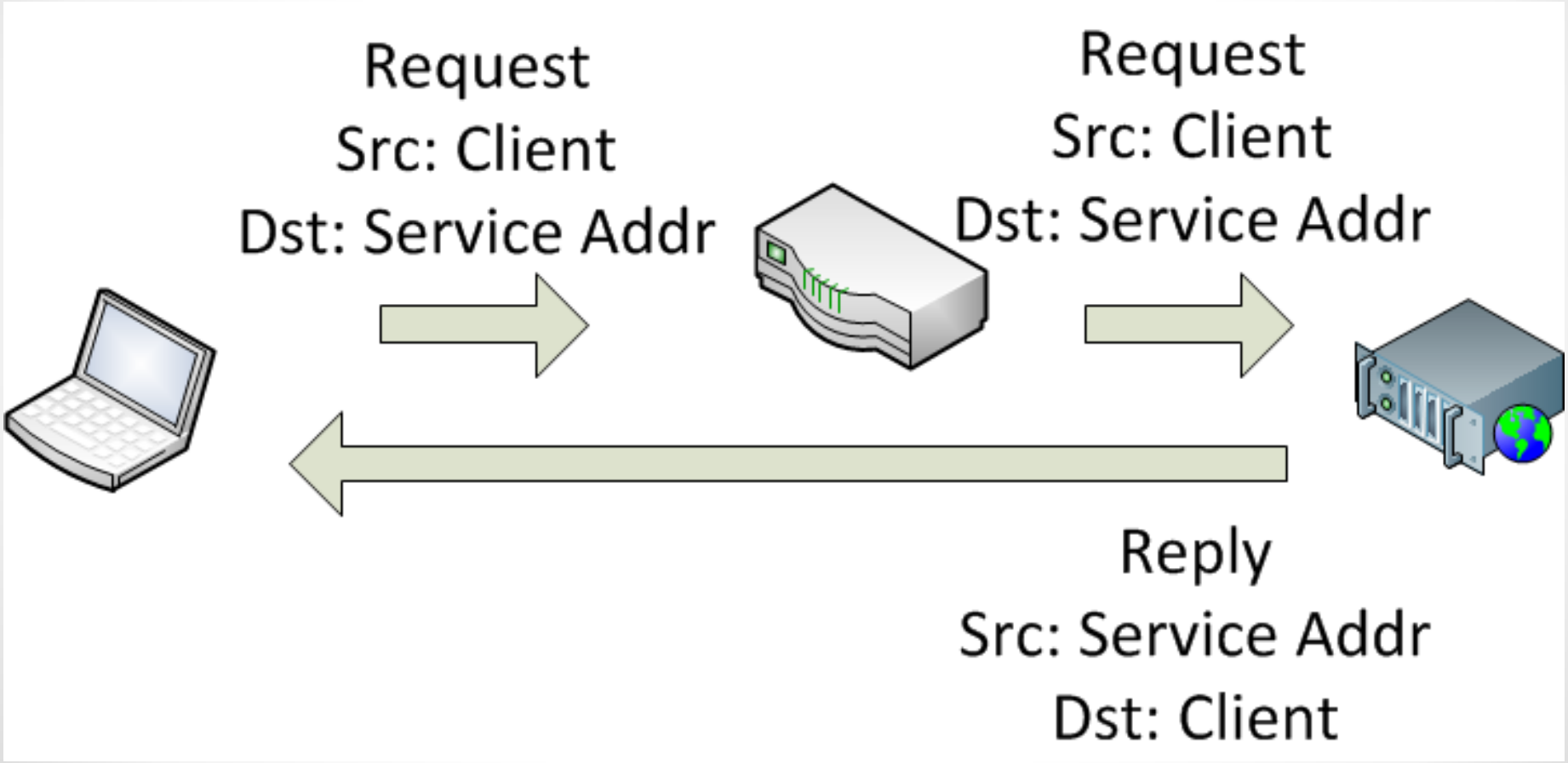
# Half NAT


Request
Src: Client
Dst: Load Balancer
Request
Src: Client
Dst: Server
Reply
Src: Load Balancer
Dst: Client
Reply
Src: Server
Dst: Client

# Half and Full NAT

- Full NAT
  - Similar to an application proxy
  - Destination still doesn't know source IP
  - All packets still go through the load balancer
- Half NAT
  - Destination IP is changed, source IP is not
  - Allows the application to know the client
  - All packets still go through the load balancer

# Direct Server Return



# Direct Server Return

- Incoming packets pass through the load balancer
- Outgoing direct to the gateway / client
- Most scalable
- Most complex to configure
- Application servers must all have public application IP, non-ARP
  - via arptables, loopback, etc

# Apache mod_proxy_balancer

- Application (layer 7) proxy for web
- Runs under any cluster manager
- Cookie based persistence
- Apache rewrite, redirect, etc at the load balancer
- Web (http, https) traffic only
- SSL offload / SSL issues
- Anything that runs Apache (even Windows)

# Apache mod_proxy_balancer

```
<VirtualHost my.site.com:80>
  ServerName my.site.com
  ProxyPass / balancer://mysite/ lbmethod=byrequests
  ProxyPassReverse / balancer://mysite
  <Proxy balancer://mysite>
    BalancerMember http://10.0.0.1/ route=mysite1
    BalancerMember http://10.0.0.2/ route=mysite2
  </Proxy>
  ProxyPreserveHost On
</VirtualHost>
```

# pen

- Runs under any cluster manager
- Simple layer 4 or layer 7 proxy
- Very simple configuration
- Moderate traffic
- Really shines for internal services
- Already IPv6 ready!
- Linux, BSD, Solaris

# pen

- Configuration via command line options
- Use init scripts from web site, or roll your own
- Init scripts store command line options in pen.cf

pen –x 6144 –c 262144 –h –H –p <pidfile> 192.168.232.20:80 192.168.232.21:80 192.168.232.22:80

pen –x 500 –c 16384 –h –p <pidfile> 192.168.232.20:993 192.168.232.23:993 192.168.232.24:993

# IPVS / Pulse / Piranha

- These work together as a system
- IPVS: load balancing
- Pulse: cluster manager (lightweight)
- Piranha: web interface for configuration
- EL5 version is IPv4 only
- EL6 version is IPv4 / IPv6
- Layer 4, in-kernel, Linux only

# IPVS

- IP Virtual Server, implemented via Netfilter
- Controlled via ipvsadm
- Or use a front-end like piranha
- Supports persistence, many schedulers

Command line:

ipvsadm –A –t 192.168.23.20:80 –s rr

ipvsadm –a –t 192.168.23.20:80 –r 192.168.23.21:80 –m

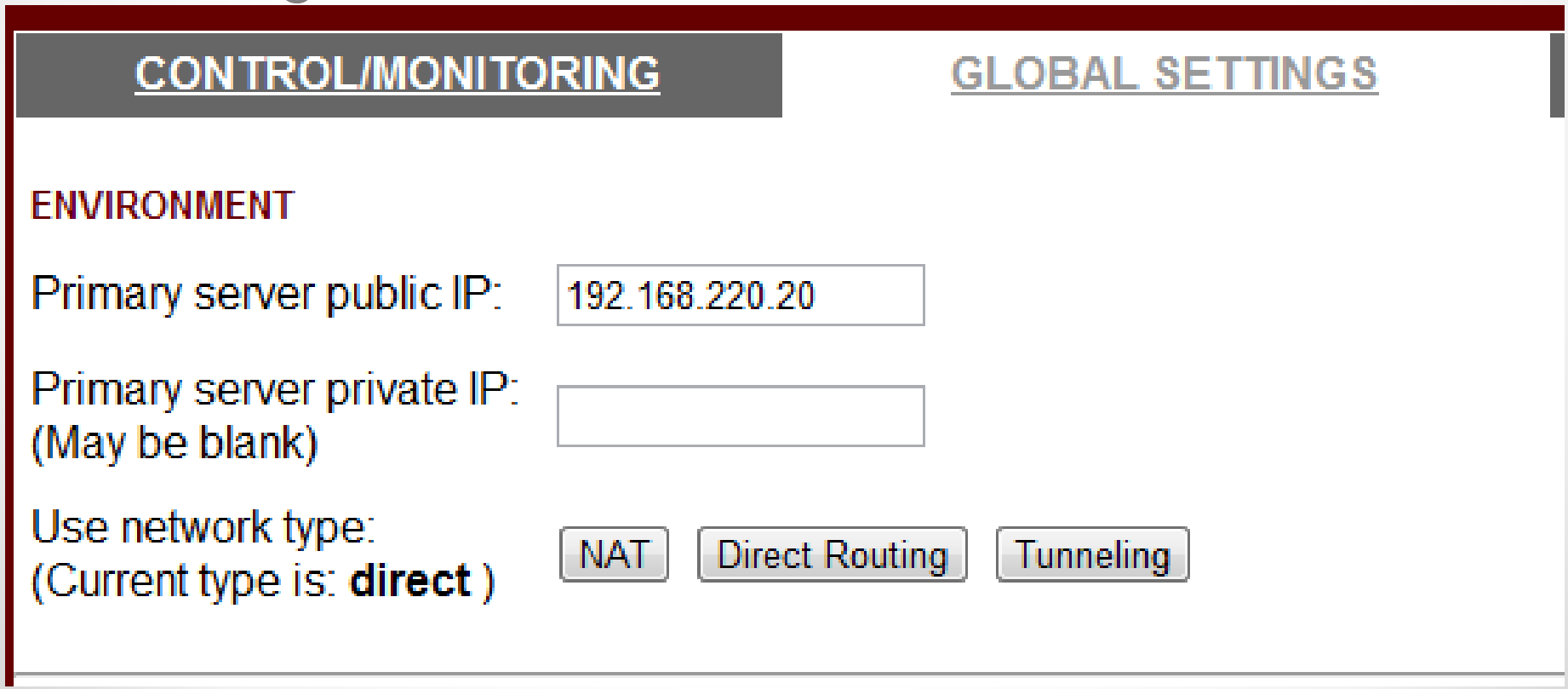
Ipvsadm –a –t 192.168.23.20:80 –r 192.168.23.22:80 –m

# Piranha

- Graphical configuration interface
- Manage Pulse and IPVS configuration
- Web based, some expensive LB use it too
- Handles half-NAT, full-NAT and DSR topologies
- Runs on port 3636, password protected
- Recommend access via ssh tunnel

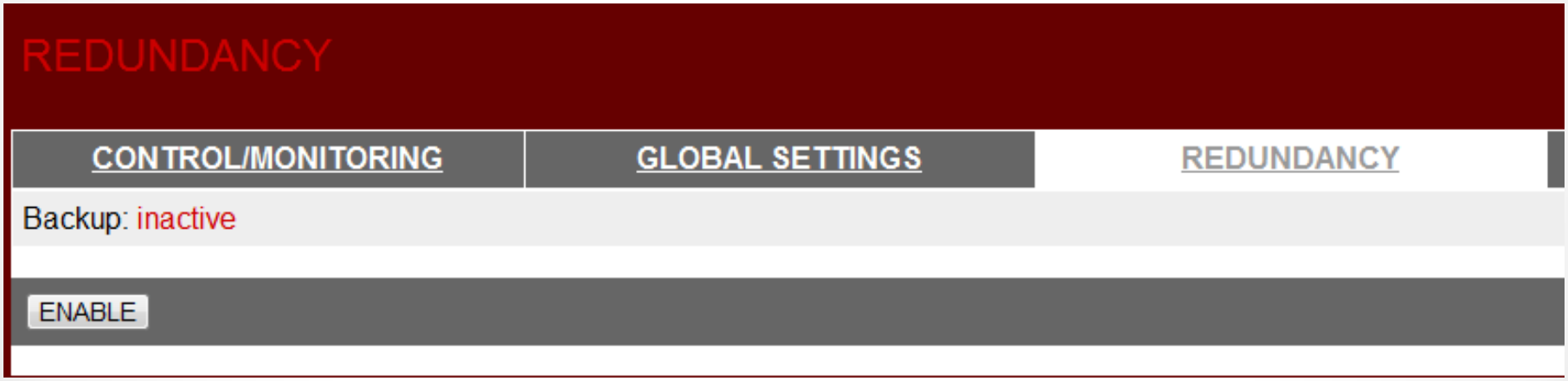
# Piranha - Pulse

- Simple, single purpose cluster manager
- Only supports 2-node active/passive failover
- Configured via Piranha web interface



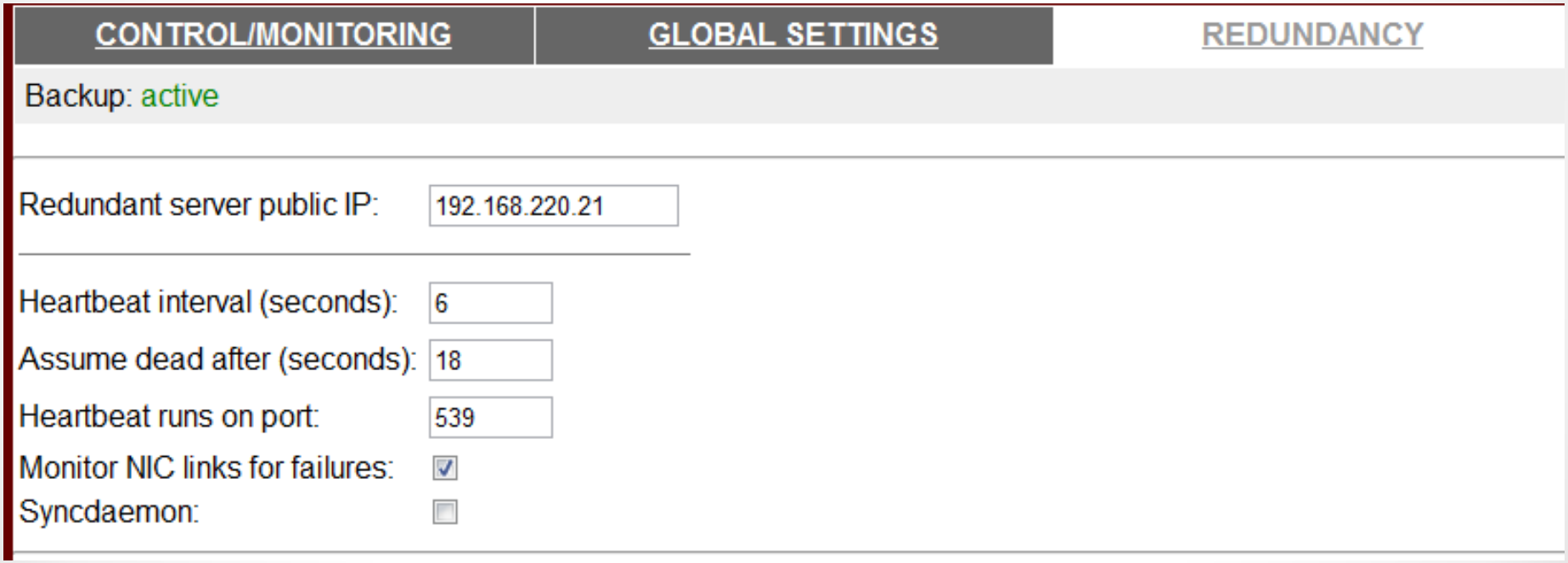
# Piranha - Pulse

Enable the Backup Server for HA



# Piranha - Pulse

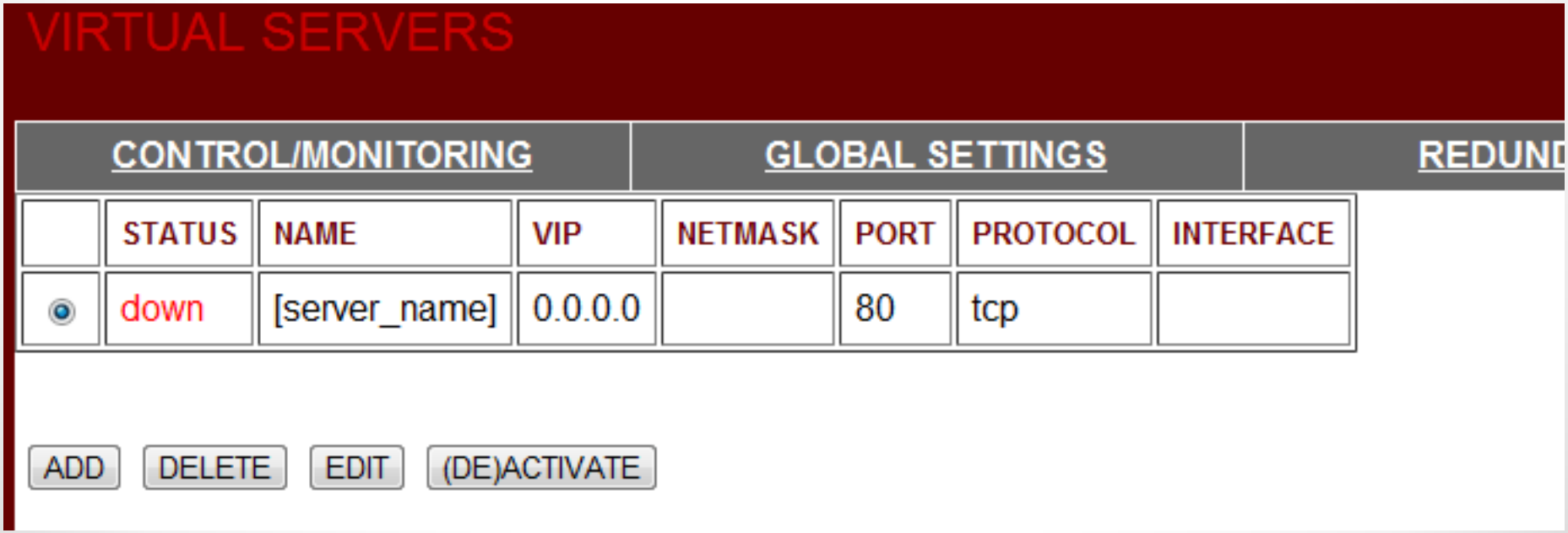
Configure the Redundant IP, Sync options



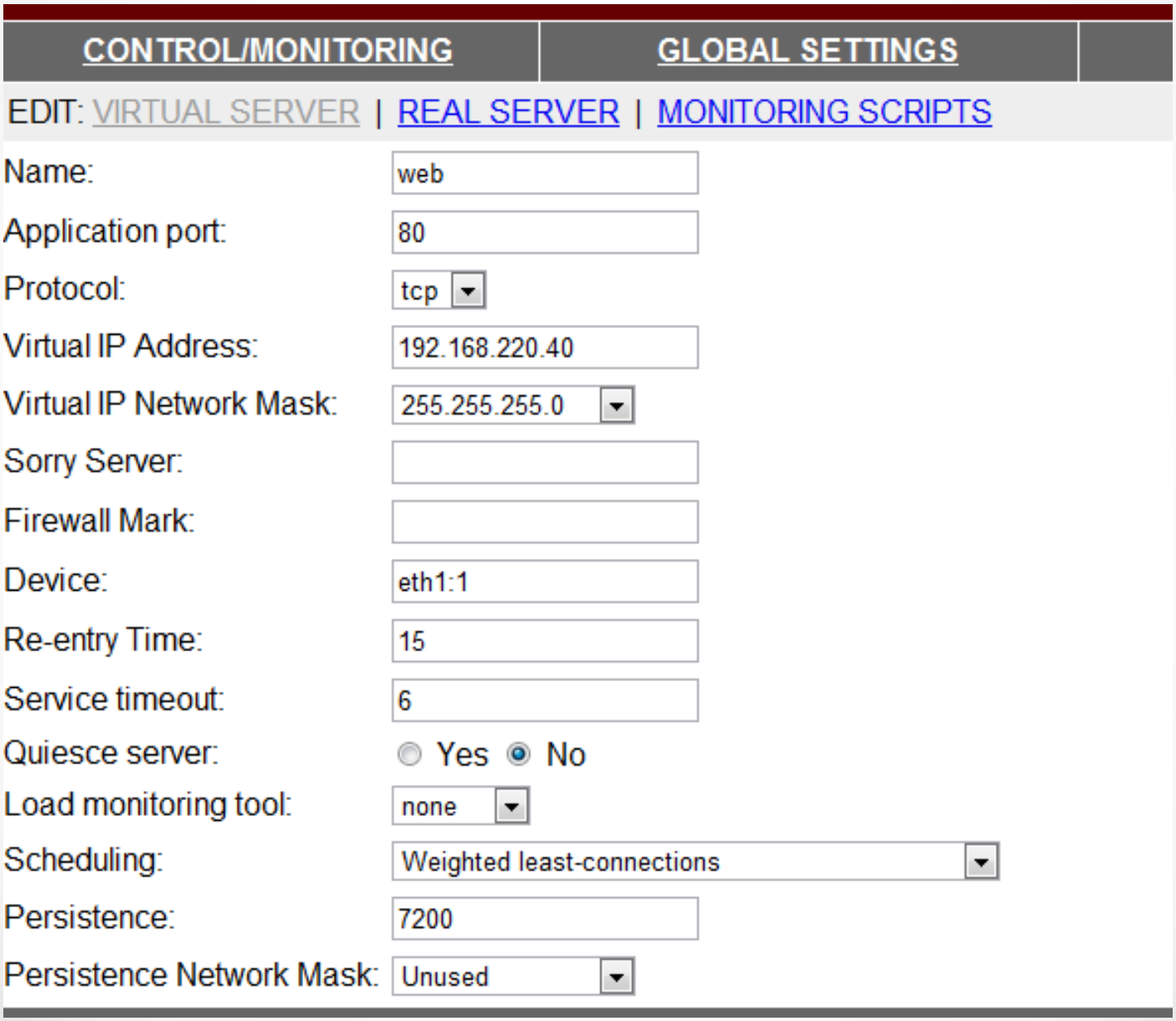
# Piranha – Virtual Server

Add a virtual server, then Edit its configuration

Be sure to make all changes on **BOTH** hosts!

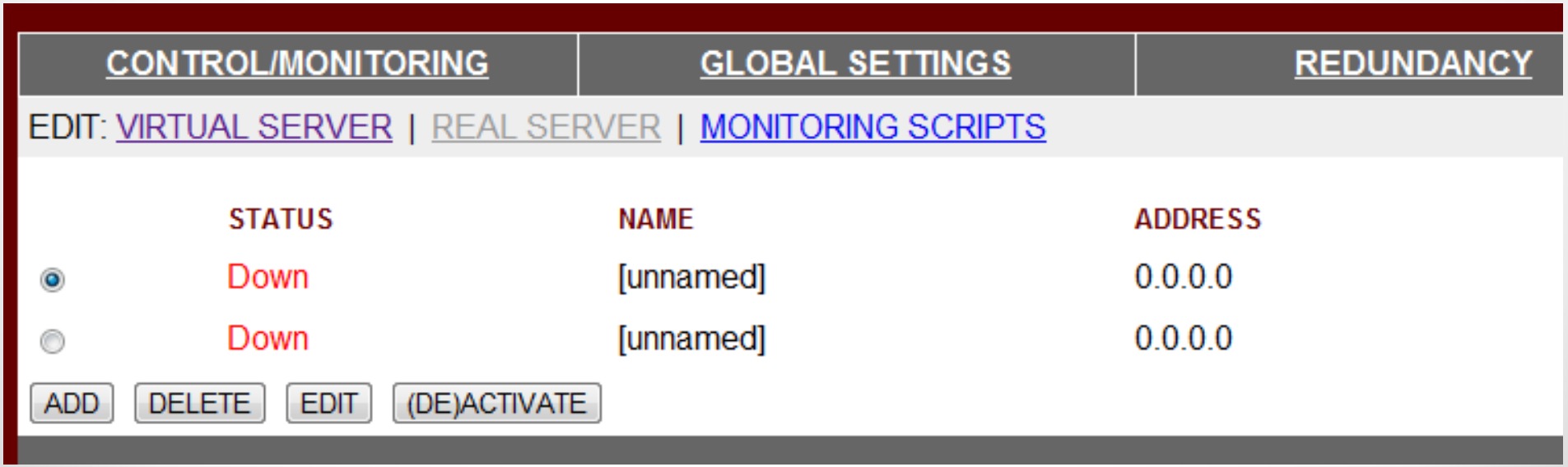


# Piranha – Virtual Server



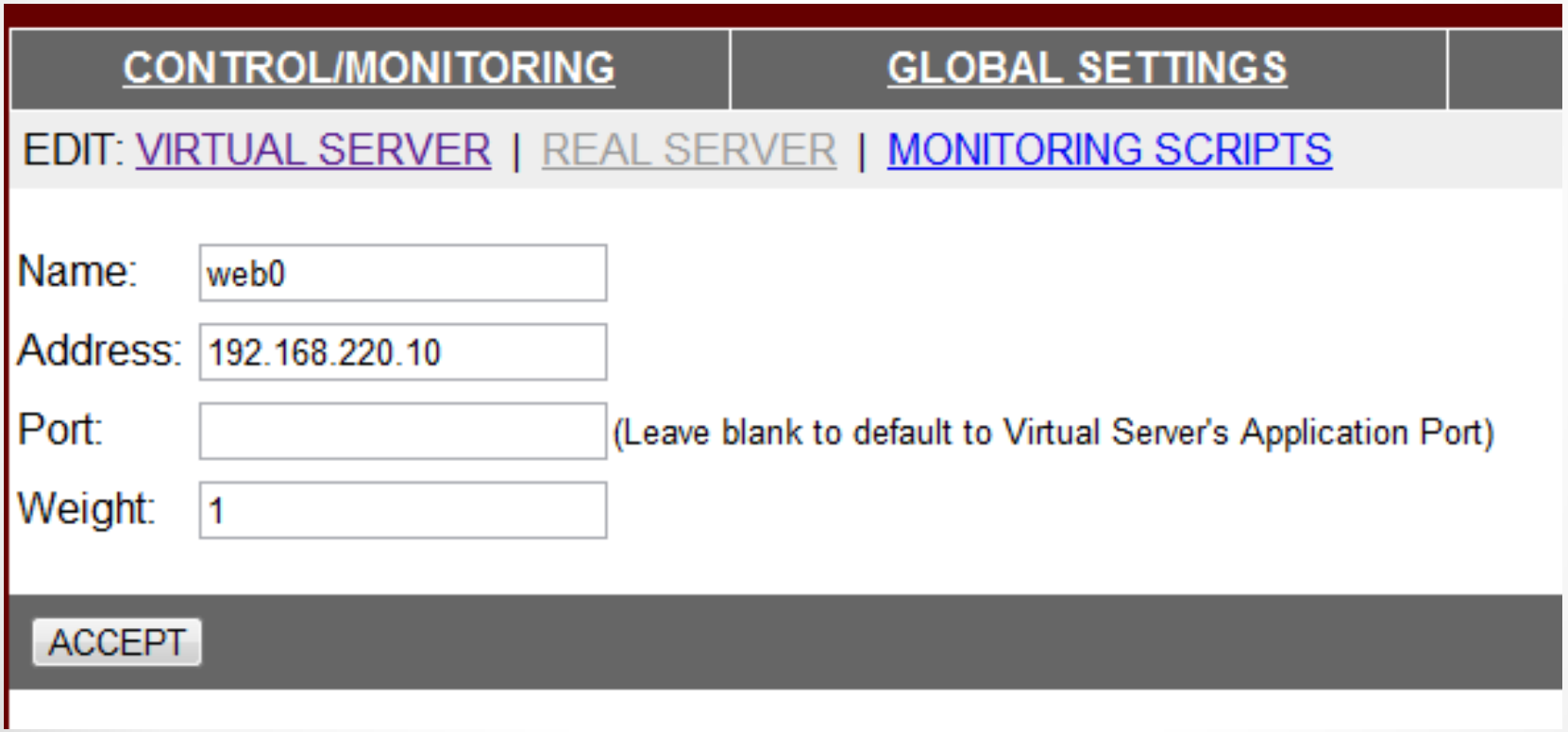
# Piranha – Real Servers

Add two real servers, and prepare to edit



# Piranha – Real Server

Configure both real servers on both hosts



# Piranha - Finalize

- Configure monitoring scripts (write if needed)
- Activate real servers
- Activate virtual servers
- Add non-ARP'd VIPs on actual real servers (if using DSR)
- Start pulse (init script) on both servers
- Test, verify, debug!

# Cluster Managers

- LVS / IPVS fits well with Pulse
- Pen and Apache are simple, run under virtually any cluster manager
- Positive experience with Heartbeat
- Choose based on organizational needs
- (aka use what your team knows!)
- Simple services, limited needs from CM

# Heartbeat, pen, Apache

- Apache (on EL5/EL6) has good init scripts
- Pen init scripts from web site need killall in stop section (otherwise it doesn't work)
- Run under Heartbeat v1 configuration as a service and an IP Address
- Apache init scripts ready for Heartbeat v2 / Pacemaker / CRM
- Pen init scripts will need a rewrite

# Minimal ha.cf

- ucast eth1 192.168.232.10
- ucast eth1 192.168.232.11
- keepalive 2
- warntime 10
- deadtime 30
- initdead 120
- udpport 694
- auto_failback on
- node lb0
- node lb1
- respawn hacluster /usr/lib64/heartbeat/ipfail

# V1 style haresources for Load Balancing

lb0 192.168.232.20 pen httpd

lb1

# IPv6!

- Bootstrapping problem, you can help!
- LVS / IPVS supports IPv6 in EL6 but not EL5
- Pen supports IPv6 out of the box
- Apache mod_proxy supports IPv6
- Reports mixed on mod_proxy_balancer
- Could use IPv6 mod_proxy in front of IPv4 mod_proxy_balancer

# Easy IPv6

- One command line, as promised!
- Uses pen, mostly cross platform (Linux / Solaris / BSD)
- Must run on a dual stack box
- Application must be TCP, not UDP
- Run under a cluster manager for HA

**pen <regular options> ipv6addr:svcport ipv4addr:svcport**

**Now you can IPv6 enable your web site!**

# Final Thoughts

- Lots of options in terms of software and topology
- This does not cover global load balancing
- This can be layered with global LB or ADN
- Balance performance, cost, complexity
- Think about organizational and application needs

# Questions and resources

http://siag.nu/pen/

http://httpd.apache.org/

http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html-single/Virtual_Server_Administration/index.html

http://lbwiki.org/

http://www.linuxvirtualserver.org/