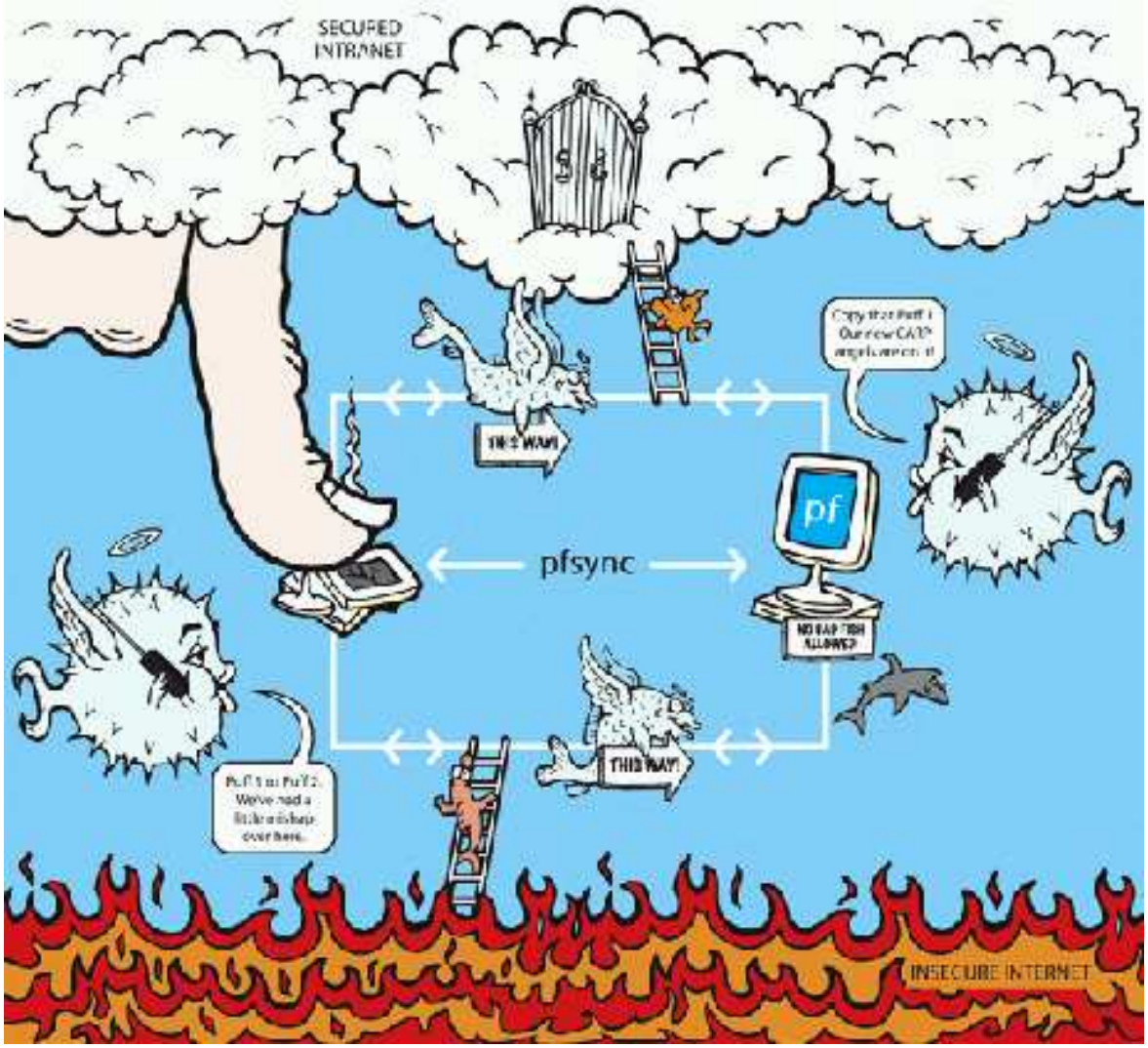
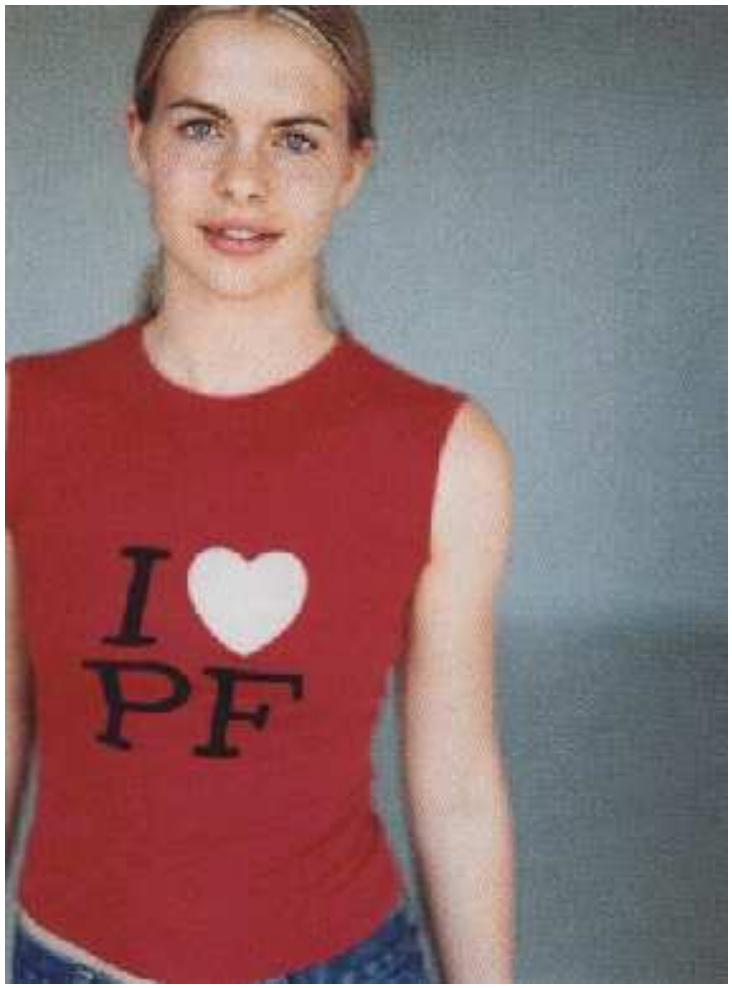
Introduction to pf



Ryan McBride <mcbride@openbsd.org>

Overview

- The Basics
  - Normalisation
  - Filtering
  - Translation
- Advanced Toolkits
  - Denial of Service Mitigation
  - Firewall Redundancy
  - Load Balancing
- New stuff



Normalization (scrub)

- Sanitizes packet content to remove ambiguity
- IP fragment reassembly
- IP normalisation
- TCP normalisation
  - Illegal flag combinations
  - TCP options
  - PAWS (Protect Against Wrapped Sequence Numbers)
- Enforce minimum TTL

Filtering

- Filterable Attributes
- Stateful Rules
- Tables
- Anchors

Filterable Attributes (protocol independent)

- Interface
- Direction
- Address family
- Protocol
- Source/destination address
- TOS
- Fragments
- IP options
- Tagging

Filterable Attributes (protocol dependant)

- Source/destination port (TCP and UDP)
- ICMP code and type (ICMP)
- User/group (TCP and UDP)
- TCP flags (TCP)
- Source OS (TCP)

OS Fingerprints

- Source OS only
- Looks at initial TCP packet
- Based on p0f, by lcamtuf@coredump.cx
- Can filter by general OS or specific version/patchlevel

- Can be spoofed
  - A policy tool, not a security tool

Tagging

- Rules can apply a named tag to a packet
  - Only one tag per packet
  - Pass rules with tagging must be stateful
- Subsequent rules can match on that tag
- Bridge code can also tag packets

- Allows the separation of classification and policy

Stateful Rules

- States indexed in a red-black tree
  - State searches are faster than rule lookup
- States increase security
  - Can control who initiates a connection
  - TCP segments must be within window
  - reset must be on edge of window

Tables

- Implemented as radix tree
- Very fast lookups
- Bytes/packet counters for each table entry

Anchors

- Placeholder for rules to be loaded later
- Changing anchor does not change main ruleset

Translation

- nat - source address translation
- rdr - destination address translation
- binat - bidirectional address translation

Denial of Service Mitigation

- Caveat: very difficult to combat bandwidth-based DoS

- synproxy
- Adaptive Timeouts
- max-src-states and max-src-nodes
- ALTQ
- Input queue congestion handling

synproxy

- pf completes the 3 way handshake
- Does 3 way handshake with destination
- Remaining traffic is a normal stateful connection
  - (with modulated sequence numbers)

Adaptive Timeouts

- Scales timeouts as the total number of states increases
- Unused states die more quickly

max-src-states and max-src-nodes

- Works with ’source-tracking’
  - states tracked by source IP
- max-src-states limits states per source
- max-src-nodes limits number of sources

ALTQ

- Bandwidth shaping
- Can filter traffic based on filter attributes
- Works only with stateful rules
- Multiple queueing disciplines supported

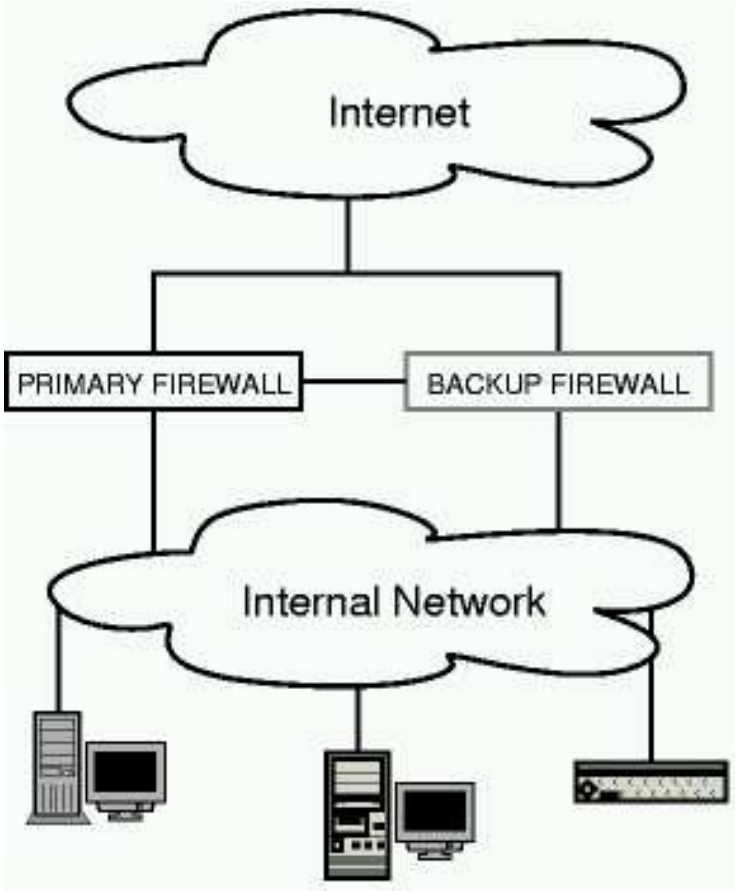
- Most effective in front of bandwidth bottleneck
  - eg at upstream ISP(s)

Input queue congestion handling

- Under some dDoS attacks CPU is overloaded
  - Input queue fills up
  - Machine becomes unresponsive
- When input queue is full stop evaluating ruleset
  - stateful packets are passed
  - stateless packets dropped unconditionally

- Packets would have gotten dropped anyways
- Machine stays responsive

Firewall Redundancy



Components

- pfsync
  - State synchronisation
- CARP
  - Address failover

pfsync

- Each firewall sends out state changes via multicast
  - Inserts
  - Updates
  - Deletes
- States have a unique id
  - Incrementing counter and host id
- Best effort
  - Systems tend towards complete synchronisation
- Mechanisms to limit bandwidth/packets
  - Updates contain only changing state information
  - Multiple updates are merged into one

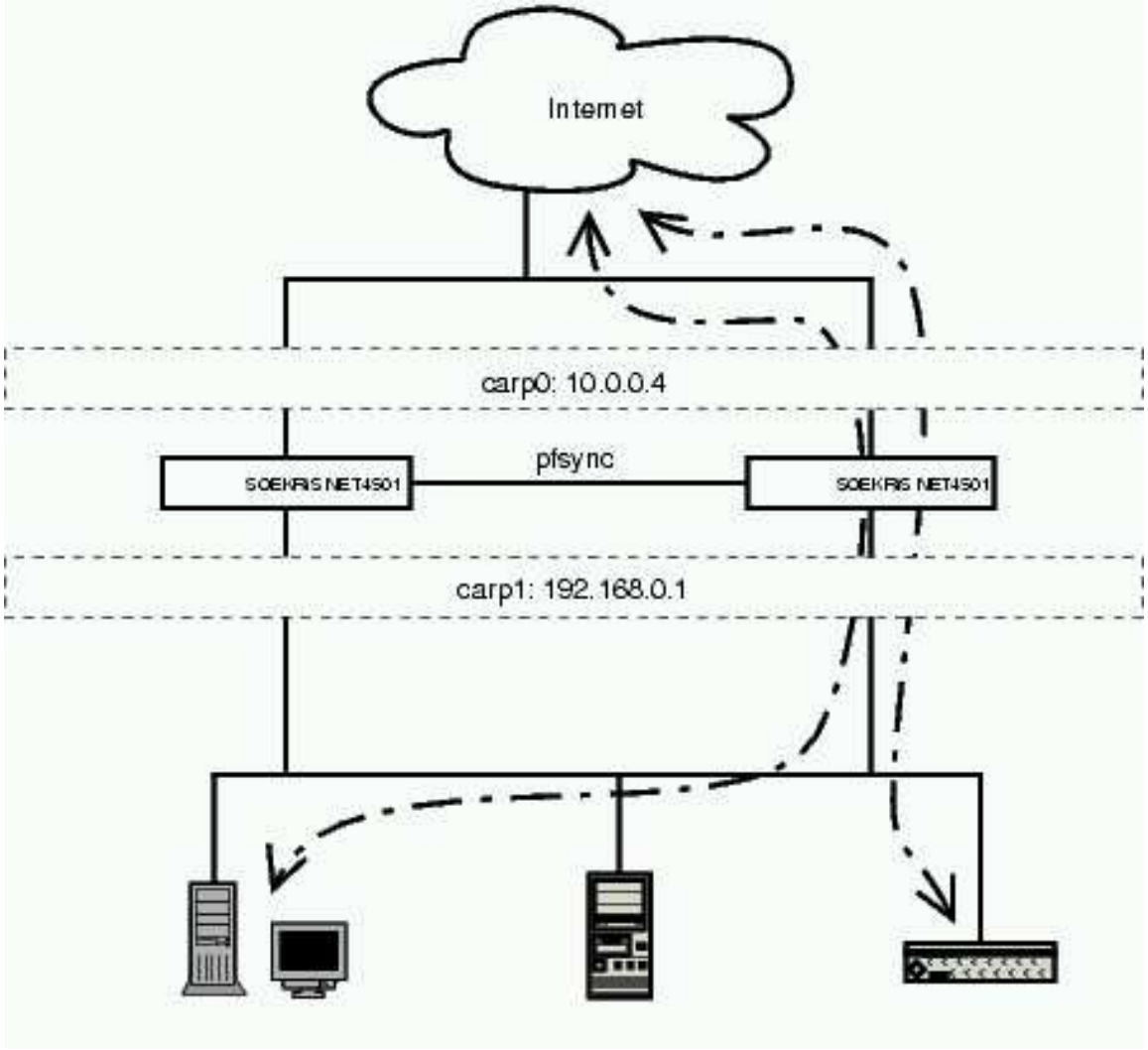
CARP

- Similar in some ways to VRRP
  - Multicast Advertisement
  - Address moved by moving a virtual MAC address
  - Multiple virtual addresses on same network
- Variable advertisement interval
  - most frequent advertiser becomes master
- Advertisement protected by a SHA1 HMAC
- Addresses not in Advertisement, but in HMAC
- Supports layer 2 load balancing (ARP based)
- IPv4 and IPv6 support

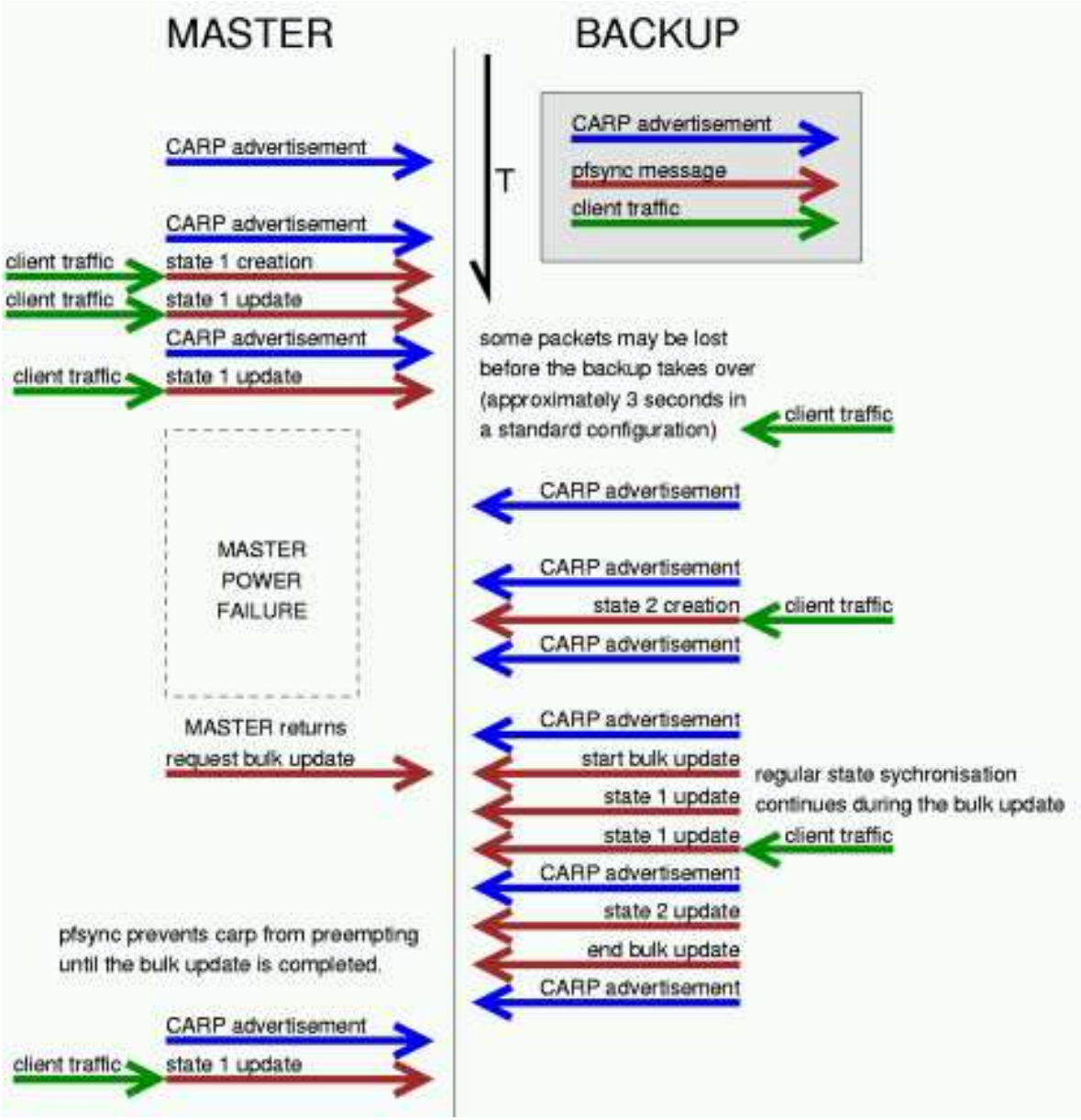
pfsync and CARP integration

- pfsync requests a bulk update when system comes up
- Prevents CARP preemption until bulk update complete

Example


Internet
carp0: 10.0.0.4
pfsync
SOEKRIS NET4501
SOEKRIS NET4501
carp1: 192.168.0.1

Timeline


MASTER
BACKUP
CARP advertisement
pfsync message
client traffic
T
CARP advertisement
client traffic
state 1 creation
client traffic
state 1 update
CARP advertisement
client traffic
state 1 update
some packets may be lost
before the backup takes over
(approximately 3 seconds in
a standard configuration)
client traffic
CARP advertisement
MASTER
POWER
FAILURE
CARP advertisement
state 2 creation
client traffic
CARP advertisement
MASTER returns
request bulk update
CARP advertisement
start bulk update
regular state sychronisation
state 1 update
continues during the bulk update
state 1 update
client traffic
CARP advertisement
state 2 update
end bulk update
CARP advertisement
pfsync prevents carp from preempting
until the bulk update is completed.
CARP advertisement
client traffic
state 1 update

Load Balancing

- rdr / nat with multiple addresses
- CARP

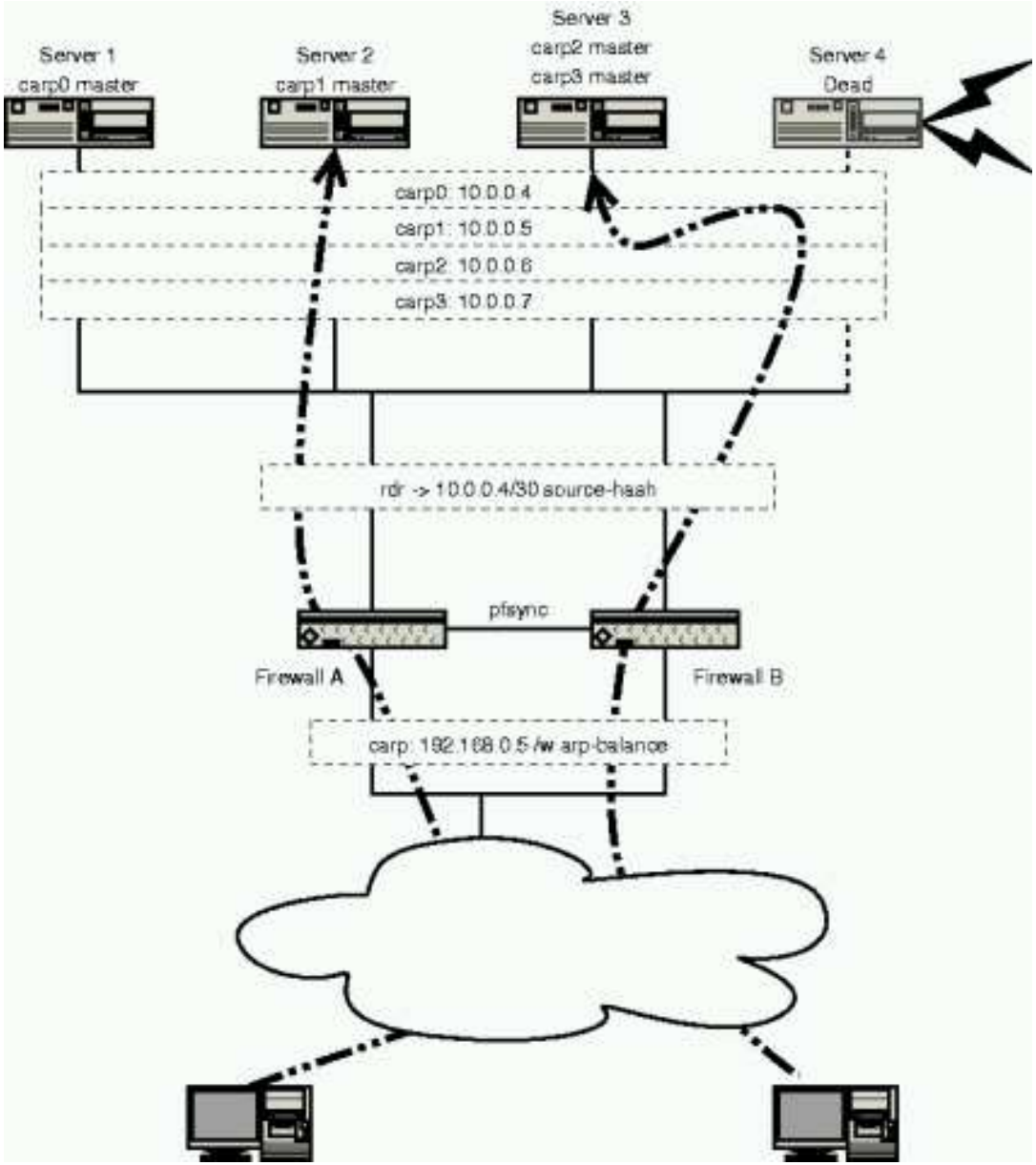
rdr / nat with multiple addresses

- Several address selection options
  - bitmask
  - source-hash
  - random
  - round-robin
- sticky-address
  - Can be used with ’random’ and ’round-robin’
  - Ties the source address to the translation address

CARP

- Can also provide failover to hosts as well as routers
- ’arpbalance’ balances based on arp requests
  - Multiple carp groups (one per host)
  - Group selected based on ARP request source
  - Master of that group responds with ARP
  - Only works on local segment

Example


Server 1
carp0 master
Server 2
carp1 master
Server 3
carp2 master
carp3 master
Server 4
Dead
carp0: 10.0.0.4
carp1: 10.0.0.5
carp2: 10.0.0.6
carp3: 10.0.0.7
rdr -> 10.0.0.4/30 source-hash
pfsync
Firewall A
Firewall B
carp: 192.168.0.5 /w arp-balance

(Other) New stuff

- Recursive anchors
- More carp and pfsync integration
- probability
- +++