Zürcher
Hochschule
Winterthur

Mitglied
der Zürcher
Fachhochschule

Communication and Information Technology

High Available Security Gateway

with Debian GNU/Linux 3.0 on x86 architecture

Lecturer	Prof. Dr. A. Steffen
Team	E. Marchionni T. Schneider
Submission date	20.02.2004
Version	1.0

Executive Summary

Virtual Private Network is getting an important topic more and more. One of the most important components of such a VPN is the VPN Gateway which handles all the connections from the clients or other sites.
As soon as this service has to be available for a defined period «high availability» is the buzzword. There are some commercial products around, «Cisco VPN 3000 Series Concentrator» are probably the most popular products which can be implemented with a redundant configuration but they are also quite expensive (for a redundant configuration which can handle a encryption throughput of 100Mbit one would have to pay about 80'000$ or more). Many features of such a commercial solution can also be provided by a Linux based solution with the help of some additional free open source projects.
Therefore this document should give you an overview of how to set up a high available security gateway and accentuate some points which should be considered while planning the implementation of such a setup.
The document is segmented into four parts:

The chapter *Introduction* contains the conceptual formulation and differentiation. In the chapter *Analysis* one finds an overview about the basic components and ideas of a high availability security gateway. The next chapter deals with our *Implementation*. And last but not least the *Howto* gives a user a basic Linux knowledge the possibility to setup such a HA security gateway.

We archived to set up an implementation which is able, in case of a failure, to take over from the master to the slave node in about 25 seconds. This is enough fast that most client applications don't timeout. Though the condition for a seamless take over is, that the VPN client is supporting the feature called «dead peer detection».

Explanation concerning the independent writing of this work

With giving of this documentation the students insure that they have written the work independent and unassisted.

Place, Date:	Signatures:
....	....
	....

Table of Contents

1 Introduction

1.1 Preface

We wanted to figure out an implementation which is able to do a quick take over, is able to handle a high load and also is attached redundantly to the network.

At this point we want to thank Hewlett-Packard Switzerland for their generous support. We got two HP ProLiant DL380 G3 servers for the duration of our project, with which we were able to realize our formulated goals.

The original version of this document can be found on the homepage[1] of the «ZHW Security Group».

1.2 Formulation

Secure network communication (SNK)

project work in winter semester 03/04 - PAKI1 Sna01

High Available Security Gateway

students:

- Eric Marchionni, KI3d
- Thomas Schneider, KI3d

dates:

■ expenditure:	Tuesday 18th November 2003
■ submission:	Friday 20th February 2004

A firewall separates an enterprise network against attacks from the Internet. A security gateway however permits the external access to the company resources to entitled users over a cryptographically secured VPN tunnel. In order to be able to guarantee the VPN service around the clock and 7 days a week, the security gateway must be laid out redundantly. The goal of this project is to practically examined, how one can develop a high-available security gateway with economical PC hardware and free-available Linux open source software. Current configuration data are to be mirrored thereby. Further has to be theoretically clarified, with which mechanisms the existing VPN tunnel could be taken over interruption-free by the redundant gateway including the momentary session key. Also the possibilities of load balancing in a gateway cluster are to be examined.

tasks:

- study of the Linux High-Availability mechanisms
- installing and testing a redundantly gateway-cluster
- realization of the automated mirroring of the configuration files
- analysis of possible load balancing mechanisms

1 http://security.zhwin.ch/ha_security_gateway.pdf
http://security.zhwin.ch/ha_security_gateway_config.tar.gz

- clarify of the interruption-free takeover of VPN sessions
- creation of a howto for installing a redundantly VPN gateway

infrastructure:

■ room:	E523
■ hardware:	4 computers
■ software:	Linux HA, FreeS/WAN

literature / links:

- High-Availability Linux
  http://www.linux-ha.org
- High-Availability FreeS/WAN
  http://www.freeswan.ca/lk2003/paper/kbantoft-evpn.pdf
- FreeS/WAN Download
  http://www.freeswan.ca
- FreeS/WAN with X.509 certificates
  http://www.strongsec.com/freeswan/install.htm
- Linux advanced routing and traffic control
  http://www.lartc.org

Winterthur, 17. November 2003

a. steffen

Prof. Dr. Andreas Steffen

1.3 Differentiation

We do not cover the following topics in this documentation:

- logging
- alerting
- longtime tests
- evaluation of Windows VPN clients
- implementation of a load-balancing mechanism

2 Analysis

2.1 Openswan

2.1.1 Description

Openswan is an implementation of IPSEC for the Linux operating system. It's a code fork of the FreeS/WAN project and includes already the patches for X.509 digital certificates and more.

IPSEC means *Internet Protocol SECurity* and is a ITEF standard. It allows to build secure tunnels through untrusted networks. All data passing the untrusted network are encrypted by the IPSEC gateway and decrypted by the gateway on the other side. The gateway also takes care of the authentication. The resulting network is indeed private and called VPN for *Virtual Private Network*. It can include hosts from many different places all around the world connected by the insecure Internet.

The main features of IPSEC are the following:

- strong encryption (both integrity and privacy are guaranteed)
- IKE-based authentication

Unlike many other security standards such as SSL (transport layer) or PGP (application layer) IPSEC builds on the low network layer and is therefor largely platform and application neutral. This way a lot of different kind of network communication can be encrypted transparently.

2.1.2 VPN types

There exist different types of VPN tunnels. The two most common ones are described below:

Site-to-Site

As its name already reveals this architecture basically is used to connect two networks which are located on different geographic places together. In practice this is primarily used to connect branch offices to the head quarter of an enterprise. The picture below exemplifies how it works.

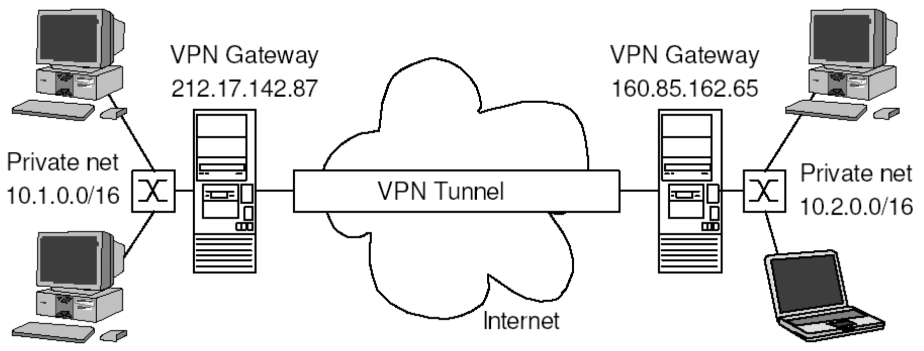


Illustration 1: site-to-site VPN

Roadwarrior

Remote access VPNs, also known as Roadwarrior, aspire to provide remote secure access to data through the Internet to any arbitrarily point. This gives for example an employee the possibility to connect to the companies network getting access to information no matter if he is working at home and/ore remote. Because the IP address is assigned dynamically by the ISP the configuration differs from the one of a site-to-site VPN. Also there has to be given a virtual IP to the Roadwarrior to avoid routing problems. The remote access scenario is the one we're going to focus on in this document.

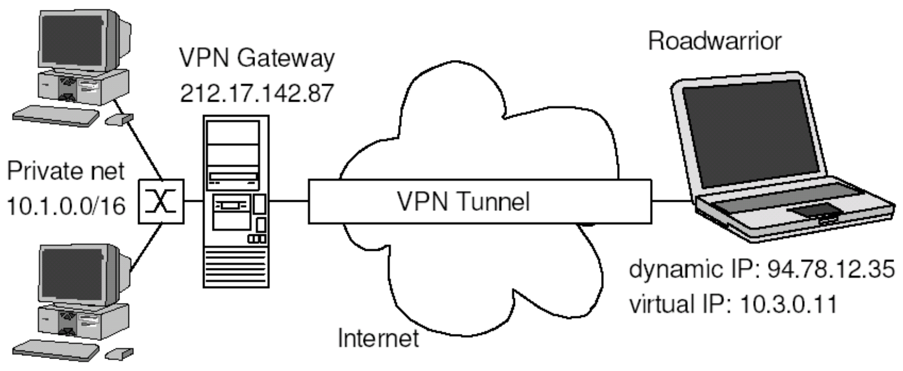


Illustration 2: remote access scenario

2.2 heartbeat

2.2.1 Description

Heartbeat is a loss management software. This package monitors hosts and informs the cluster when one of them dies and then starts the disaster management.
So far heartbeat supports only two nodes in a cluster. While one node is the master and has services running the other node is in the hot-standby mode. In addition to the nodes normal IP addresses a virtual IP address is created and assigned to the master node. Clients now connect to the virtual IP address and use services from the master. A so called heartbeat is being sent over ethernet or serial cable every few seconds from one node to the other to check if the master is still available. In case of a breakdown of the master the slave node which is in hot-standby takes over the virtual IP address and starts the services. Therefor the services are still provided on the same IP address to the end users.

The following graphic gives a simple overview:

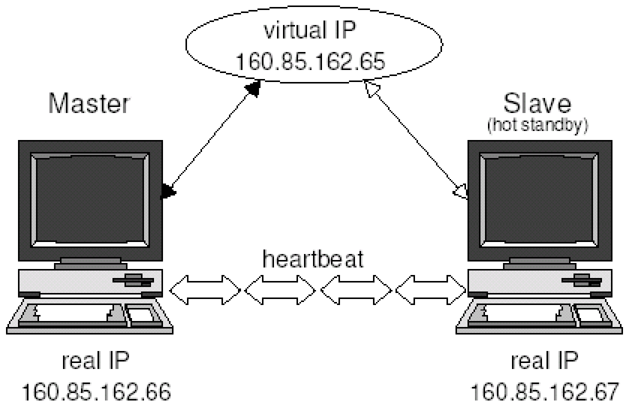


Illustration 3: heartbeat

2.2.2 IP address takeover

Basically there exist three different approaches to a network failover:

Basicaly three different approaches exist to a network failover:

- IP address takeover
- MAC address takeover
- dynamic DNS reconfiguration

Each of these methods has its advantages and disadvantages which result in different takeover times and stability: MAC address takeover is known to happen almost instantly but being a little bit messy. IP address takeover is almost as fast while being a bit more reliable. The dynamic DNS reconfiguration surely is slower but provides some interesting load balancing features. Since heartbeat uses code from the fake project which is based on a IP address takeover we'll focus on this method.

Heartbeat creates an logical network interface (IP Alias) on the active node to which the virtual IP address is assigned to. IP address takeover happens by the hot standby machine bringing up a logical interface with the virtual IP address. To ensure that all traffic now is being sent to the former slave, heartbeat has to take care of the ARP cache on all routers and hosts in the network. No matter if the broken down master still is able to do ARP replies or not, all the traffic won't reach its destination as long as the wrong hardware address can be found in the ARP caches and the entry hasn't expired.

So what heartbeat does is making use of a technique called *gratuitous ARP*. This is an ARP request for the own IP address. The result beside experiencing if there is a second machine with the same IP in the net is that all the hosts reached by the broadcast ARP request update their ARP caches with the received IP. If the slave sends gratuitous ARP often enough (before the entry in the hosts and routers caches expires) no ARP requests for the virtual IP will be sent out what makes it impossible that the master sends out ARP replies in case of still having a working logical network interface.

Since gratuitous ARP can be used by an attacker to maliciously take over an IP address of a host some switches and routers ignore gratuitous ARP. For heartbeat taking over an IP address successfully the whole broadcast domain has to be configured to accept gratuitous ARP!

2.2.3 ipfail

ipfail is a plugin which constantly checks if a so called *ping node* still is reachable. If not the plugin contacts the other node in our cluster and looks if the ping node is also unreachable. If this isn't the case some hardware failure between the master and the ping node occurred. This may be an interface on the master or any other network component on the route to the ping node. Then ipfail causes heartbeat to fail over.

2.3 Synchronizing configuration

There are a few different ways to duplicate configuration files which need to be the same on two machines:

- manual
- rsync
- drbd

Manual procedures have the disadvantage that they most likely get forgotten or an uninformed person isn't even aware of the procedure. The result is the same: your system gets messed up pretty fast.
Rsync is a tool for efficiently transferring files across a network. It does it's job very good and is widely known for its usability in backup scripts.
When it comes to realtime duplication of data no way leads past at *drdb*! DRBD means *Distributed Replicated Block Device* and offers the possibility to mirror file systems over a network in realtime. Think of it as a network raid-1.

Since our configuration changes rarely a cronjob which is copying the requisite files is sufficient. The use of drbd would be overkill and the installation and configuration would unnecessarily complicate things.

2.4 Load-balancing

Load-balancing is a mechanism which distributes the load of two or more servers if one server is not enough to handle the load. Normally a load-balancing implementation is also providing some kind of high availability, because it's still working if one of the multiple nodes fails.
To provide high availability the load-balancing unit itself hast to be set up redundantly.
The first possibility to do this, is to have a redundant «routing device» which routes the VPN connections alternately to two or more VPN Gateways and in case of a failure it would automatically detect the failing node and route all connections to the remaining node(s). This implementation could have the disadvantage, that it isn't any longer capable to handle a high load.
The second possibility (and also more expensive one) would be to implement redundant nodes, so that every node (from point of view of the load-balancer) is a cluster itself with a master and slave node.
Both of this possibilities is causing a more complicated setup which is harder to administrate.

Hence we only recommend to use load-balancing for a VPN gateway if you are not able to handle the load on one server. Before you have to handle more than 100Mbit encrypted data throughput you need a accordingly connection to the Internet.

Before thinking about an redundant and load-balancing VPN gateway it's probably easier and cheaper to invest into more powerful hardware. It's no problem to handle 100Mbit encrypted throughput with modern x86 server hardware.
If you still want to implement a load-balancing setup we recommend to take a look at the free open source projects «Linux Virtual Server Project»[2] and «Ultra Monkey»[3]

2.5 VPN session takeover

In case of a master breakdown the slave takes over the part of the master node. But the incoming traffic from the Roadwarriors is encrypted with a session key only the master knows. Since this session key for security reasons isn't saved in a file one cannot simply copy the session key to the slave. The session key is only present in the memory! Further investigations showed that the whole connection negotiation basically is handled in the file *openswan-1.0.0/pluto/connections.c* . So one approach to this issue would be to write a patch which adds additional functionality to ipsec such as duplicating the session key to a remote ipsec instance. This would go far beyond the scope of our project.

The cleanest way to do this is to use the so called *dead peer detection* on the client side. In this method the IKE SA works through ping packets. If enough keepalive packets don't get a reply the death of the gateway is assumed. Now if the VPN client software supports this feature a reestablishment of the connection can by achieved. Except the little timeout pretty much what we are looking for!
Sadly VPN clients supporting DPD are very rare for Windows operating systems. However with the Openswan VPN client everything works well.

2 http://www.linuxvirtualserver.org/
3 http://www.ultramonkey.org/

3 Implementation

3.1 Overview

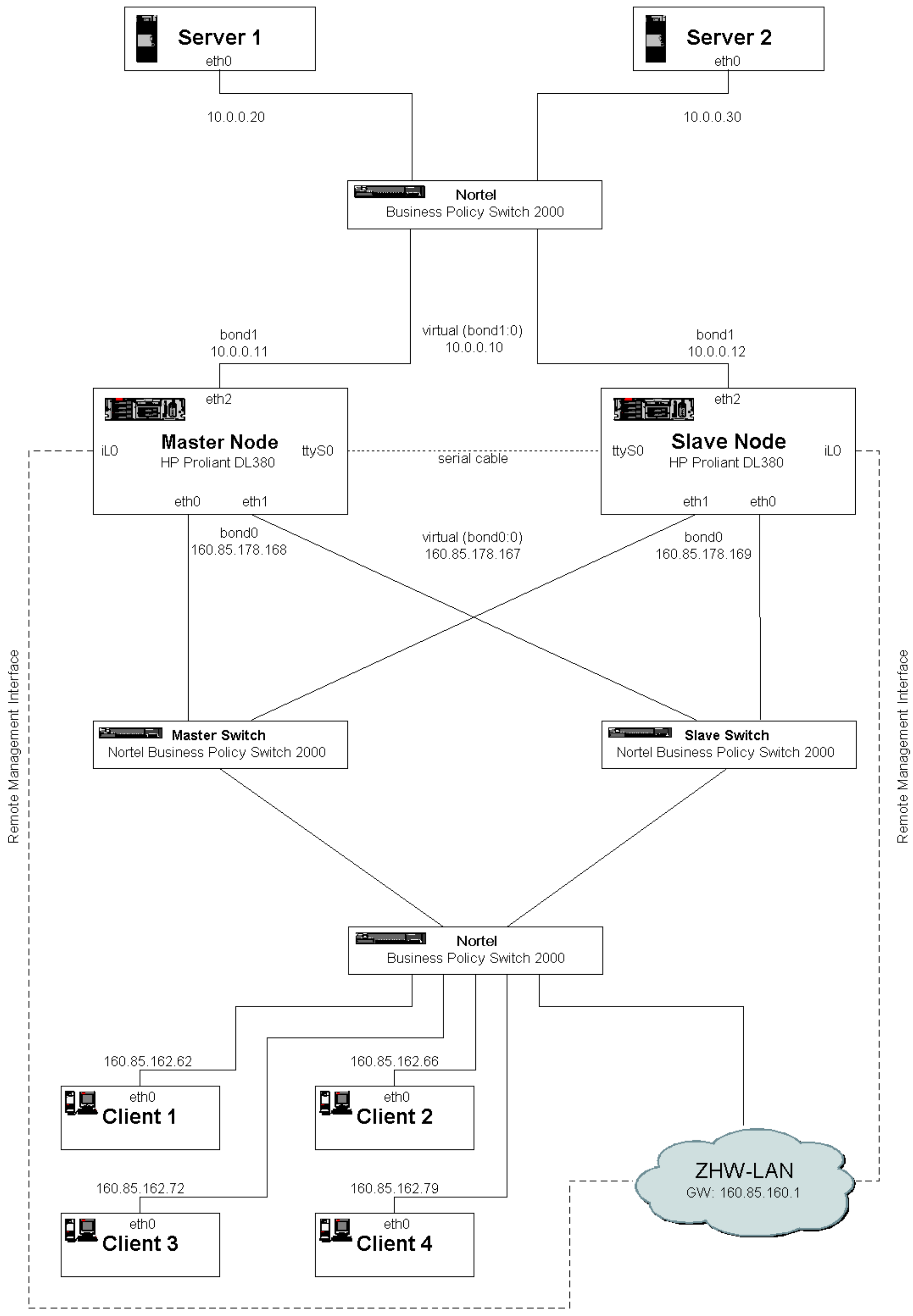


Illustration 4: network scheme

3.2 Our setup

3.2.1 Hardware

For the following setup, two computers with at least two network interfaces are needed. Because ipsec is quite demanding depending computing power we propose you not to take the slowest hardware available. But of course it depends on how many connections at which speed the system has to handle. The chapter «Performance tests» is maybe helpful to get an impression what kind of hardware you need.

Concerning the network switches you can take almost any product for this setup. There are no special features needed because the fail over in case of a network failure is handled by high available security gateway itself.

We used the following hardware components:

Pieces	***Used as***	***Model***	***Description***
1	Master node	HP ProLiant DL380 G3	3.06 GHZ Xenon, 1024MB Ram, 2x 1Gbit NIC, 1x 100Mbit NIC, 2x 36GB U320 HD
1	Slave node	HP ProLiant DL380 G3	2.8 GHZ Xenon, 512MB Ram, 2x 1Gbit NIC, 1x 100Mbit NIC, 2x 15GB U320 HD
4	Roadwarrior Clients	Generic Hardware	450 MHZ PII, 512MB Ram, 1x 100Mbit NIC
2	Server	Generic Hardware	450 MHZ PII, 512MB Ram, 1x 100Mbit NIC
4	Switches	Nortel Business Policy Switch 2000	

Table 1: hardware

3.2.2 Software

When we were discussing about the Linux distribution to use we were not able to arrange us :-)

So what we did was to implement a benefit-value analysis. Here is what we got:

criterion	***emphasis***	***Debian (stable)***		***Gentoo***		***SuSE***	
		X	B	X	B	X	B
Topicality	20	1	20	5	100	3	60
Our own skills	10	5	50	5	50	3	30
Updates / security	15	5	75	5	75	1	15
Independence	15	5	75	5	75	1	15
Stability	15	5	75	3	45	3	45
Packet manager	20	5	100	5	100	3	60
Documentation / support / mailing-lists	5	3	15	5	25	3	15
Total benefit	100		410		470		240
Preference order of the alternatives		Rank 2		**Rank 1**		Rank 3	
X = valuation (good = 5, pleasable = 3, bad = 1) B = benfit per distribution							

Table 2: Linux distribution

Of course these are our own percepts and others might rate the criteria different. Nevertheless it allowed us

to commit ourself to a distribution.
But when we first tried to install Gentoo 1.4 on the machines provided by the university we experienced that the *Live CD* had problems to load the driver for the SCSI hardware and got stuck in the boot process. Even with older releases of Gentoo it didn't work.
Since Debian ended up with only few less points than Gentoo in the analysis the choice wasn't difficult. And what wonder: everything went just fine with Debian Woody.
So besides two software packages which were simply too outdated in the Debian package database everything comes with Woody and can easily be installed with apt. This is the final shopping... aehm download list:

- Debian Woody 3.0[4]
- Openswan 1.0.0[5]
- heartbeat[6]

3.3 High available security gateway

3.3.1 VPN cluster

First of all we installed a slim Debian installation on the two ProLiant DL380 G3 servers sponsored by HP. With *apt* we post installed some needed tools such as *openssl*, *rsync*, *wget* and more. As already mentioned above the FreeS/WAN and Heartbeat packages in the stable tree of Debian are totally outdated. Because of that we installed newer versions of those programs manually. As just Openswan 1.0.0 got released in this time we switched from FreeS/WAN to Openswan in the very beginning of our project work because it supports X.509 certificates already and DPD “out of the box”. FreeS/WAN would have needed a patch.
When we got ipsec running we implemented Heartbeat. Though a basic configuration was set up pretty fast we were missing some features. For example if the master node lost connectivity to the internet the heartbeat still was interchanged every two seconds between master and slave. And for this reason heartbeat didn't failover although the service wasn't available to the clients anymore. Further investigations brought us to ipfail, a module for heartbeat. ipfail checks the availability of specified ping nodes and if the master doesn't see this ping node anymore but the slave still is able to ping the host a takeover is initiated. Similar behavior can be found in HA pairs from Checkpoint.
Another issue was the mirroring of configuration files. A public key authentication for ssh was realized and we stated in a shell script the files to be copied with rsync via a ssh tunnel. We decided that it will be sufficient to run that script every ten minutes and configured a cronjob for that.
At the end we did some tweaking like disabling unnecessary services and configure ntpdate for having the same time on both machines what is useful for debugging purposes (logfiles).

3.3.2 Redundant network attachment

To have a real high available security gateway it's also important to connect the master and slave node

4 http://www.debian.org/CD/http-ftp/
5 http://www.openswan.org/code/
6 http://www.linux-ha.org/download/

redundant to the network. Because we had not enough switches we only implemented the redundant network connection to the external network. But there is nothing speaking against doing the same also for the internal network attachment.

Linux is providing the necessary tools with the actual kernels (since 2.4.12). The feature is called «Bonding driver support»[7] and provides different possibilities to profit by adding more than one network interface. The «bonding» is transparent to all applications.

You can either set up a trunk to double your speed (this has to be supported by the opposite component) or configure a active-backup bond (there are also some other modes available) which uses only one of two or more network interfaces and fails over to the next interface including a take over of the MAC address when the primary interface fails.

We implemented the active-backup node mode because it fits our needs (high availability) best.

In this mode, only one slave interface in the bond is active. A different slave interface becomes active if, and only if, the active slave interface fails. The bond's MAC address is externally visible on only one port (network adapter) to avoid confusing the switch.

It is important that the slave node is using the slave switch as primary device (see chapter «Overview»), because the other way round it could lead to a deadlock. If the master switch itself is working and for example a link between the gateway and master switch is down but between slave switch and gateway every thing is working, Heartbeat would not fail over because the slave node would also use the master switch as primary device.

3.3.3 Dead peer detection

To permit a client the reestablishing of the connection to the gateway cluster in case of a failover without any user interaction we enabled DPD into Openswan. We specified a timeout of 15 seconds. Though one could minimize this time it wouldn't affect the over-all timeout because the VPN client has to wait for heartbeat setting up the virtual interfaces, sending out gratuitous ARP request and starting openswan on the slave.

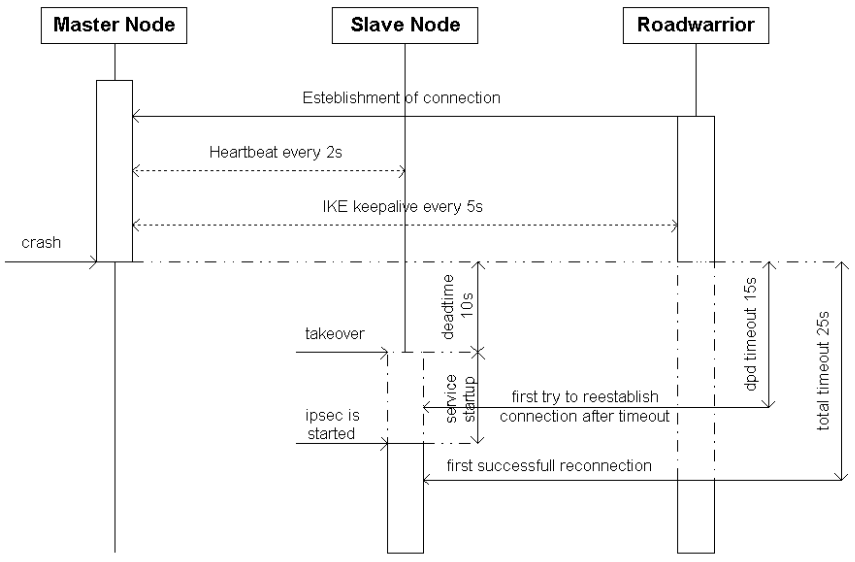


Illustration 5: sequence diagram takeover

7 /usr/src/linux/Documentation/networking/bonding.txt

The sequence diagram above makes it clear.

Since dead peer detection[8] hasn't even the status of a *RFC* yet it's not astonishing that there aren't many VPN client software with DPD available for Windows at the moment.

3.4 Performance tests

We did some performance tests with our implementation, but they were not done in a scientific way. They should just give a brief overview of what this implementation is capable and show the difference between 3des and AES encryption concerning load.

We took the program «top» to observe the load, that's why we always give a range of the load and not a exact number.

We were not able to fully load the master or slave node because our clients and servers were to weak to generate enough traffic. For this reason we also took one of the nodes as client for some of the tests.

The load was generated by up- and downloading a 650MB ISO file via ftp.

No.	***VPN GW***	***Server(s)***	***Client(s)***	***Algo.***	***Down speed***	***Up speed***	***Load on GW***
1a	Master node	Server01	Slave node	3des	10.69 MB/s		45-50% (total 10.7 MB/s)
1b	“	“	“	AES128	10.46 MB/s		30-35% (total 10.5 MB/s)
1c	“	“	“	AES256	10.59 MB/s		30-40% (total 10.6 MB/s)
1d	“	Server01 Server02	Slave node Client01 Client02	“	10.31 MB/s	2.68 MB/s 2.66 MB/s 5.34 MB/s	40-50% (total 15.65 MB/s)
1e	“	Server01 Server02	Slave node Client01 Client02 Client03 Slave node	“	3.91 MB/s 2.39 MB/s 2.42 MB/s 2.20 MB/s 10.9 MB/s	3.53 MB/s	40-50% (total 14.5 MB/s)
2a	Slave node	Server01 Server02	Client01 Client02 Client03 Client04	AES256	2.95 MB/s 2.95 MB/s 2.57 MB/s 8.47 MB/s	2.65 MB/s	40-50% (total 11.1 MB/s)
3a	Slave node	Server01 Server02	Master node Client01 Client02	AES256	10.7 MB/s	2.84 MB/s 2.89 MB/s 5.73 MB/s	45-55% (total 16.5 MB/s)

Table 3: performance tests

As you can see the maximum total speed we got, was about 16.5MB/s. More was not possible because the

8 http://www.ietf.org/internet-drafts/draft-ietf-ipsec-dpd-04.txt

servers providing the ftp daemon could not handle more traffic (hard disks to slow). At this speed the load of the slave node was sometimes near 60%. Hence we believe that one could handle 100Mbit fullduplex with a Xenon 2.8 GHZ.

We didn't have the possibility to test how openswan would have scaled with a dual CPU setup.

During this tests we also simulated a network failure relating the master node only. The take over to the slave node passed without any problems and the ftp transfers continued after a timeout of about 25 seconds.

4 Howto

This Howto describes a general set up of a HA VPN Gateway Cluster with Debian GNU/Linux 3.0 on a x86 basis. Special steps needed to set up a HP ProLiant DL380 G3 server are marked specially. We expect some general knowledge about setting up a Linux system from the reader of this document. A general Debian installation manual can be found on the Debian homepage.[9]
Without any further instructions the following steps are to be done on both, the master and the slave node.

4.1 Basic configuration

Note: We propose to read the «Securing Debian Manual»[10] for a more secure and hardened Debian installation.

4.1.1 BIOS

DL380 Note (BIOS): Enter BIOS of DL380 and change “System Options” > “OS Selection” to “Linux”.
When working with a Xenon 3.06 GHZ or higher disable “MPS Table” according to the message appearing while changing “OS Selection”. After the Installation of Debian you can enable “MPS Table” again.

4.1.2 Starting

After you booted from the Debian CD No. 1 type “bf24” on the boot manager prompt to start the installation with a 2.4 kernel.

4.1.3 Partitioning HD

We chose the following partitions for our nodes:

Mounting Point	Size	File system
/boot	16 MB	ext2
swap	1024 MB	
/	rest	ext3

Table 4: partitioning

DL380 Note (RAID): We propose to set up a hardware RAID 1 (eventually with a hot spare) for the system using the integrated Smart Array Controller 5i (see HP documentation for further details).
The device for the first array on this controller is /dev/cciss/c0d0

4.1.4 Configuration parameters

We set up the both nodes using the default values proposed by the Debian installation expect the following

9 http://www.debian.org/releases/stable/installmanual
10 http://www.debian.org/doc/user-manuals#securing

ones:

- enable md5 passwords
- mail system: 4 “local delivery only”

We also set up the network and added the adequate apt sources.

DL380 Note (Network): The needed network drivers for the DL380 are not available from the Debian 2.4 kernel. We finished installation choosing CD-ROM as source for apt and didn't set up the network at this stage.

4.1.5 Needed packages

With taskselect we chose “conventional unix server” and added with dselect the following additional packages: gwak, bzip2, libncurses5-dev, openssl and wget.
Some of this packages are needed for the following kernel modifications.

4.2 Kernel

Download kernel 2.4.24 or transfer it with a CD-ROM to your nodes into the directory /usr/src and prepare it for the configuration:

```
# tar xvjf linux-2.4.24.tar.bz2
# ln -s linux-2.4.24 linux
# cd linux
# make menuconfig
```

We chose most time the default configuration and disabled not needed options like “NFS support”, “power management”, “sound”, “AGP” and “ISA support”.
We enabled “High Memory Support” (needed if you want to use more than 900MB RAM), “SMP support” (also enable it, if you are using CPU's which are capable of «Hyper Threading»), “Advanced Router”, “Iptables” (with the necessary modules), “ext3 file system support”, the appropriate network card modules, SCSI support including the right adapter (compile your kernel including SCSI drivers and do not use them as modules if you use a SCSI hard disk for the system) and “Bondig driver support” (under “Network device support”).

Note: Be aware that it seems as if you could not use dynamic IP addresses on your nodes. To be able to get dynamic IP addresses you would have to enable «Socket Filtering» but this option is messing up with ipsec (see this posting[11] for more details).

11 http://lists.freeswan.org/pipermail/bugs/2002-November/000421.html

DL380 Note (Kernel options): Enable SMP support (also If you only have one CPU, cause the actual Intel Xenon CPUs support «Hyper Threading»), "Serverworks CSB5 chipsets support" and "Broadcom Tigon3 support" ("Network device support" > "Ethernet (1000 Mbit)") as module.
You could also download the newest official Broadcom drivers from their homepage[12].
Furthermore you need to include "Compaq Smart Array 5xxx Support" (in "Block devices") into your kernel (not as module).

After you finished the kernel configuration save your modifications and proceed with the following commands:

```
# make dep
# make clean
# make bzImage modules modules_install
# cp /usr/src/linux/arch/i386/boot/bzImage /boot
```

edit /etc/lilo.conf and add your new kernel:

Code listing: /etc/lilo.conf

```
[...]
default=Linux

image=/boot/bzImage
label=Linux
        read-only
#       restricted
#       alias=1

image=/vmlinuz
        label=LinuxOLD
        read-only
        optional
#       restricted
#       alias=2
[...]
```

```
# lilo
```

Now restart your nodes and see if your new kernel is booting without problems.

DL380 Note (BIOS settings after new kernel): You can now enable the «MPS Table» in den BIOS again.

4.3 Network

To have a redundant network connection we connected the nodes with two network interfaces to two different switches. To make this work we used the kernel feature «bonding» which offers different setups –

12 http://www.broadcom.com/drivers/downloaddrivers.php

we chose the option for more redundancy (and not more performance).

Note: If you want to use bonding for redundancy reasons you need at least three network interfaces (redundancy to external or internal network) - or four of them (redundancy to external and internal network)

4.3.1 Installation

In order to make bonding work the «mii-tool» utility needs to recognize your network interfaces. To see if it does, just execute it:

```
# mii-tool
eth0: negotiated 100baseTx-FD, link ok
eth1: negotiated 100baseTx-FD, link ok
eth2: negotiated 100baseTx-FD, link ok
```

If it does not recognize all or one of your network interfaces you have to update the package «net-tools» to a newer version, for example to the Debian unstable package «net-tools 1.60-9»

DL380 Note (Bonding): The Debian stable version of mii-tool does not recognize the integrated network interfaces. Though one has to update it. mii-tool is included in the package «net-tools» a working version is included in version 1.60-9 (Debian unstable) of this package. In order not upgrade all programs included in this package to unstable, you can download this package with wget and extract its content and just overwrite mii-tool with the new version:

```
# dpkg --extract net-tools_1.60-9_i386.deb net-tools_1.60-9_i386
# cp net-tools_1.60-9_i386/sbin/mii-tool /sbin
```

It's a bit nasty because this way you are skipping the package management of Debian but we didn't want to upgrade the other commands included in this package, though there is probably not speaking much against upgrading the whole package to unstable.

Finally, bonding need to be compiled as kernel module (see «Compiling New Kernel»)

4.3.2 Bonding

To activate bonding you need to add the following lines to /etc/modutils/aliases:

Code listing: /etc/modutils/aliases

```
alias bond0         bonding
alias bond1         bonding
options bond0         miimon=100 mode=1 multicast=1 primary=eth0
options bond1         -o bonding1 miimon=100 mode=1 multicast=1 primary=eth2
```

Check the kernel documentation[13] for more details about the possible parameters.

To update your /etc/modules.conf execute the following command:

13 /usr/src/linux/Documentation/networking/bonding.txt

```
# update-modules
```

In order to be able to attach network interfaces to a bonding device you need one more program:

```
# apt-get install ifenslave
```

Note: Because bonding is only checking the link state (with mii-tool), it does for example not recognize when the link somewhere else between master node and network gateway is broken. If this happens Heartbeat will fail over (because ipfail can detect such situations if you configured the ping nodes accordingly) if the route over the slave switch is ok. At this point it's important that eth0 of the slave node is connected to the slave switch (and not to the master switch like in case of the master node) otherwise it could lead to a deadlock. The other possibility to avoid this (if you want to connect on both nodes eth0 to the master switch) would be to change the bonding parameters on the slave mode like this (/etc/modutils/aliases):

```
options        bond0        miimon=100 mode=1 multicast=1 primary=eth1
```

4.3.3 Configuration

First stop your current network configuration:

```
# /etc/init.d/networking stop
```

Then you need to edit your network setup accordingly on the master node:

Code listing: /etc/network/interfaces

```
# The loopback interface
auto lo
iface lo inet loopback

# The external bonding interface
auto bond0
iface bond0 inet static
        address 160.85.178.168
        netmask 255.255.224.0
        network 160.85.160.0
        broadcast 160.85.191.255
        gateway 160.85.160.1
        up ifenslave -E bond0 eth0 eth1
        down ifenslave -D bond0 eth0 eth1

# The internal bondig interface
auto bond1
iface bond1 inet static
        address 10.0.0.11
        netmask 255.255.255.0
        network 10.0.0.0
        broadcast 10.0.0.255
        up ifenslave -E bond1 eth2
        down ifenslave -D bond1 eth2
```

And on the slave node:

Code listing: /etc/network/interfaces

```
# The loopback interface
auto lo
iface lo inet loopback

# The external bonding interface
auto bond0
iface bond0 inet static
        address 160.85.178.169
        netmask 255.255.224.0
        network 160.85.160.0
        broadcast 160.85.191.255
        gateway 160.85.160.1
        up ifenslave -E bond0 eth0 eth1
        down ifenslave -D bond0 eth0 eth1

# The internal bondig interface
auto bond1
iface bond1 inet static
        address 10.0.0.12
        netmask 255.255.255.0
        network 10.0.0.0
        broadcast 10.0.0.255
        up ifenslave -E bond1 eth2
        down ifenslave -D bond1 eth2
```

Now you can restart your network:

```
# /etc/init.d/networking start
```

DL380 Note (Network): Because you it was not possible to set up the network during primary installation of Debian (missing network interfaces drivers) you need to add the needed drivers as modules to /etc/modutils/aliases:

```
[...]
alias eth0      tg3
alias eth1      tg3
[...]
```

after updating this file, execute:

```
# update-modules
```

You probably also want to add some DNS server working for you, therefor edit /etc/reslov.conf:

```
search
nameserver xxx.xxx.xxx.xxx
nameserver xxx.xxx.xxx.xxx
```

Note: Because Heartbeat has difficulties recognizing bond and eth interfaces at the same time, we configured our internal network interface also as a bonding device, even there was only on internal interface per node.
(We propose to take two network interfaces for the internal network if you want real high availability)

As you now should have a working network setup (either with generic hardware or a Proliant DL380) you should check if apt uses the resources you want.
For example (using a Swiss mirror):

Code listing: /etc/apt/sources.list

```
deb ftp://sunsite.cnlab-switch.ch/mirror/debian/ stable main non-free contrib
deb-src ftp://sunsite.cnlab-switch.ch/mirror/debian/ stable main non-free contrib
deb ftp://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src ftp://non-us.debian.org/debian-non-US stable/non-US main contrib non-free

deb http://security.debian.org/ stable/updates main contrib non-free
```

4.4 Openswan

Get openswan-1.0.0 from the openswan homepage[14]

4.4.1 Installation

In order to compile openswan you need some additional packages:

```
# apt-get install libgmp3 libgmp3-dev libgmp3-doc
```

Extract openswan:

```
# tar xvzf openswan-1.0.0.tar.gz
```

You need to commend out a line (LIBSPLUTO += -lpam) in the following makefile at least in openswan-1.0.0 (if you don't to this, «pluto» is not going to be compiled):

Code listing: /usr/src/openswan-1.0.0/pluto/Makefile

```
[...]
LIBSPLUTO = -lgmp -lresolv -lpthread # -lefence

#LIBSPLUTO += -lpam

LDFLAGS =
[...]
```

14 http://www.openswan.org/code/

You are now able to compile, patch and recompile the kernel and install the needed tools with the following command at once:

```
# cd /usr/src/openswan-1.0.0
# make menugo && make minstall
```

Just exit and save the «Linux Kernel v2.4.24 Configuration» when it appears during the execution of the above command.

After compilation has finished you can copy the new kernel to the appropriate place and update lilo (you may want to save your actual working kernel before doing this and make a new entry in /etc/lilo.conf so you have working fallback in case your new kernel is not working):

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot
# lilo
```

Restart your nodes and see if your new kernel is booting without problems.

4.4.2 Certificates

We presume that you have a X.509 certificate as PKCS#12, named «ha_gateway_cert.p12»

If you want to make a new certificate, please take a look at one of the available Howtos[15] on the Internet or give «TinyCA»[16] a try.

Extract and move the key out of ha_gateway_cert.p12:

```
# openssl pkcs12 -nodes -nocerts -in ha_gateway_cert.p12 -out ha_gateway_key.pem
# chmod 400 ha_gateway_key.pem
# mv ha_gateway_key.pem /etc/ipsec.d/private
```

Modify ipsec.secrets:

Code listing: /etc/ipsec.secrets

```
: RSA ha_gateway_key.pem

```

There is nothing (uncommented) left in this file except this line. Note that there should be a vertical spacing after this line.

Extract and move the CA certificate out of ha_gateway_cert.p12:

```
# openssl pkcs12 -nokeys -cacerts -in ha_gateway_cert.p12 -out cacert.pem
# mv cacert.pem /etc/ipsec.d/cacerts
```

Extract and move the certificate out of ha_gateway_cert.p12:

15 http://www.evolvedatacom.nl/freeswan-3.html
16 http://tinyca.sm-zone.net/

```
# openssl pkcs12 -nokeys -clcerts -in ha_gateway_cert.p12 -out ha_gateway_cert.pem
# mkdir /etc/ipsec.d/certs
# mv ha_gateway_cert.pem /etc/ipsec.d/certs
```

Get the root CA crl file from your CA (if available) and move it to right place:

```
# wget http://yourca/root_ca.crl
# mv root_ca.crl /etc/ipsec.d/crls
```

Note: Install the same certificates on both nodes, otherwise you might get problems after a failover to the slave node if a client is using dead peer detection.

4.4.3 Nodes configuration

There is one configuration file for ipsec:

– ipsec.conf the main configuration is done in this file.

Edit your ipsec.conf:

Code listing: /etc/ipsec.conf

```
# basic configuration
config setup
        # %defaultroute is okay for most simple cases.
        interfaces="ipsec0=bond0:0"
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        klipsdebug=none
        plutodebug=none
        # Use auto= parameters in conn descriptions to control startup actions.
        plutoload=%search
        plutostart=%search
        # Don't wait for pluto to complete every plutostart before continuing
        plutowait=no
        # Close down old connection when new one using same ID shows up.
        uniqueids=yes

# Defaults for all connection descriptions
conn %default
        keyingtries=0
        disablearrivalcheck=no
        rightrsasigkey=%cert
        authby=rsasig
        leftcert=certs/ha_gateway_cert.pem

#Roadwarrior connection
conn roadwarrior
        left=160.85.178.167
        leftid=@gateway1.zhwin.ch
        leftsubnet=10.0.0.0/24
        right=%any
        dpddelay=5
        dpdtimeout=15
        auto=add
```

Some notes to the most important lines (please check openswan[17] readme file for more details):

interfaces="ipsec0=bond0:0"	uses the virtual bonding interface, which is created by heartbeat for ipsec
left=160.85.178.167	uses the “virtual” ip address, which is necessary also because of heartbeat
dpddelay=5 and dpdtimeout=15	defines the dead peer detection values

17 /usr/src/openswan-1.0.0/

In order to be able to use AES encryption you have to load the ipsec_aes module. To do is automaticaly, modify the following file:

Code listing: /usr/local/lib/ipsec/_realsetup

```
[...]
killklips() {
[...]
        then
                rmmod ipsec_aes
                rmmod ipsec
        fi
}
[...]
case "$1" in
  start|--start|_autostart)
[...]
        if test " $IPSECpluto" != " no"
        then
                modprobe ipsec_aes
                if ipsec _plutorun $re --debug "$IPSECplutodebug" \
[...]
```

For ipsec to work properly you need to enable source routing (hence disabling spoof protection) and ip routing:

Code listing: /etc/network/options

```
ip_forward=yes
spoofprotect=no
syncookies=no
```

Finally consider to disable ipsec to be started automatically, because it will get started by heartbeat.
You can use rcconf to deactivate or activate services from the init scripts.

4.4.4 Roadwarrior configuration

Here is an example for a Roadwarrior configuration. If you want to use AES encryption you have modify the file _realsetup for automated loading of the ipsec_aes module (see code listing: /usr/local/lib/ipsec/_realsetup
in the chapter «Nodes configuration» above.

Code listing: /etc/ipsec.conf

```
# basic configuration
config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search
        plutowait=no
        uniqueids=yes

# Defaults for all connection descriptions
conn %default
        keyingtries=0
        disablearrivalcheck=no
        rightrsasigkey=%cert
        authby=rsasig
        leftcert=certs/roadwarrior01_cert.pem

conn ha_gateway
        left=%defaultroute
        right=160.85.178.167
        rightsubnet=10.0.0.0/24
        rightid=@gateway1.zhwin.ch
        ike=aes256-sha,twofish256-sha
        esp=aes256-sha1,twofish256-sha1
        auto=start
        dpddelay=5
        dpdtimeout=15
        dpdaction=hold
```

Some notes to the most important lines (please check openswan[18] readme file for more details):

ike=aes256-sha,twofish256-sha
esp=aes256-sha1,twofish256-sha1
	setting up the tunnel with AES encryption

dpddelay=5 and dpdtimeout=15	defines the DPD values (need to be the same as on the nodes)
dpdaction=hold	if ipsec tunnel goes down, it will retry to setup a new tunnel with the same parameters automatically

18 /usr/src/openswan-1.0.0/

4.4.5 Logging

If you would like to have a log file containing ipsec warnings add the following line to syslog.conf:

Code listing: /etc/syslog.conf

```
[...]
*.=warn                          /var/log/warn
[...]
```

For not getting logs directly written to the standard output (mostly your screen), add the following parameter to klogd:

Code listing: /etc/init.d/klogd

```
[...]
KLOGD="-c 4"
[...]
```

```
# /etc/init.d/sysklogd restart
```

4.5 Iptables

If you use a Iptables script on your nodes you can dynamically update your Iptables rules for the ipsec tunnels with a script provided by openswan: /usr/local/lib/ipsec/_updown
Though there is a problem with this script with the interfaces (probably because of the bonding). We didn't analyze this problem any further.

But if you get this script working you can add it to the ipsec.conf:

Code listing: /etc/ipsec.conf

```
[...]
# Defaults for all connection descriptions
conn %default
        keyingtries=0
        disablearrivalcheck=no
        rightrsasigkey=%cert
        authby=rsasig
        leftcert=certs/ha_gateway_cert.pem
        leftupdown=/usr/local/lib/ipsec/_updown
[...]
```

4.6 Heartbeat

4.6.1 Installation

For ipfail working properly (we experienced some problems with heartbeat prior version 1.1.4) we install the following Debian packages, mostly downloaded from www.ultramonkey.org[19], in the given order:

- libpils0_1.1.4-1_i386.deb
- libpils-dev_1.1.4-1_i386.deb
- libstonith0_1.1.4-1_i386.deb
- libstonith-dev_1.1.4-1_i386.deb
- iproute (from stable tree, installed by just typing “apt-get install iproute” as root)
- heartbeat_1.1.4-1_i386.deb
- heartbeat-dev_1.1.4-1_i386.deb

To get the packages installed on your system simply type in the following command as root and don't forget to adjust the path to your deb-files!

```
# dpkg --install /path/to/package/<package>
```

Note: We propose to use the latest version of Heartbeat, at the end of our project 1.2.0 stable got released[20] which has some important bugfixes (for example the parameter auto_failback off is now working as it should).

4.6.2 Configuration

Basically there are three important files for heartbeat:

– ha.cf	In this file the main configuration is done.
– haresources	This file takes care of the IP addresses and the providing services.
– authkeys	Within this file the authentication method is set.

They should look exactly the same on both machines and be located in /etc/ha.d/ . If this isn't the case create them or make copies from the templates located in /usr/share/doc/heartbeat. Let's start with the first config file:

19 http://www.ultramonkey.org/download/heartbeat/1.1.4/debian_woody/
20 http://www.ultramonkey.org/download/heartbeat/1.2.0/debian_woody/

Code listing: /etc/ha.d/ha.cf

```
debugfile /var/log/ha-debug
logfile/var/log/ha-log
logfacility       local0

keepalive 2
deadtime 10
initdead 120

serial /dev/ttyS0
baud    19200

auto_failback on

node gw-master-dl380
node gw-slave-dl380

respawn root /usr/lib/heartbeat/ipfail
apiauth ipfail uid=root
ping 160.85.160.1 10.0.0.12
```

The first three lines affect the logging. The parameter *keepalive* sets the time in seconds between heartbeats. *deadtime* is the time after which a host is declared dead. Since the hardware or operating system needs some time to bring network up and start working right an initial deadtime called *initdead* takes care of that.
The next section is about the serial communication. *serial* defines the serialportname to use and *baud* sets the baud rate for the serial port. To check if your serial setup works you can do following on the machines:

```
Node1:~# cat < /dev/ttyS0
```

```
Node2:~# echo hello > /dev/ttyS0
```

On Node1 now a hello-string should appear at the promt!

Note: Instead or coexistent of using serial cable one can make the heartbeat using UDP. On the project side[21] the setup for a UDP-based heartbeat is explained very well.

The parameter auto_failback on causes to take over from the slave to the master node as soon as the master node is back. Probably you do not want to take over again automatically, because if there is somewhere a problem it could lead to a loop, switching to slave and master again and again.
But auto_failback off was not working properly with Heartbeat 1.1.4.

After node the hostname of the cluster member is stated. Since there are two of them this parameter appears two times in the config. ATTENTION: The hostname string has to be identically with the output of:

```
# uname -n
```

21 http://www.linux-ha.org/download/GettingStarted.html

The last paragraph informs ipfail to run as root and defines the ping nodes:

– 160.85.160.1	ping node befor the cluster
– 10.0.0.12	ping node behind the cluster

Next we tell heartbeat the services to start and the virtual IP addresses to use:

Code listing: /etc/ha.d/haresources

```
gw-master-dl380 160.85.178.167 10.0.0.10 ipsec
```

After the hostname of the master the virtual IP addresses to be created have to be stated. At the end the names of services heartbeat has to take care of are listed. Heartbeat looks for these service names in the directories /etc/init.d/ and */etc/ha.d/resource.d/* !

Finally we define the authentication method:

Code listing: /etc/ha.d/authkeys

```
auth 2
2 sha1 s3cr3t.p4sswd
```

Note: If you want to enable alerting, hence getting an email in case of a failover you have to edit the following line in /etc/ha.d/haresources accordingly:

```
[...]
gw-master-dl380 160.85.178.167 10.0.0.10 ipsec Mailto::email@yourdomain.com::ha-failover
[...]
```

This is only working if you configured your nodes to be able of sending emails.

Note: If you want to observe the ipsec process itself you have to take a look at the open source project «mon»[22]. Their is a Howto[23] which describes the interaction of Heartbeat and mon.

22 http://www.kernel.org/software/mon/
23 http://www.geocities.com/latompa/ha/apache_heartbeat.html

4.7 Rsync

For the config files being identical on both nodes we add a cronjob which copies the necessary files let's say every ten minutes from the master to the slave (since testing of changes in the configuration on the slave before implementing them on the master doesn't work for several reasons we chose this way).

First we install on both nodes *rsync* if we haven't done this yet:

```
# apt-get install rsync
```

The following commands have to be done on the master node only!

First we have to generate a RSA authentication key. When asked for the place where to save the key define */root/.ssh/id_rsa* by simply pressing enter. Also make sure to specify a blank passphrase (we don't want one!). So basically you finish up by pressing three times enter after having entered the command.

```
Master:~# ssh-keygen -t rsa
```

Now we have to copy the key to the slave (substitute <Slave> with the DNS name or IP address of your slave node:

```
Master:~# ssh-copy-id -i /root/.ssh/id_rsa.pub root@<Slave>
```

Then we specify in a shell script the configuration files to copy. Again substitute <Slave> with the real IP address or hostname of your slave node and save it to */root/config-synchro.sh* .

Code listing: /root/config-synchro.sh

```
#!/bin/bash
rsync -e ssh -a /etc/ipsec.conf root@<Slave>:/etc
rsync -e ssh -a /etc/ipsec.secrets root@<Slave>:/etc
rsync -e ssh -a /etc/ha.d/* root@<Slave>:/etc/ha.d
```

To make the script executable and make sure nobody can do anything nasty with our shell script we set the correct permissions:

```
Master:~# chmod 700 /root/config-synchro.sh
```

To finish this chapter we add a line to the crontab which tells the cron daemon to execute our script every ten minutes:

```
Master:~# crontab -e
```

Code listing: crontab entry

```
*/10 * * * * /root/config-synchro.sh 2&> /dev/null
```

It would be a good idea to test your setup by changing a value in a config file on the master and check on the slave ten minutes later if the changes were made!

4.8 Trix und Gägs us Gargamel's Wundertütä

4.8.1 Disabling unnecessary services

We propose to disable all services on the nodes which aren't needed. You can use the tool rcconf for a simple configuration of the init scripts.
If not already installed:

```
# apt-get install rcconf
```

Now you can execute rcconf end disable the not needed services. For example you probably don't need the following services: lpd, inetd, nfs-common and nfs-kernel-server
Also disable the service ipsec because it will be started by heartbeat.

```
# rcconf
```

4.8.2 Ntpdate

It's important to have on both nodes the same time. At the latest if you are trying to compare logs from both nodes. Therefor we installed ntpdate (though you could also use the ntpd servic if a even more accurate time is needed):

```
# apt-get install ntpdate
```

When it asks for a timeserver you can take for example this one: time.ethz.ch

Now you can add a crontab entry which regularly updates the time on your nodes:

```
# crontab -e
```

Code listing: crontab entry

```
11 11 * * * ntpdate -u time.ethz.ch 2&> /dev/null
```

Note: Please adjust the server hostname to a ntp time server next to you[24]!

24 http://www.eecis.udel.edu/~mills/ntp/servers.html

4.8.3 Make sshd more secure

To add more security you could disable root login via the sshd deamon. This way you first have to login as a normal user and then change with su to root:

Code listing: /etc/ssh/sshd_config

```
[...]
# Authentication:
LoginGraceTime 600
PermitRootLogin no
StrictModes yes
[...]
```

If you us rsync to exchange your configuration files to the other node automated with a crontab entry you have to setup another sshd daemon (or use inetd) on a different port. On this sshd daemon root login has to be enabled but you can restrict access only to the ip of the other node.

Now you can use this sshd daemon for rsync like this:

```
# rsync -a -e 'ssh -p500' ...
```

If the second sshd daemon is running on port 500.

4.8.4 Update your debian

Don't forget to update from time to time your nodes:

```
# apt-get update
# apt-get upgrade
```

Note: Pay attention to programs like openswan, heartbeat or mii-tool because they need to by updated separately if there are for example some security holes!

5 Summary

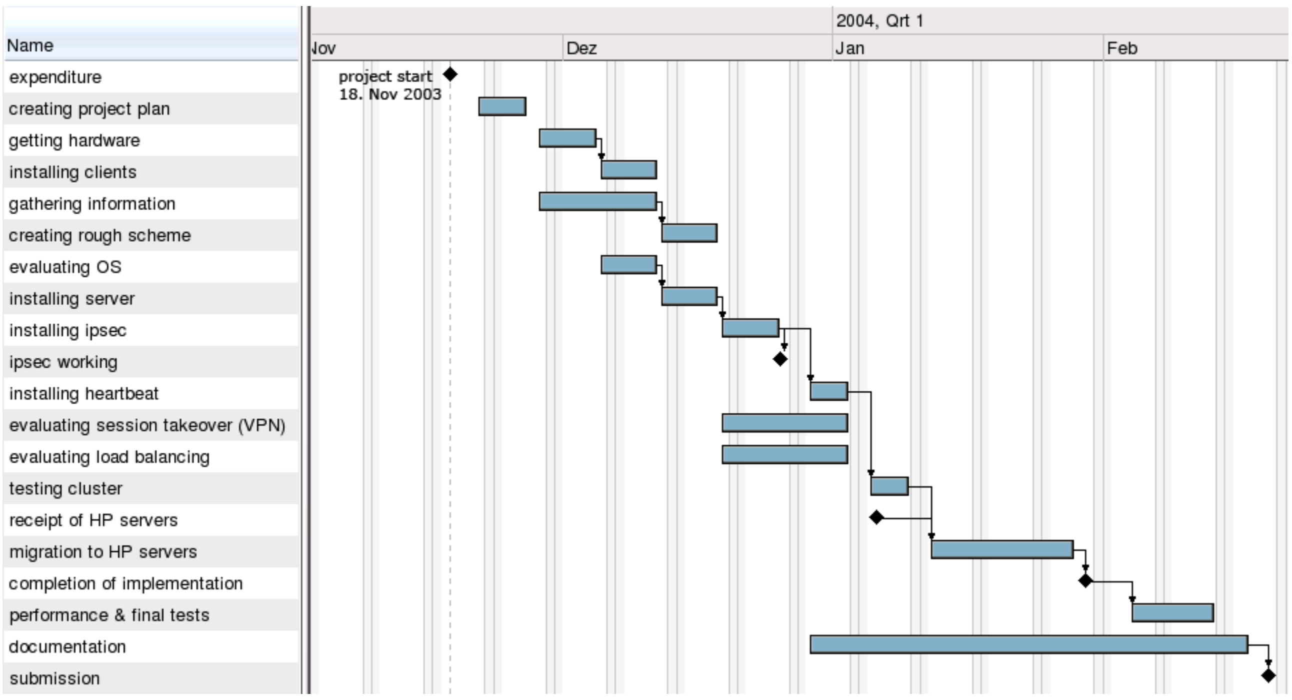
In this project we had the opportunity to work ourself into an interesting topic. We were provided with a good infrastructure (thanks to HP) and were pretty free in our timings.
Though the main focus of our project was high availability we first had to spend some time understanding how exactly IPSEC and IKE work since we had no experience with this topic. Here it would have been of some help if we had been able to attend the SNK (Secure Network Communications) classes.
While researching for Heartbeat we discovered that there are many possibilities to improve a HA setup or to add additional features. For example of ipfail, bonding, mon and more.
We are very contented with the result of our project work as we reached all our goals.
We think this project could be developed further regarding load balancing issues and administration tools. Thereby a equal alternative to expensive VPN high available hardware of commercial manufacturers could move in professionals environments.

Finally we would like to thank our lecturer Prof. Dr. Andreas Steffen for all the information and tips we received and for his friendly supervision and Hewlett-Packard Switzerland for its sponsorship.

6 Appendix

6.1 Time schedule

Generally speaking we were able to comply with the time schedule we created at the beginning. The only exception was the implementing of IPSEC which took us longer. However we had no problems to make up for that.



6.2 Configuration files

All the configuration files we did changes to can be found here[25].

6.3 Master node

/etc/apt/sources.list

/etc/ha.d/authkeys

/etc/ha.d/haresources

/etc/ha.d/ha.cf

/etc/init.d/klogd

/etc/modutils/aliases

/etc/network/interfaces

/etc/network/options

/etc/ipsec.secrets

/etc/ipsec.conf

/etc/lilo.conf

/etc/syslog.conf

25 http://security.zhwin.ch/ha_security_gateway_config.tar.gz

/usr/local/lib/ipsec/_realsetup
/usr/src/linux-2.4.24/.config
/usr/src/openswan-1.0.0/pluto/Makefile
/root/config-synchro.sh

6.4 Slave node

/etc/apt/sources.list
/etc/ha.d/authkeys
/etc/ha.d/haresources
/etc/ha.d/ha.cf
/etc/init.d/klogd
/etc/modutils/aliases
/etc/network/interfaces
/etc/network/options
/etc/ipsec.secrets
/etc/ipsec.conf
/etc/lilo.conf
/etc/syslog.conf
/usr/local/lib/ipsec/_realsetup
/usr/src/linux-2.4.24/.config
/usr/src/openswan-1.0.0/pluto/Makefile

6.5 Roadwarrior

/etc/ipsec.conf

6.6 Illustration list

6.7 Table list

6.8 Sources

6.8.1 Links

http://www.freeswan.ca/

http://www.openswan.org/

http://www.linux-ha.org/

http://www.strongsec.com/freeswan/

http://www.ultramonkey.org/2.0.1/ip_address_takeover.html

http://pheared.net/devel/c/ipfail/

6.9 Glossary

APT	Advanced Packaging Tool
ARP	Address Resolution Protocol
AWK	pattern scanning and processing language
Bonding	Ethernet channel bonding
Cluster	virtual server built on a cluster of real servers
DPD	Dead Peer Detection
DRBD	Distributed Replicated Block Device
Gratuitous ARP	ARP request for own IP address
HA	High Availability
HP	Hewlett Packard
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPSEC	Internet Protocol SECurity
ISO	International Organization for Standardization
ISP	Internet Service Provider
Master node	active node in a cluster
Master switch	primarily used switch by master node
Perl	high-level programming language with an eclectic heritage
Ping node	target for ipfail to test the connectivity to this point
RAID-1	Redundant Array of Inexpensive Disks (level 1)
RFC	Request For Comment
Roadwarrior	VPN user connecting from home
RSA	algorithm invented by Rivest, Shamir and Adleman in 1977 which is often used for cryptographic schemes
Rsync	open source utility that provides fast incremental file transfer
SA	Security Association
SHA	Secure Hash Algorithm

Slave node	passive node in a cluster, mostly in hot standby
Slave switch	primarily used switch by the slave node
SSL	Secure Sockets Layer
S/WAN	Secure Wide Area Network
Trix und Gägs us Gargamel's Wundertütä	tweaking tips
VPN	Virtual Privat Network
wget	free software package for retrieving files using HTTP, HTTPS and FTP