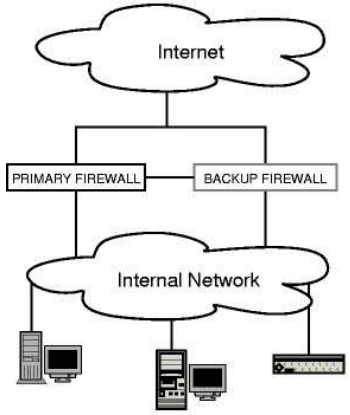
Home
Consulting

Firewall Failover with pfsync and CARP

On most networks, the firewall is a single point of failure. When the firewall goes down, inside users are unable to surf the web, the website goes dead to the outside world, and email grinds to a halt. The 3.5 release of OpenBSD includes a number of components which can be used to solve this problem, by placing two firewalls in parallel. All traffic passes through the primary firewall; when it fails the backup firewall assumes the identity of the primary firewall, and continues where it left off. Existing connections are preserved, and network traffic continues as if nothing had happened.



Not only does such a configuration increase the reliability of the network, it can also increase the security in some subtle ways. It is now trivial to do upgrades without impacting the network, by taking the firewalls offline one at a time. The result? Hopefully the firewalls will be upgraded more frequently and there will be less resistance to applying patches "because the network will go down". Furthermore, in many corporate environments there is strong pressure to keep the network up "no matter what". Frequently then a firewall failure means running unprotected rather than waiting until a new one can be brought up - obviously increasing firewall reliability reduces the risk of this happening.

The tools

The two main components provided by OpenBSD are CARP (the **C**ommon **A**ddress **R**edundancy **P**rotocol), which allows a backup host to assume the identity of the primary, and pfsync, which ensures that firewall states are synchronised so that the backup can take over exactly where the master left off and no connections will be lost.

CARP

The Common Address Redundancy Protocol manages failover at the intersection of Layers 2 and 3 in the OSI Model (link layer and IP layer). Each CARP group has a virtual MAC (link layer) address, and one or more virtual host IP addresses (the common address). CARP hosts respond to ARP requests for the common address with the virtual MAC

Home
Consulting

address, and the CARP advertisements themselves are sent out with this as the source address, which helps switches quickly determine which port the virtual MAC address is currently "at".

The master of the address sends out CARP advertisement messages via multicast using the CARP protocol (IP Protocol 112) on a regular basis, and the backup hosts listen for this advertisement. If the advertisements stop, the backup hosts will begin advertising. The advertisement frequency is configurable, and the host which advertises most frequently is the one most likely to become master in the event of a failure.

A reader who is familiar with VRRP will find this is somewhat familiar, however there are some significant differences:

- The CARP protocol is address family independent. The OpenBSD implementation supports both IPv4 and IPv6, as a transport for the CARP packets as well as common addresses to be shared.
- CARP has an "arpbalance" feature that allows multiple hosts to share a single IP address simultaneously; in this configuration, there is a virtual MAC address for each host, but only one IP address.
- CARP uses a cryptographically strong SHA-1 HMAC to protect each advertisement.

Besides these technical differences, there is another significant difference (perhaps the most important one, in fact): CARP is not patent encumbered. See this page for details on the history of CARP and our reasons for avoiding a VRRP implementation.

pfsync

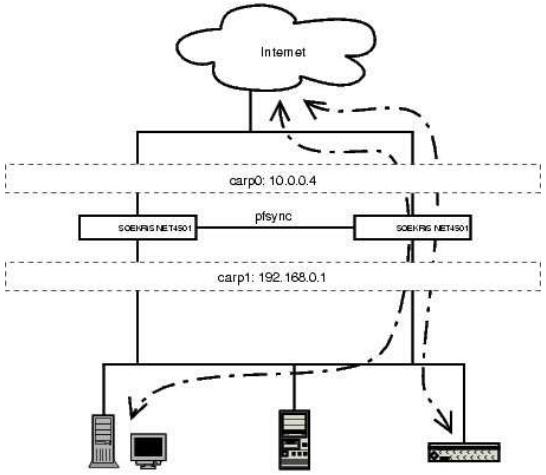
pfsync transfers state insertion, update, and deletion messages between firewalls. Each firewall sends these messages out via multicast on a specified interface, using the PFSYNC protocol (IP Protocol 240). It also listens on that interface for similar messages from other firewalls, and imports them into the local state table.

In order to ensure that pfsync meets the packet volume and latency requirements, the initial implementation has no built-in authentication. An attacker who has local (link layer) access to the subnet used for pfsync traffic can trivially add, change, or remove states from the firewalls. It's possible to run the pfsync protocol on one of the "real" networks, but because of the security risks, it is **strongly** recommended that a dedicated, trusted network be used for pfsync. This can be as simple as a crossover cable between interfaces on two firewalls

A basic example

My firewall cluster at home consists of two Soekris Engineering 4501s. Each device has three interfaces - one interface connects to a hub on the external, Internet side, one interface connects to the internal network, and the third interface connects the two firewalls to each other via a crossover cable: a dedicated link for the pfsync protocol.

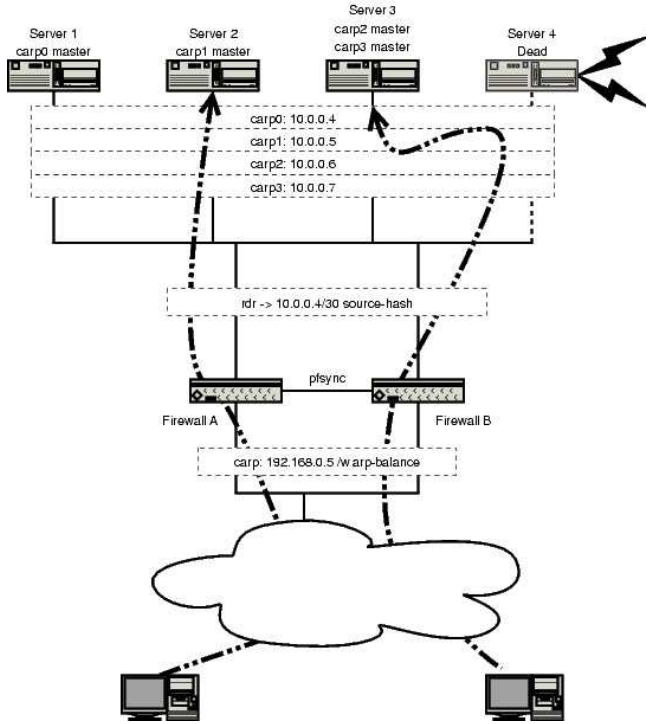
Home
Consulting



Configuration details for this setup are available below.

Something bigger

This larger configuration is in place at a large educational institution, providing load balancing and redundancy for a cluster of web servers:



Home
Consulting

In this configuration, both outer firewalls are handling connections. They both answer to the same IP address, but have unique MAC address. Which MAC address to hand out is determined based on the source address of the incoming ARP request.

Passing through the firewalls, traffic is redirected from the single, outer IP address to one of the hosts on the inside. The source-hash option is used which, by selecting a redirection target based on a hash of the source address, ensures that multiple connections from the same client will all be redirected to the same server.

On the second layer, the 4 application servers take part in 4 different carp groups, backing each other up. If one of the servers dies, as Server 4 has done in the above example, one of the other servers will take over that address, and serve requests for both its own and the one it is backing up.

Preemption

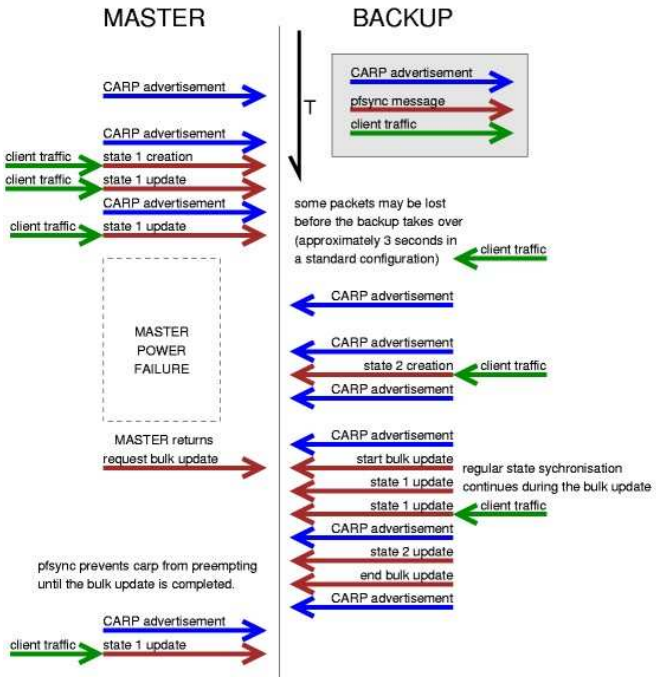
In many cases, the firewalls in the cluster are identical, and it doesn't matter which one is currently master. Allowing hosts to hold on to the virtual address indefinitely reduces the number of transitions, but if for some reason it is preferable that one firewall handle the traffic whenever possible, the intended master can be told to preempt the backup and take back the address.

When preemption is enabled, each CARP host will look at the advskew parameter in the advertisements it receives from the master, to try to determine whether it can advertise more frequently. If so, it will begin advertising, and the current master, seeing that there is another host with a lower advskew, will bow out.

The Failover Sequence

The diagram below illustrates a time-line of events in a typical failover, and illustrates what happens with preemption enabled. When the pfsync interface first comes up, pfsync broadcasts a request for a bulk update of the entire state table. After this, all updates to the state table are on a per-state, best effort basis. pfsync attempts to prevent carp from taking ownership of the common addresses until the bulk update has completed.

Home
Consulting



Scalability

There is essentially no limit to how many pfsync+carp hosts can participate in a cluster. Except for the bulk update at bootup, the traffic generated by the pfsync protocol scales linearly with the amount of regular traffic passing through the firewall cluster, and besides brief periods when a new master is being selected, only one host in a carp group is advertising at any given time.

In test environments, we have run up to 4 pfsync+carp hosts (all different architectures: i386, sparc, sparc64, and amd64!), randomly rebooting them. TCP sessions were not interrupted through over two days of such torture testing.

Sample configuration

The sample configuration given here is for the basic example above. Each box has three sis(4) interfaces; sis0 is the external interface, on the 10.0.0.0/24 subnet, sis1 is the internal interface, on the 192.168.0.0/24 subnet, and sis2 is the pfsync interface, using the 192.168.254.0/24 subnet. A crossover cable connects the two firewalls via their sis2 interfaces. On all three interfaces, firewall A uses the .254 address, while firewall B uses .253.

The interfaces are configured as follows:

/etc/hostname.sis0:

```
inet 10.0.0.254 255.255.255.0 NONE
```


Home
Consulting

/etc/hostname.sis1:

```
inet 192.168.0.254 255.255.255.0 NONE
```

/etc/hostname.sis2:

```
inet 192.168.254.254 255.255.255.0 NONE
```

/etc/hostname.carp0:

```
inet 10.0.0.1 255.255.255.0 10.0.0.255 vhid 1 pass foo
```

/etc/hostname.carp1:

```
inet 192.168.0.1 255.255.255.0 192.168.0.255 vhid 2 pass bar
```

/etc/hostname.pfsync0:

```
up syncif sis2
```

pf(4) must also be configured to allow pfsync and CARP traffic through. The following should be added to the top of /etc/pf.conf:

```
pass quick on { sis2 } proto pfsync
pass on { sis0 sis1 } proto carp keep state
```

When writing the rest of the pf ruleset, it is important to keep in mind that from pf's perspective, all traffic comes from the physical interface, even if it is routed through the carp address. However, the address is of course associated with the carp interface. Therefore, in the interface context, such as "pass in on $extif ...", $extif would be the physical interface, but in the context of "from $foo" or "to $foo", the carp interface should be used, as it's being meant in the address context.

Preemption

If preemption is desired, the server intended to be the backup needs to have a higher advskew than the primary. In this case, the carp configuration for the backup firewall would be as follows:

/etc/hostname.carp0:

```
inet 10.0.0.1 255.255.255.0 10.0.0.255 vhid 1 advskew 100 pass foo
```

/etc/hostname.carp1:

```
inet 192.168.0.1 255.255.255.0 192.168.0.255 vhid 2 advskew 100 pass bar
```

And of course, the following must be added to /etc/sysctl.conf on *both* firewalls:

```
net.inet.carp.preempt=1
```

Other uses

Although this article focuses on applications using both these components, they can also be used independently - pfsync can be used on its own where dynamic routing is used to handle failover, and CARP can be used on a cluster of servers rather than routers, to provide redundancy for a specific application (somewhat like the example

"Something bigger" above).

Future work

A number of enhancements are planned for both pfsync and CARP:

pfsync

- Configurable strong authentication for the various pfsync message types.
- Broader support for other structures used by pf (source-tracking nodes, tables, ALTQ, etc).
- Tighter integration with CARP.

CARP

- Replay detection for the cryptographically signed CARP packets.
- The ability to configure carp to use an interface with an address on a different subnet from the carp address, or even no address at all.
- Tighter integration with pfsync.

Additionally, a daemon which monitors network connectivity by link state transitions on network interfaces and performs active network tests is in the process of being developed. This daemon can be used to run local commands to reconfigure the system or start and stop services based on network events it detects.

Related Links

- The OpenBSD project
- The OpenBSD packet filter (pf)

Ryan McBride - mcbride@openbsd.org

Ryan McBride will be speaking on this and other pf-related topics at BSDCan in Ottawa, which runs May 13-16, 2004. BSDCan is a technical conference for people working on 4.4BSD based operating systems and related projects.

Countersiege

Home
Consulting