Global Information Assurance Certification Paper

CARP

The Free Fail-over Protocol

Pieter Danhieux
GSEC Practical v1.4b, Option 1
June 24, 2004

Table of Contents

Abstract

Availability is one of the three fundamentals of the CIA security model. It will become more and more important in the Information Security business since disasters of one form or another - such as terrorist actions, hurricanes or even riots - can have a serious impact on your business. Business Continuity Planning and Disaster Recovery Planning are already common practices in critical infrastructures.

On an operational level, organizations' servers, routers and all other network devices which provide a service need to be reliable. The Maximum Tolerable Downtime of a business unit is often over-estimated. This is why network architects should avoid single points of failure in a network infrastructure. They should provide reliable fail-over systems or high availability systems. CARP tries to provide a reliable and free fail-over system and it is doing a great job at that. This paper explains the design and implementation of CARP on OpenBSD 3.5, some sample setups in which CARP provides redundancy, and future improvements of CARP.

Terms and Definitions

CARP	Common Address Redundancy Protocol
Cluster	A group of nodes
HA	High Availability
Hardware address	An address inherent to the link layer device
HMAC	Keyed-Hashing for Message Authentication
HSRP	Hot Standby Router Protocol
Node	A system or device which is a member of a fail-over group
VRRP	Virtual Router Redundancy Protocol

Introduction

So why did the OpenBSD people design a new fail-over protocol when there's already VRRP? It all started in the late nineties when some parts of IETF's proposed standard (VRRP) were claimed[1] as patented by Cisco. Their redundancy protocol, HSRP, has some differences from VRRP, but the main similarity is the usage of virtual addresses which floated between the fail-over nodes.

The people at OpenBSD only want to implement *free* protocols in their operating system and they searched for an alternative for VRRP. While no other free alternatives were available they decided to design their own protocol and made sure it had some fundamental differences from HSRP. The inclusion of cryptography is one of the many discrepancies.

CARP Mechanism

CARP uses the technique of a virtual IP address floating around several CARP nodes. There is one master node and one or more slave nodes in the normal setup. The master node will always respond to request to the virtual IP address. The slave nodes will discard those requests; they just need to find out when the master node is unavailable and one of them must take over the master function whenever this happens.

For every setup, you need a virtual host id, a virtual IP address and a virtual hardware address. The form of the virtual hardware address depends on the type of hardware layer and is automatically created by CARP. The other two parameters need to be configured. For example:

Virtual host id (vhid):	1
Virtual IP:	192.168.0.100

Every CARP node in the cluster will have the same virtual hardware address, virtual host id and virtual IP address. Every node needs to have 3 extra parameters to make this work. The first two are the *advertisement base* and *advertisement skew* which both influence the interval (= advbase + (advskew / 255)) by which the advertisements are sent. The last parameter is the *password* used to authenticate advertisements.

Advertisement base (advbase):	1
Advertisement skew (advskew):	100
Password (pass):	changeme

[1] OpenBSD. 3.5: "CARP License" and "Redundancy must be free". May 2004. URL: http://www.openbsd.org/lyrics.html#35

Now let's create a CARP interface on a node which will be selected as the master node:

```
# ifconfig carp0 create
# ifconfig carp0 vhid 1 advskew 100 advbase 1 pass changeme 192.168.0.100
# ifconfig carp0
carp0: flags=41<UP,RUNNING> mtu 1500
        carp: MASTER vhid 1 advbase 1 advskew 100
        inet 192.168.0.100 netmask 0xffffff00
```

and on the node which will be slave:

```
# ifconfig carp0 create
# ifconfig carp0 vhid 1 advskew 110 advbase 1 pass changeme 192.168.0.100
# ifconfig carp0
carp0: flags=41<UP,RUNNING> mtu 1500
        carp: BACKUP vhid 1 advbase 1 advskew 100
        inet 192.168.0.100 netmask 0xffffff00
```

The master node in the CARP infrastructure sends out CARP advertisements with the highest frequency. This can be influenced by changing the advbase and advskew parameters. The function of the master is to reply to ARP requests for the virtual IP address with the virtual hardware address. This is the tcpdump output when an ARP request for 192.168.0.100 is made by a client machine:

```
# tcpdump -i ne1
tcpdump: listening on ne1
21:20:47.652452 VRRPv2-advertise 36: vrid=3 prio=0 intvl=1 (DF) [tos 0x10]
21:20:48.662442 VRRPv2-advertise 36: vrid=3 prio=0 intvl=1 (DF) [tos 0x10]
21:20:49.672469 VRRPv2-advertise 36: vrid=3 prio=0 intvl=1 (DF) [tos 0x10]
21:20:50.575462 arp who-has 192.168.0.100 tell laptop.test.lab
21:20:50.575505 arp reply 192.168.0.100 (0:0:5e:0:1:3) is-at 0:0:5e:0:1:3
21:20:50.682452 VRRPv2-advertise 36: vrid=3 prio=0 intvl=1 (DF) [tos 0x10]
21:20:51.692451 VRRPv2-advertise 36: vrid=3 prio=0 intvl=1 (DF) [tos 0x10]
…
```

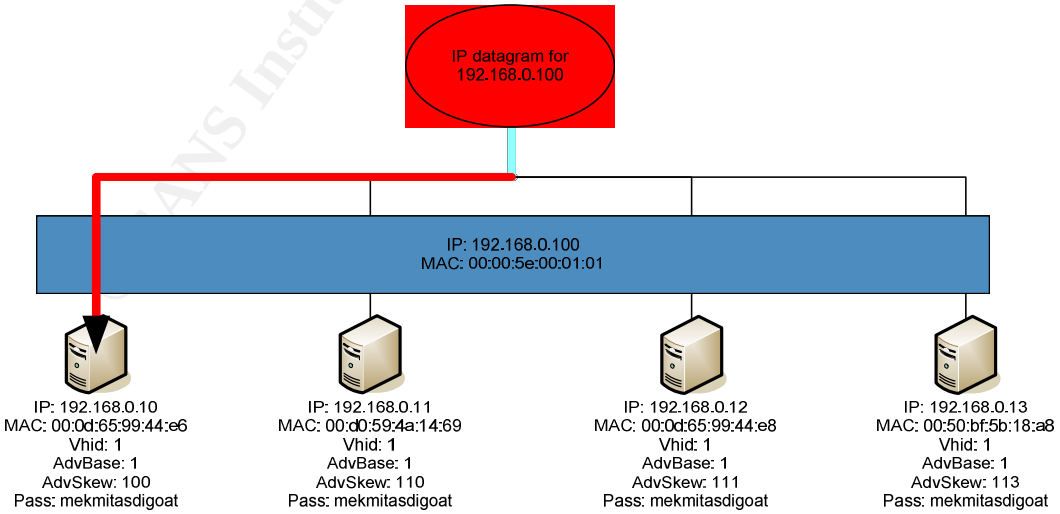


Figure 1: CARP infrastructure in a normal state

Every node on the infrastructure listens to the advertisements of the master node and checks if their advertisement interval is smaller than the one of the master. If for some reason the master node fails to send out advertisements, all slave nodes will notice this and will sent out an advertisement based on their own parameters. All the nodes will listen to the other advertisements, and the one with the highest frequency will take over the function of the master node.

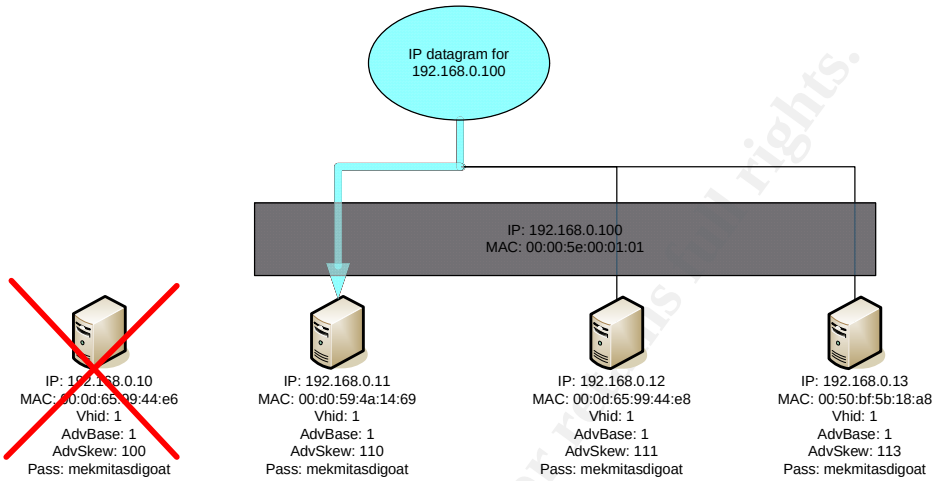


Figure 2: CARP infrastructure when master fails

If the original master node joins the CARP cluster again, it will be a slave node. The same election procedure will start and if configured correctly, it will become the master of the CARP infrastructure again.

Configuration on OpenBSD

There are two commands on OpenBSD which control the behavior of CARP on your system: s*ysctl(8)*[2] and *ifconfig(8)*[3]*.*

With sysctl you can control some of the behavior and logging of CARP.

Option	Description
net.inet.carp.allow	This options enables or disabled CARP on the system. This on by default, and necessary for the usage of CARP.
net.inet.carp.arpbalance	The arpbalance feature must be enabled if you want to use the load-balancing next to the fail-over function. You should create different virtual host id's for the same virtual IP on different nodes. If a request is made for the virtual IP, every node in the clusters will create a hash of the source IP and based on that hash the node which will answer is selected.
net.inet.carp.log	Log CARP errors via syslog. Can be useful for debugging.
net.inet.carp.preempt	Survival of the fittest, the node which can advertise the fastest, and thus the fit for the job, will be selected as master node.

To create a CARP interface, we need the ifconfig command:

```
ifconfig [carp interface] create
```

To configure the parameters on the CARP interface:

```
ifconfig [carp interface]   vhid [virtual host id]
                            advbase [advertisement base]
                            advskew [advertisement skew]
                            pass [Shared secret]
                            [virtual ip address]
```

[2] OpenBSD. sysctl(8) manual page. December 18, 2002. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&apropos=0&sektion=8&manpath=OpenBSD+3.5&arch=i386&format=html

[3] OpenBSD. ifconfig(8) manual page. September 3, 1998. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&apropos=0&sektion=0&manpath=OpenBSD+3.5&arch=i386&format=html

CARP Packet Format

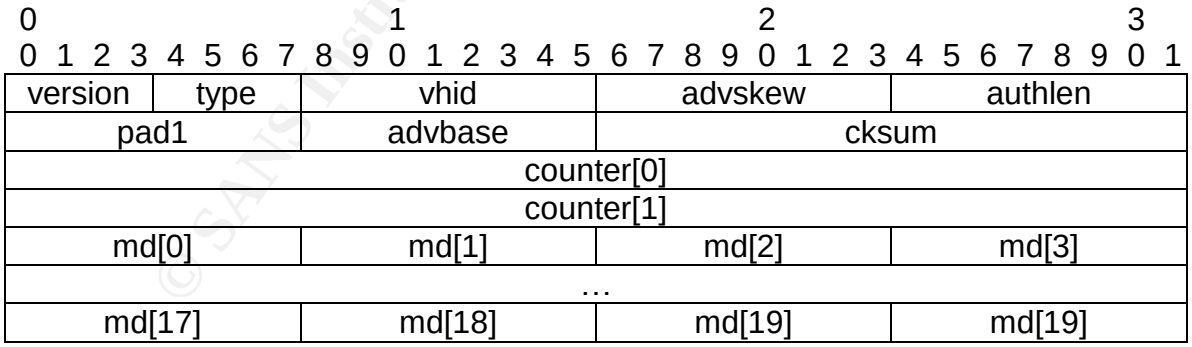
CARP has support for Ethernet, FDDI and Token Ring. The standard virtual hardware address for Ethernet and FDDI is 00:00:5E:00:01:XX (XX = virtual host id), and for Token Ring 03:00:XX:YY:00:00 (XX and YY are derivates from the virtual host id). A CARP packet is encapsulated in an IP datagram which has as source address the primary IP address of the interface through which the packet was sent, and has as destination the VRRP multicast address 224.0.0.18.

Similar to VRRP[4], the Time-To-Live value MUST always be 255. A CARP packet which does not use this TTL value is dropped immediately. The protocol number used is 112 (0x70), the same as VRRP uses. The request for an official IANA number was declined and therefore they were forced to choose a number which would not conflict with anything else.

Now, let's dissect the payload of a CARP packet. In the OpenBSD 3.5 CARP source code[5] you can find the structure of the CARP packet header.

```
struct carp_header {
  u_int8_t     carp_version:4,
               carp_type:4;
  u_int8_t     carp_vhid;
  u_int8_t     carp_advskew;
  u_int8_t     carp_authlen;
  u_int8_t     carp_pad1;
  u_int8_t     carp_advbase;
  u_int16_t    carp_cksum;
  u_int32_t    carp_counter[2];
  unsigned char carp_md[20];
};
```

Let's translate this into a readable packet format.



[4] R. Hinden, Ed. "Virtual Router Redundancy Protocol" RFC 3768. April 2004. URL: http://www.ietf.org/rfc/rfc3768.txt

[5] OpenBSD. "3.5 CARP source code: ip_carp.h". November 16, 2003. URL: http://www.openbsd.org/cgi-bin/cvsweb/src/sys/netinet/ip_carp.h?rev=1.4&content-type=text/x-cvsweb-markup&only_with_tag=OPENBSD_3_5

CARP Packet Fields

Field	# bits	Description
version	4	The version of the CARP protocol. This is statically defined as 2 in the header file /usr/src/netinet/ip_carp.h.
type	4	The type field defines the type of CARP packet. This value can be 0x01 (advertisement) or 0x02 (leave group), but the latter is only defined in the header file; I have not seen it being used anywhere.
vhid	8	Virtual host id.
advskew	8	Advertisement skew.
authlen	8	Size of Counter field + md field in 32 bit chunks. Statically defined as 7 in the header file.
Pad1	8	Unused, must be 0.
advbase	8	Advertisement interval.
cksum	16	Checksum for Internet Protocol family headers.
counter	64	Two counters used for replay detection. (not implemented yet)
Md	160	SHA-1 HMAC generated with the *pass* parameter as secret key, and counter, version, type, vhid, and virtual IP address as the message digest.

CARP features

Message authentication

Verifying the integrity and authenticity of information processed by fail-over systems is a prime necessity. If someone is able to alter the information in a message or replay the information of a message, it is possible that the fail-over system can be disorganized and this is a threat to the availability of your setup. For example, if a hacker is able to send fake advertisements on the network with a higher frequency than the master node and the master can not check the authenticity of the message, it will stop functioning and none of the slave nodes will take over his function which results in a Denial of Service attack on your high-availability system.

There are three key issues which CARP solves with their message authentication mechanism: Confidentiality, Integrity and Non-repudiation. CARP uses a HMAC SHA-1 scheme to check the integrity and authenticity of an advertisement. It also protects the data in the CARP packet by using symmetric encryption and thereby preserves the confidentiality of the information about the cluster such as the virtual IP address. The non-repudiation still needs to be implemented, the necessary fields in the packet are already available (counter fields) but they are not yet used.

Internet Protocol version 6

CARP was designed with support for IPv4 and IPv6 in mind, so CARP did not need any modification to run on an IPv6 network.

Load-balancing

CARP can be configured to not only provide a reliable fail-over system, but also load balance requests over different nodes in the cluster. Two CARP clusters should be created with the same virtual IP address and a different virtual host id. The sysctl arpbalance option mentioned in the *Configuration on OpenBSD* section must be enabled and this will make sure only one of the cluster nodes will answer the requests.

CARP vs. HSRP/VRRP

The big difference between HSRP/VRRP and CARP is not in the technique of virtual floating addresses, but in some features which were added to CARP such as supporting IPv6, adding a security control by using strong SHA-1 HMAC[6], and hiding the virtual IP address in the CARP advertisement.

	CARP	HSRPv0	VRRPv2
Support IPv6	Yes	No	In draft
Message authentication	SHA-1 HMAC	Clear-text password MD5 HMAC (v2 only)	Clear-text password MD5 HMAC
Patented	No	Yes	Patent claim

IPv6 Support

CARP is already supporting IPv6 on OpenBSD 3.5.

VRRPv3, which implements IPv6, is currently in draft[7].

Message Authentication

HSRPv0 only has an authentication field which contains a clear-text password, and thus does not really provide any security[8]. It must only be used in a testing environment for detection configuration mistakes such as timeout values or HSRP group issues and is definitely not designed to be used in a production environment. In version 2 of HSRP, Cisco provided HMAC MD5 authentication.

VRRP has three options for authentication: no authentication, a simple clear-text password and HMAC MD5.

CARP uses the strongest HMAC scheme by using SHA-1 hash algorithm which results in a 160 bit hash while MD5 has only an output of 128 bits and is thus more prone to collisions by brute forcing.

[6] H. Krawczyk, M. Bellare, R. Canetti. “HMAC Keyed-Hashing for Message Authentication” RFC 2104. Febuari 1997. URL: http://www.ietf.org/rfc/rfc2104.txt
[7] R. Hinden. “Virtual Router Redundancy Protocol for IPv6” draft, February 13, 2004. URL: http://www.ietf.org/internet-drafts/draft-ietf-vrrp-ipv6-spec-06.txt
[8] Li, T., Cole, B., Morton, P. and D. Li. "Cisco Hot Standby Router Protocol (HSRP)" RFC 2281. March 1998.URL: http://www.ietf.org/rfc//rfc2281.txt

Patented

This is the biggest and most important non-technical discrepancy between CARP and the two other HA protocols. HRSP is a proprietary protocol of Cisco and is patented under 5,473,599[9]. VRRP is a complicated case. Two patent claims were made on this technology, one from Cisco - Standby Routing Protocol 5,473,599 and one from IBM - Cluster patent, 5,371,852[10].

The IETF concluded that patented technology in a standard is allowed, as long as it is licensed under reasonable and non-discriminatory terms, so they published VRRP as a new standard.

The OpenBSD developers did not want to implement a standard from IETF which had patent claims on it, especially because Cisco's lawyer informed them that Cisco would defend its patent for VRRP implementations[11]. So they designed and implemented CARP and released it under the BSD license[12].

[9] "Patent 5,473,599". December 5, 1995. URL: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=5,473,599.WKU.&OS=PN/5,473,599&RS=PN/5,473,599

[10] "Patent 5,371,852". December 6, 1994. URL: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=5,371,852.WKU.&OS=PN/5,371,852&RS=PN/5,371,852

[11] OpenBSD. 3.5: "CARP License" and "Redundancy must be free". May 2004. URL: http://www.openbsd.org/lyrics.html#35

[12] OpenBSD. "Copyright Policy". URL: http://www.openbsd.org/policy.html

CARP caveats

It is impossible to write the perfect fail-over system, and certainly on such a sort period as the OpenBSD team developed CARP. That is why I think there are some features missing in CARP (and some of them are already scheduled to be implemented). These remarks are based on my personal experience with CARP on OpenBSD 3.5 on i386 architecture only.

There is no notification to userland if the state of the CARP interface changes. If a master node goes down and a slave node takes his function over, a server should know what his function is and maybe stop or start some services. It's always possible to create a script which checks the status of the interfaces regularly, but this is less reliable than userland interaction.

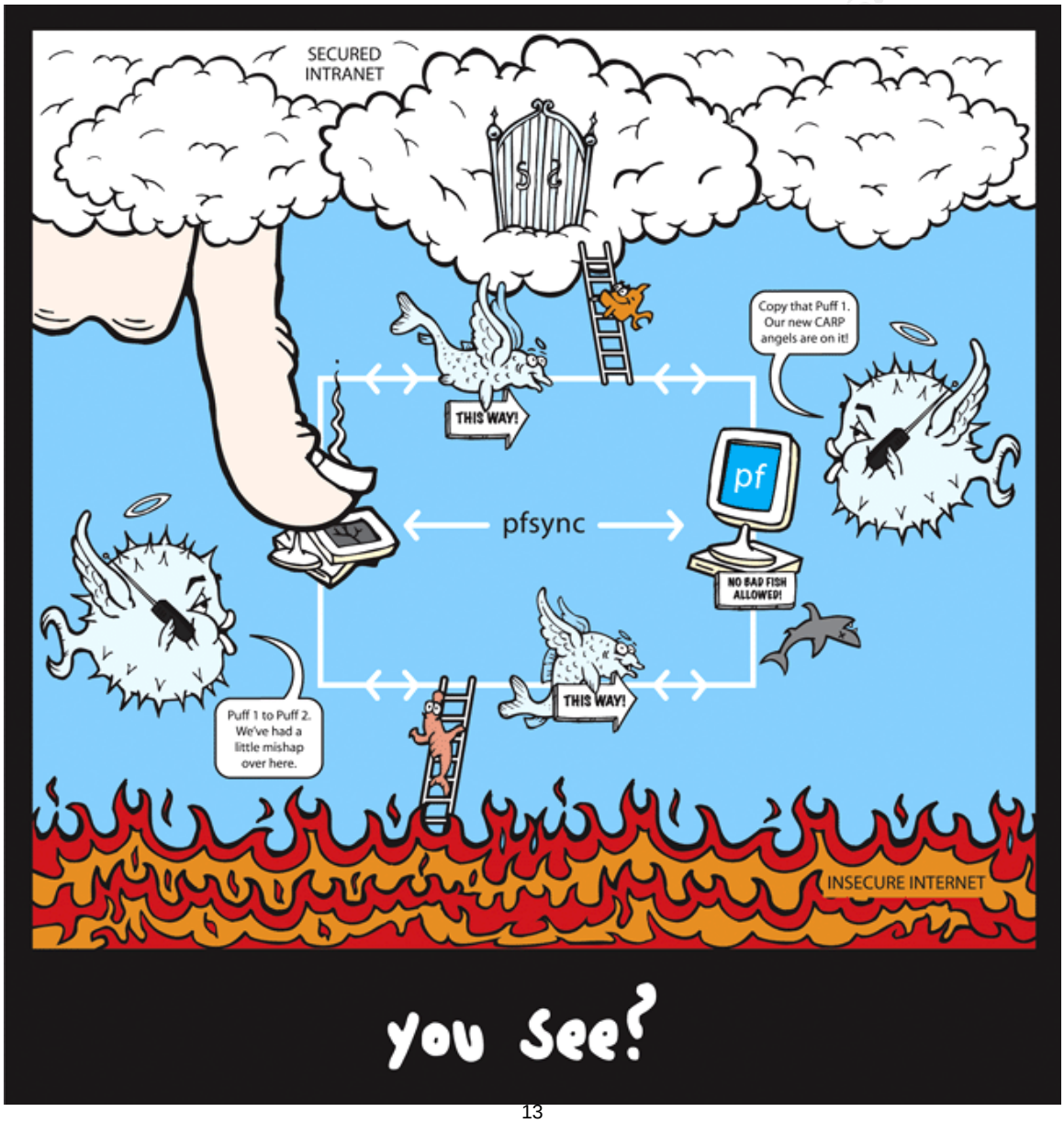
It's also impossible with CARP to link a virtual IP address to an interface in another IP range. For example if you have a webserver on a public IP address, and you want to create a cluster with 2 nodes to create a reliable fail-over system. You cannot assign the public IP address as a virtual IP to the webservers if their physical network interfaces have a private IP address. They MUST have an IP in the same range of the public IP address to make this work.

For some reason it is impossible to tcpdump the traffic on the carp interface when it's in MASTER mode, which could come in handy when you need to know what network traffic is destined for the virtual IP address.

CARP has been implemented only on several free operating systems, such as OpenBSD, FreeBSD, NetBSD and Linux. Unfortunately, none of the commercial vendors have shown interest to implement CARP on their operating systems, which means that CARP can not interact with other operating systems as those mentioned above.

Conclusion

While CARP is still young and needs to be tested in large setups, it is already a good, patent-free alternative for HSRP or VRRP. Like every young software project it's only in the first stage of its life-cycle, and the grand mass is now testing it. This will hopefully result in bugfixes, improvements and new features which can help the OpenBSD team in making CARP the *perfect* alternative.



13

[13] OpenBSD, "Number 21: the 3.5 CARP tshirt". URL: http://www.openbsd.org/tshirts.html#21

References

OpenBSD. 3.5: "CARP License" and "Redundancy must be free". May 2004. URL: http://www.openbsd.org/lyrics.html#35

OpenBSD. sysctl(8) manual page. December 18, 2002. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&apropos=0&sektion=8&manpath=OpenBSD+3.5&arch=i386&format=html

OpenBSD. ifconfig(8) manual page. September 3, 1998. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&apropos=0&sektion=0&manpath=OpenBSD+3.5&arch=i386&format=html

R. Hinden, Ed. “Virtual Router Redundancy Protocol” RFC 3768. April 2004. URL: http://www.ietf.org/rfc/rfc3768.txt

H. Krawczyk, M. Bellare, R. Canetti. “HMAC Keyed-Hashing for Message Authentication” RFC 2104. Febuari 1997. URL: http://www.ietf.org/rfc/rfc2104.txt

R. Hinden. “Virtual Router Redundancy Protocol for IPv6” draft, February 13, 2004. URL: http://www.ietf.org/internet-drafts/draft-ietf-vrrp-ipv6-spec-06.txt

Li, T., Cole, B., Morton, P. and D. Li. "Cisco Hot Standby Router Protocol (HSRP)" RFC 2281. March 1998.URL: http://www.ietf.org/rfc//rfc2281.txt

“Patent 5,473,599”. December 5, 1995. URL: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=5,473,599.WKU.&OS=PN/5,473,599&RS=PN/5,473,599

“Patent 5,371,852”. December 6, 1994. URL: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=5,371,852.WKU.&OS=PN/5,371,852&RS=PN/5,371,852

OpenBSD. “Copyright Policy”. URL: http://www.openbsd.org/policy.html

Jeremy Andres. “KernelTrap - Interview: Ryan McBride”. April 7, 2004. URL: http://kerneltrap.org/node/view/2873

Ryan McBride. “Firewall Failover with pfsync and CARP” Countersiege. URL: http://www.countersiege.com/doc/pfsync-carp/

OpenBSD. “3.5: Programmer’s Manual”. October 16, 2003. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=carp&apropos=0&sektion=0&manpath=OpenBSD+3.5&arch=i386&format=html

OpenBSD. “3.5 CARP source code: ip_carp.c”. March 26, 2004. URL: http://www.openbsd.org/cgi-bin/cvsweb/src/sys/netinet/ip_carp.c?rev=1.44&content-type=text/x-cvsweb-markup&only_with_tag=OPENBSD_3_5

OpenBSD. “3.5 CARP source code: ip_carp.h”. November 16, 2003. URL: http://www.openbsd.org/cgi-bin/cvsweb/src/sys/netinet/ip_carp.h?rev=1.4&content-type=text/x-cvsweb-markup&only_with_tag=OPENBSD_3_5

OpenBSD, “Number 21: the 3.5 CARP tshirt”. URL: http://www.openbsd.org/tshirts.html#21

Last Updated: June 26th, 2017

Upcoming Training

SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
Community SANS San Diego SEC401	San Diego, CA	Aug 21, 2017 - Aug 26, 2017	Community SANS
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS