

RRDNS (Round robin DNS) +IP TAKEOVER USING PING.

If you serve a popular site, you will find a stage after which your server can't serve more requests.

How to overcome:

- *Distribute load across multiple machines; by adding two or more servers to the server pool.
- *Let the public do the work of distributing the load for you.
- *Using the magic of RRND

Round robin DNS:

Round robin DNS is a technique in which [load balancing](#) is performed by a [DNS server](#) instead of a strictly dedicated machine. This technique is usually only implemented on large networks.

How Round robin works:

Round robin works by responding to DNS requests not with a single [IP address](#), but a list of IP addresses (all of which would assumedly host the same content). The order in which IP addresses from the list are returned is the basis of the round robin name. The IP address at the top of the list is returned a set number of times before it is moved to the bottom, thus promoting the second IP address to the top of the list. This cycle is continual and allows the DNS server to assist in balancing requests between servers.

Adding Multiple A record for a single host say linux.org

```
www 60 IN A 192.168.0.100
www 60 IN A 192.168.0.200
www 60 IN A 192.168.0.300
```

```
linux@marco:~$ host www.linux.org
www.linux.org has address 192.168.0.200
www.linux.org has address 192.168.0.100
```

The TTL has been set low (60 seconds) to prevent the caching DNS server to hanging onto one sort order for too long.

If one server goes down you can't change the DNS and wait for it to propagate to the entire internet, you will need something that takes over the down server immediately.

A Check for www.rediffmail.com

```
shibu@shibu-laptop:~$ host www.rediffmail.com
```

```
www.rediffmail.com has address 202.54.124.154
www.rediffmail.com has address 203.199.83.5
www.rediffmail.com has address 203.199.83.131
```

```
shibu@shibu-laptop:~$ host www.rediffmail.com
www.rediffmail.com has address 203.199.83.131
www.rediffmail.com has address 202.54.124.154
www.rediffmail.com has address 203.199.83.5
```

No internet standard:

There is no internet standard for deciding which address will be used by the requesting application - a few resolvers even re-order the list to give priority to numerically "closer" networks.

Who uses RRDNS:

IRC networks

Large and established networks

Many [IRC networks](#) use round robin DNS to distribute users across the servers on their networks. Indeed, virtually all the large and established networks have separate round robin DNS setup for each continent or country in which they have servers - so users can use a 'random' server local to them.

Easy to implement

Round robin DNS has important drawbacks

TTL ([Time to live](#)) values, which allows for address caching and can be very difficult to manage.

Round robin DNS **must not** solely be relied upon for service availability. If a service at one of the addresses in the list goes down, the DNS will continue to hand out that address and clients will still attempt to reach the dead service.

TTLs in the [Domain Name System](#) (DNS), where they are set by an authoritative [nameserver](#) for a particular Resource Record. When a Caching (recursive) nameserver queries the authoritative nameserver for a Resource Record, it will cache that record for the time (in seconds) specified by the TTL. If a stub resolver queries the caching nameserver for the same record before the TTL has expired, the caching server will simply reply with the already cached resource record rather than retrieve it from the authoritative nameserver again.

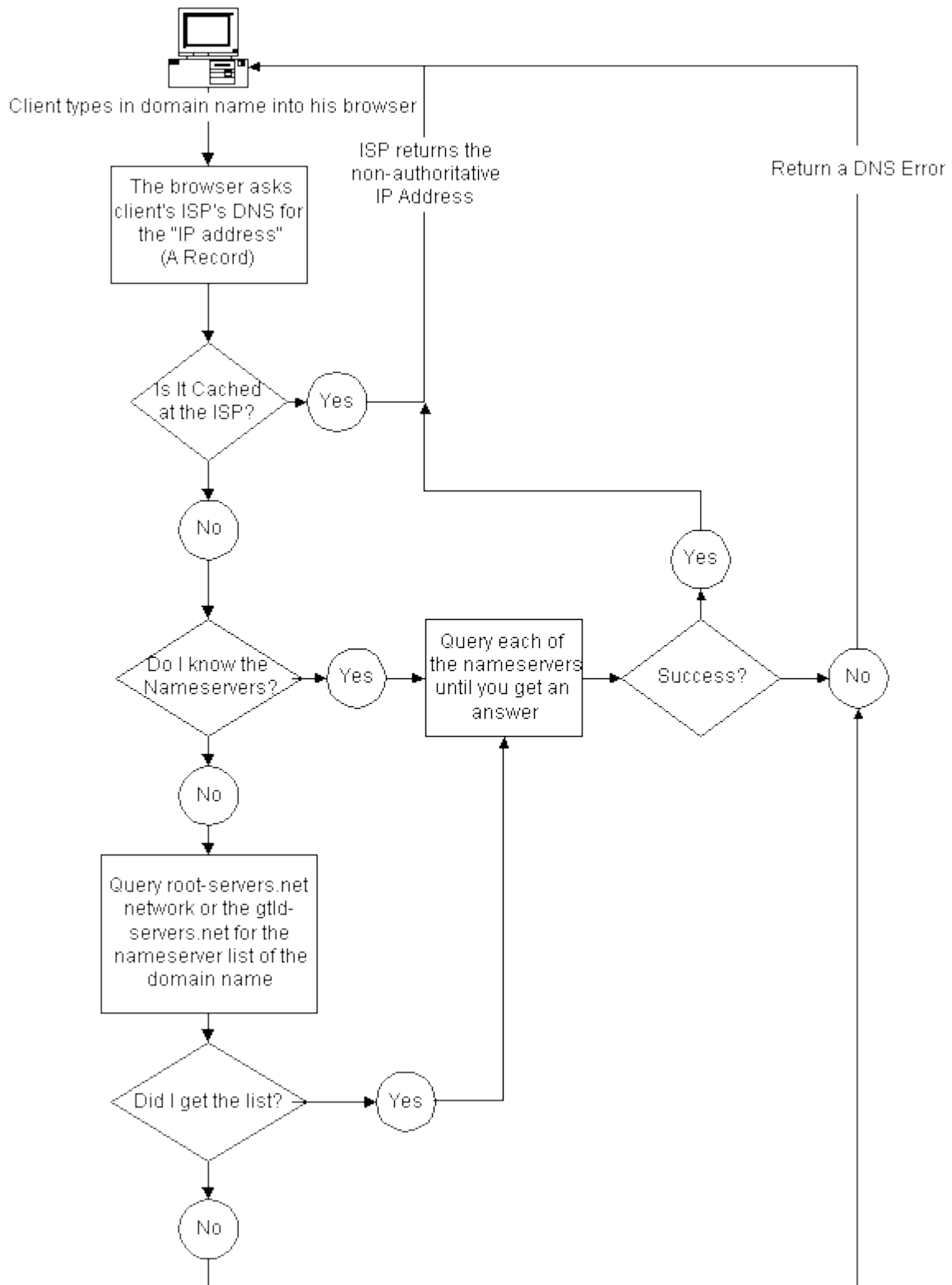
Round robin DNS load balancing will only work for services with a large number of uniformly distributed connections to servers of equivalent capacity. Otherwise it just does [load distribution](#).

To overcome this limitation:

Routinely poll servers mirroring content to see if they are online.

If a server does not return a reply as expected, the server can be temporarily removed from the DNS pool.

How DNS Works



How to keep the servers in sync

```
rsync --verbose --progress --stats --compress --rsh=/usr/local/bin/ssh
      --recursive --times --perms --links --delete \
      --exclude "*.bak" --exclude "*~" \
      /www/* webserver:simple_path_name
```

Let's go through it one line at a time. The first line calls rsync itself and specifies the options "verbose," "progress" and "stats" so that you can see what's going on this first time around. The "compress" and "rsh" options specify that you want your stream compressed and to send it through ssh (remember from above?) for security's sake.

The next line specifies how rsync itself operates on your files. You're telling rsync here to go through your source pathname recursively with "recursive" and to preserve the file timestamps and permissions with "times" and "perms." Copy symbolic links with "links" and delete things from the remote rsync server that are also deleted locally with "delete."

Now we have a line where there's quite a bit of power and flexibility. You can specify GNU tar-like include and exclude patterns here. In this example, I'm telling rsync to ignore some backup files that are common in this Web tree ("*.bak" and "*~" files). You can put whatever you want to match here, suited to your specific needs. You can leave this line out and rsync will copy all your files as they are locally to the remote machine. Depends on what you want.

Finally, the line that specifies the source pathname, the remote rsync machine and rsync "path." The first part "/www/*" specifies where on my local filesystem I want rsync to grab the files from for transmission to the remote rsync server. The next word, "webserver" should be the DNS name or IP address of your rsync server. It can be "w.x.y.z" or "rsync.mydomain.com" or even just "webserver" if you have a nickname defined in your */etc/hosts* file, as I do here. The single colon specifies that you want the whole mess sent through your ssh tunnel, as opposed to the regular rsh tunnel. This is an important point to pay attention to! If you use two colons, then despite the specification of ssh on the commandline previously, you'll still go through rsh. Oops. The last "www" in that line is the rsync "path" that you set up on the server as in the sample above.

Yes, that's it! If you run the above command on your local rsync client, then you will transfer the entire "/www/*" tree to the remote "webserver" machine except backup files, preserving file timestamps and permissions -- compressed and secure -- with visual feedback on what's happening.

Note that in the above example, I used GNU style long options so that you can see what the commandline is all about. You can also use abbreviations, single letters -- to do the same thing. Try running rsync with the "--help" option alone and you can see what syntax and options are available.

-v, --verbose	increase verbosity
-q, --quiet	suppress non-error messages
-c, --checksum	skip based on checksum, not mod-time & size
-a, --archive	archive mode; same as -rlptgoD (no -H)
-r, --recursive	recurse into directories
-R, --relative	use relative path names
--no-relative	turn off --relative
--no-implied-dirs	don't send implied dirs with -R
-b, --backup	make backups (see --suffix & --backup-dir)
--backup-dir=DIR	make backups into hierarchy based in DIR
--suffix=SUFFIX	backup suffix (default ~ w/o --backup-dir)

-u, --update	skip files that are newer on the receiver
--inplace	update destination files in-place
-d, --dirs	transfer directories without recursing
-l, --links	copy symlinks as symlinks
-L, --copy-links	transform symlink into referent file/dir
--copy-unsafe-links	only "unsafe" symlinks are transformed
--safe-links	ignore symlinks that point outside the tree
-H, --hard-links	preserve hard links
-K, --keep-dirlinks	treat symlinked dir on receiver as dir
-p, --perms	preserve permissions
-o, --owner	preserve owner (root only)
-g, --group	preserve group
-D, --devices	preserve devices (root only)
-t, --times	preserve times
-O, --omit-dir-times	omit directories when preserving times
-S, --sparse	handle sparse files efficiently
-n, --dry-run	show what would have been transferred
-W, --whole-file	copy files whole (without rsync algorithm)
--no-whole-file	always use incremental rsync algorithm
-x, --one-file-system	don't cross filesystem boundaries
-B, --block-size=SIZE	force a fixed checksum block-size
-e, --rsh=COMMAND	specify the remote shell to use

Displaying Interface Statistics

```
netstat -i
```

Displaying Connections

```
netstat -ta
```

What program has the socket open.

```
netstat -pa
```

Networking is set up in 4 files:

```
* /etc/sysconfig/network
* /etc/sysconfig/network-scripts/ifcfg-eth0
* /etc/resolv.conf
```

ifconfig and route output

```
# ifconfig eth0
```

```
# route -n
```

Bringing down a network interface with ifconfig

```
# ifconfig eth0 down
```

Bringing up an Ethernet interface with ifconfig

```
ifconfig eth0 192.168.0.100 netmask 255.255.255.0 up
```

IP Alias

```
cd /lib/modules/`uname -r`/
```

```
/sbin/ifconfig eth0 up  
/sbin/ifconfig eth0 192.168.0.12  
/sbin/ifconfig eth0:0 192.168.0.127  
/sbin/ifconfig eth0:1 192.168.0.128
```

192.168.0.12 is the main IP #, while .127 and .128 are the aliases.

The Script that does the Work.

Logic of the script... if not able to ping marconi from 192.168.0.11 then...create auto create a new virtual IP at 192.168.0.11 with marconis virtual IP 192.168.0.127.

If able to ping marconi's IP address the take down marconi's virtual IP at 192.168.0.11.

```
#!/bin/bash  
#http://tldp.org/LDP/Bash-Beginners-Guide/html/sect\_09\_02.html --while  
#http://pegasus.rutgers.edu/~elflord/unix/bash-tute.html ----if statement  
  
marc="192.168.0.12" # marconi's main IP address.  
vmarc="192.168.0.127" # marconi's virtual IP address  
PAUSE=2  
MISSED=0  
  
while true; do  
  if ! ping -c 1 -w 1 $marc > /dev/null; then  
    ifconfig eth0:marc $vmarc #Chane virtual IP at galelio when not able to ping marconi  
  else  
    ifconfig eth0:marc down #When able to ping  
  fi  
  sleep $PAUSE;  
done
```

Modified Script...

```
#!/bin/bash
PINGMARC="192.168.0.12"
MARCPUBLIC="192.168.0.127"

PAUSE=2

PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/sbin
MISSED=0

while true; do
  if ! ping -c 1 -w 1 $PINGMARC > /dev/null; then
    ((MISSED++))
    echo $MISSED
  else
    if [ $MISSED -eq 0 ]; then
      echo "take down $MARCPUBLIC down"
      ifconfig eth0:macr $MARCPUBLIC down
    fi
    MISSED=1
  fi

  if [ $MISSED -eq 2 ]; then
    echo "take up $MARCPUBLIC"
    ifconfig eth0:macr $MARCPUBLIC
    #
    # ...but see discussion below...
    #
  fi
  sleep $PAUSE;
done
```