

HOWTO Configure a High Available Firewall Cluster with `fwbuilder`

Dr. Michael Schwartzkopff*

February 24, 2009

MultiNET Services GmbH, 85630 Grasbrunn, Munich, Germany

Feb, 24th 2009: rev. 0.2.1

Abstract

In this document I want to show HOWTO configure a high available cluster that will meet all needs of a modern firewall infrastructures. The components of the cluster are the Linux OS, `iptables`, `conntrackd`, `heartbeat` and `fwbuilder` to manage the policies.

License: GNU FDL, see <http://www.gnu.org/copyleft/fdl.html>. Invariant section: Author.

Every network admin sooner or later is confronted with the problem to configure a high available firewall. The firewall is a Single Point of Failure in the connection from the internal network with the internet. A stop of this service would disable all users from watching videos at youtube and all managers from receiving mails at their blackberries. A medium catastrophe for the network admin.

I want to describe the setup of a high available firewall cluster that is managed by the graphical user interface `fwbuilder`. One node of the cluster is the active node. In case of a failure all traffic is redirected via the second node. The connection table of the active system is synced to the passive node so in case of a failover all connections are preserved.

Basically this HOWTO was written for all operating systems that `fwbuilder` would work on. There are two distinct Linux dependencies: The Linux-HA Cluster and the `iptables` firewall with connection state sync. Please look for something adequate in your OS. But the `fwbuilder` part should hold for all systems.

I am still in the process of writing this HOWTO. If you have any suggestions or miss something, please feel free to mail me.

1 Hardware

The hardware for firewalls is not so challenging any more. Modern servers can cope with GBit networks. CPU power and harddisk space are no problem any more nowadays. Perhaps you need some large partitions for logging all the traffic. Please keep this in mind when partitioning your disc. Solid State Discs are also an option. They are not as expensive as they were years ago and they are much more reliable than ordinary drives. They are especially interesting if you do not need too much space on your disc and log to an other system.

I also would setup a new system only with RAID1 discs. Harddiscs are cheap and not to compare to the trouble a crashed disc drive causes when you have to setup the complete operating system from the scratch again.

*misch@multinet.de

1.1 Network

The most crucial point for a firewall is the connection to the networks. So please be sure your newly purchased system does have enough network interfaces. The firewall needs at least these interfaces:

- 1 x external
- 1 x internal
- 2 x heartbeat cluster communication

When setting up a cluster I always like to have redundant cluster communication. There is nothing worse in the life of a cluster than a split brain situation when one node cannot see the other node.

If you want to have a DMZ interface, add this one to the list above.

For a really good high availability it would also be wise to attach the firewall to the networks with redundant bond interfaces. That makes 8 interfaces for a fully redundant firewall with DMZ. Please keep in mind that you also would need to install a redundant switch infrastructure. Without that infrastructure redundant interfaces of the firewall do not make sense.

1.2 Bond Interfaces

Redundant interfaces of firewalls can be connected to bond interfaces. It is a very good practice to do so, because the management of one bonded interface is much easier than two separate ones.

The example below is for a debian setup of */etc/network/interfaces*:

```
auto bond3
iface bond3 inet static
    address 192.168.100.1
    netmask 255.255.255.0
    slaves eth3 eth4
    bond-mode 4
    bond-miimon 100
```

See also the documentation for the debian project¹. Please see the documentation of your OS to find out how to bond interfaces.

1.3 VLANs

8 interfaces for a plain high available firewall node is quite a lot. Adding more DMZs or different internal networks require always additional interfaces. If you do not have it, virtualize it! You can use VLANs on some real interfaces.

Some people see security problems when frames of different security zones are transported on the same cable. Of course, there are if the configuration has a bug. So as a security expert you should anyway know what you are doing. So I'd say VLANs are OK. But you have to decide it your own.

Please see the documentation of your OS how to configure VLAN interfaces.

¹<http://etbe.coker.com.au/2007/08/13/ethernet-bonding-on-debian-etch/>

2 Operating System

The HOWTO concentrates somehow on the Linux OS, but the `fwbuilder` works for a large variety of OSs. If you use something other than Linux, just read the other doc about firewalling, clustering and connection table sync. I am not an expert on *BSD (with `carp`) or others that I would dare to write something here. Please mail me if you have extentions of this manual for other OSs.

3 Cluster

Linux-HA provides a stable cluster environment. Version 2 has built in checks for the functionality of IP addresses as managed resources as well as the function of the attached network. Since this should mainly be a `fwbuilder` manual I only will include some basic configurations and links to other config sites here.

Every node of the cluster has dedicated IP addresses on the interfaces. Perhaps you only configure IP addresses on the interfaces you afterwards want to manage the nodes. But I prefer to have dedicated addresses on all interfaces, so you can SSH to the machines directly. The cluster software manages virtual IP addresses the provide the default addresses for routing afterwards. Constraints of the configuration prevent these clusters to run on failed nodes. Table 1 shows the conventions for the IP addresses used later on in this document.

	node1	node2	cluster
ext / eth0	10.0.0.2	10.0.0.3	10.0.0.1
int / eth1	192.168.0.2	192.168.0.3	192.168.0.1
dmz / eth2	192.168.1.2	192.168.1.3	192.168.1.1
heart beat / bond3	192.168.100.2	192.168.100.3	-

Table 1: Definition of the IP addresses of the interfaces of the firewall cluster.

The following configuration samples are according to the `pacemaker` software. Please keep that in mind if you transfer the samples to `heartbeat` v2 software. The `pacemaker` software is quite new so it might not be included in your distribution. `pacemaker` is just the same code as it was the Cluster Resource Manager (CRM) in `heartbeat` version 2. The project provides not only the source code for `heartbeat` (without CRM), `OpenAIS` and `pacemaker`, but offers also binary packages for several distributions. Please see their project website² for the download and install instructions.

Simon Horman compiled the packages for debian systems. Please find the latest on his website³. These packages work for debian lenny. Just downlaod and install it with `dpkg`.

Basically `pacemaker` uses `heartbeat` or `OpenAIS` as a communication platform. So you need to install `pacemaker` and `heartbeat` or `pacemaker` and `OpenAIS`. The `pacemaker` project also provides a GUI which can be installed from a different package. I will use an installation of `heartbeat`, `pacemaker` and `pacemaker-gui` for the further setup.

3.1 Basic configuration

For details on the basic config files of `heartbeat`, see the project documentation: <http://www.linux-ha.org/GettingStartedV2>.

²<http://www.clusterlabs.org>

³<http://packages.vergenet.net/experimental/>

3.1.1 Shared secret in */etc/ha.d/authkeys*

Both nodes of the cluster authenticate the communication. So they need a shared secret which is located in the file */etc/ha.d/authkeys*. A sample file is included in the documentation of the project:

```
auth 2
2 sha1 MySecret
```

Please make that file only readable by root:

```
chmod go-rw /etc/ha.d/authkeys
```

Otherwise heartbeat would not start and complain about the problematic rights. This file has to be the same on both nodes.

3.1.2 Configuration in *ha.cf*

The basic options for the `heartbeat` communication is defined in the */etc/ha.d/ha.cf* file. The most basic configuration in our case is:

```
bcast bond3
autojoin any
apiauth mgmt uid=root
respawn root /usr/lib/heartbeat/mgmt
crm on
```

Do not forget the `crm on` to have the Cluster Resource Manager (CRM) taking care of the resources. If you use broadcast communication over the same interface this file can be the same on both nodes. After you connected the two cluster nodes you can start heartbeat on both nodes with

```
/etc/init.d/heartbeat start
```

The two clusters nodes have to talk to each other to exchange status information. By default this communication uses the port `udp/694`. DO NOT block this port in the `heartbeat` interfaces. In every policy installed on the nodes you have to have something like:

```
iptables -I INPUT -i bond3 -p udp --dport 694 -j ACCEPT
iptables -I OUTPUT -o bond3 -p udp --dport 694 -j ACCEPT
```

Entering `crm_mon -1` shows the cluster status. It should show you that both nodes found each other and are online after a short while. Now we can start entering resources into the cluster configuration. To work with the GUI the user `hacluster` should be given a reasonable password with the `passwd` command:

```
passwd hacluster
```

3.2 Resources

All virtual IP addresses of the cluster have to be configured in a resource group of the cluster configuration. The group will start all resources on the same node. This one will be the “active” node afterwards. I will show you step by step how to configure the resources with the help of the GUI. Please note that the steps are according to the pacemaker GUI. If you use a heartbeat version 2 (i.e. version 2.1.4) the GUI look a little different, but the basic steps are the same.

Open the GUI entering

```
/usr/lib/heartbeat-gui/haclient.py
```

Connect to the cluster management by clicking **Connect**→**login**. Enter the password for the user `hacluster` you just defined. You should see an empty status window only showing the two cluster nodes. Go to the resources and **Add a new one**. Select “Group” when asked.

The next window asks you about **Wizard** or **Dialogs**. Perhaps you have a old version of the GUI so you will not see this window. Then you will have to enter all resources, attributes and operations in the **Dialog** mode. The wizard just makes life easier. So I will go on using the wizard.

When asked about the group ID enter `groupFirewall`. The group should initially be in the state `stopped`. You will automatically be forwarded to enter the first resource, a **Primitive**. Give it the ID `resIPext`. Enter the rest of the resource information according to figure 1

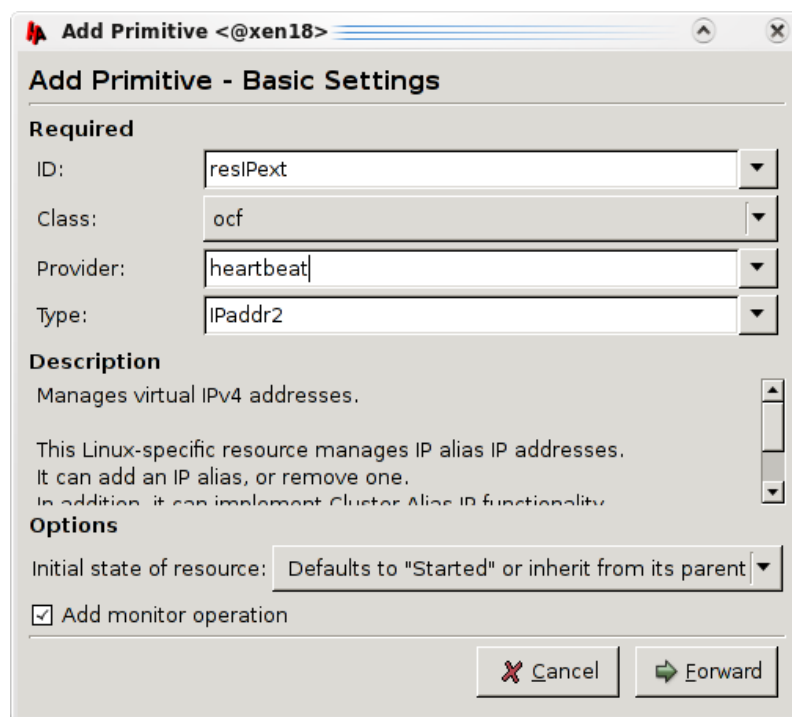


Figure 1: Basic information of the external IP address resource `resIPext`.

Entering **Forward** will lead you to the definition of the attributes of that resource. First of all, you have to give it a `instance_attribute` IP address. It is a good habit to enter `nic` and `netmask` also. Please see figure 2 for comparison. The wizard should have added the monitor operation automatically. Perhaps you want to change the behavior of the operation, if the monitoring of the resource failed. Just select the operation tab, select the monitor op and edit it. Change the `On fail` field to `restart` and press **OK**. After pressing **Apply** for the first resource you can add more cluster IP address for the interfaces.

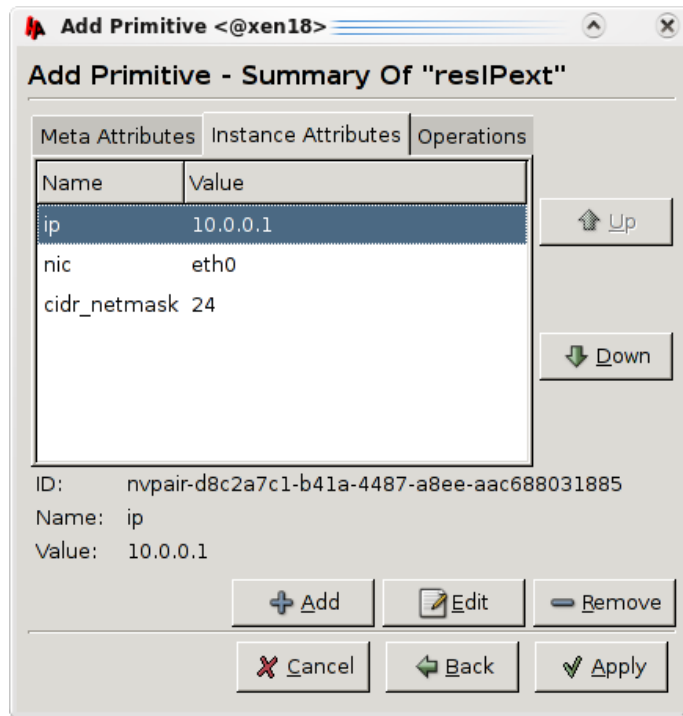


Figure 2: Entering the instance_attributes of the external IP address resource.

When you select the `groupFirewall` in the resources and let the GUI display not the List Mode but the XML mode you will see the internal representation of the group and all the primitive resources in the group. This definition should more or less coincide with the definition in the appendix (see 1).

All resources have a monitor operation and a `migration-threshold` (meta_attributes) defined. So the cluster will monitor all resources. Failures will restart the resource, but after three failures of any resource all resources will migrate to the other node, so the passive node becomes active.

You should add a `pingd` to check the attached networks. Based on the check `pacemaker` can migrate the resources in case the net is not reachable from a node any more. For details see: <http://www.linux-ha.org/pingd>

Please note that there is no firewall script included in that definition up to now. We will add it as we setup our cluster further.

Please check if the failover works as you expected. Pull the plug of one node, kill programs (i.e. the heartbeat process), and torture your cluster with real life failures.

3.3 Constraints

If you prefer to be one node preferably active you can add this by configuration of a location constraint of the group. But in a cluster it should not matter which node is the active.

Constraints are also needed for the cluster reaction the the network availability tests of pings. See the explanation there.

3.4 The Firewall Resource

Of course the clusters also needs a firewall resource that controls the `ip_forward` parameter of the kernel. For now the resource does nothing else. For the cluster operation it is essential that the resource agent (some

people call it old fashioned `init-script`) understands start, stop and status and returns the correct return codes. Please see the appendix (section 9.1) for a sample script. It is far from being perfect. If you have improvements, I would like to add your patches.

The start section would set `ip_forward` to "1". For the sake of beauty we could also call the firewall script, when necessary. Please do NOT start this firewall script in the init process during system start up, but use some basic default rules to protect the node during start up and being passive.

The stop part of the script would set `ip_forward` to "0" and install some default policy. Take care that this default policy does not block the SSH access to the node.

The status part of the script would just look whether `ip_forward` is set or not. Please keep in mind that the `fwbuilder` script is not longer controls the forwarding of the kernel.

As as basis for the firewall script I took the debian script from the `fwbuilder-common` package. It works quite well, but you have to add the status part.

Now you can add the firewall resource to the Firewall group of the cluster just behind the last IP address. This time you have to add a LSB resource because the `init-script` is LSB compatible and not OCF. See figure 3 for the definition.

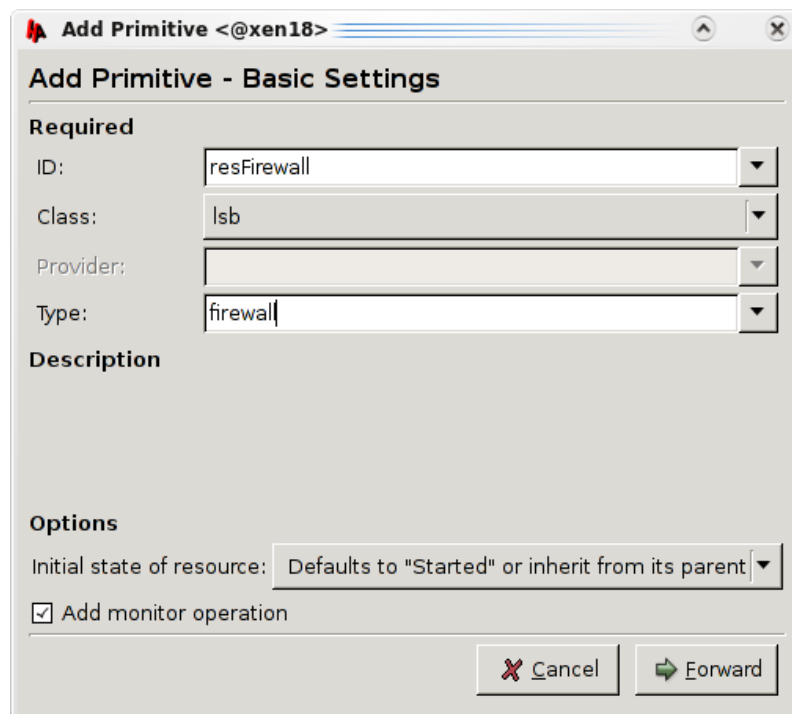


Figure 3: Definition of the firewall resource within the cluster definition

4 Synchronization of the Connection Table

OK, the firewall cluster works and in case of a problem the passive node takes over. One problem remains: The connection table of the active system is not synced to the passive one. So in case of a failover all connections would be lost and the user would have to download the file again. To some limited extend netfilter can catch up with lost tcp connections but no chance with udp or icmp packets. So we have to sync the connection states of the active system to the passive one.

`netfilter` provides the `conntrackd` to deal with this task which is included in the modern distributions or will be shortly. In the simple mode this daemon manages two caches. One cache mirrors the kernel

connection table. This cache is exported over the network via multicast. The other node receives these communication and stores the connections into a second cache, the external cache. In case of a failover the external cache is synced to the internal kernel table. There also exists an advanced mode, where the sync happens immediately. In this mode, split routing setups are possible.

4.1 Versions

I check debian lenny out of the box and it did not work. It seems that the NAT helper module of the default kernel has a problem. So I installed a new kernel. Just add

```
deb http://kernel-archive.buildserver.net/debian-kernel trunk main
```

into `/etc/apt/sources`, run `apt-get update` and you can install the latest kernel with `apt-get install` It works for me with kernel 2.6.28-1. I don't know about other distributions and kernels. If you have experience, let me know and I will add here.

When I startet up upgrade packages on my test firewal cluster I also wanted to have the latest `contrack-tools`. So I got the latest sources for `contrack-tools`, `libnfnetlink`, and `libnetfilter_contrack`. Please the INSTALL file of `contrack-tools` where to download the tarballs. All packages compile and install smoothly with `./configure`, `make` and `make install`. I finally ended up a system with

```
libnfnetlink-0.0.40, libnetfilter_contrack-0.0.99, and contrack-tools-0.9.11
```

4.2 Configuration

In our example we use the the `heartbeat` interface (`bond3`) for the sync traffic. So the simplified config for the `contrackd` on node 1 would look like:

```
Sync {
  Mode FTFW {
  }
  Multicast {
    IPv4_address 225.0.0.50
    Group 3780
    IPv4_interface 192.168.100.2
    Interface bond3
    ...
  }
}
General {
  ...
  Filter From Userspace {
    Protocol Accept {
      TCP
    }
    Address Ignore {
      IPv4_address 127.0.0.1 # loopback
      ... # put in here all virtual and dedicated addresses of the firewall
    }
  }
}
```

Please see the comments in the config file `/etc/contrackd/contrackd.conf` for more information about the parameters and options. The project also has a good manual online⁴. Beware that debian lenny stores the config file under `/etc/contrackd.conf`. If you do not move this file to the default location you would have to add `-C /etc/contrackd.conf` to all commands below to indicate the correct location of the config file. Please also have a look to the init script and `/etc/default/contrackd` if you use debian.

As you can see the `contrackd` on the both nodes communicate on port 3780 of the multicast address. Please keep in mind not to block this traffic. We will create a `fwbuilder` rule afterwards.

```
iptables -I INPUT -i bond3 -p udp --dport 3780 -j ACCEPT
iptables -I INPUT -o bond3 -p udp --dport 3780 -j ACCEPT
```

Of course you can restrict these rules even further with a `-d 225.0.0.50`. I did not put this into the rules since I consider the heartbeat link as a safe crossover cable under my control.

We also have to extend our firewall start script with the commands that manage the connection table caches. In the start function we have to add:

```
/usr/sbin/contrackd -c # commit external cache to the kernel
/usr/sbin/contrackd -f # flush internal and external cache
/usr/sbin/contrackd -R # resync with the kernel table
/usr/sbin/contrackd -B # send a bulk update on the line
```

While shutting down the resource with the stop operation you would have to call

```
/usr/sbin/contrackd -t # shorten kernel timers to remove zombies
/usr/sbin/contrackd -n # request a resync from the others
```

Please see the documentation of `contrackd` for more information about the options. The documentation also has a much better script then these lines here. Have a look into the `doc/sync` directory.

Of course the `contrackd` had to be started on your system. This is done by the normal `init` process. So at this stage the `contrackd` is not a resource of the cluster, but individually started on every node at system start. When I find time I will check if the `contrackd` could be included as a cluster resource. This could be done i.e. by cloning one primitive resource.

Use latest for stability, otherwise you may experience problems with old releases. Integrating `contrackd` into the cluster as a resource is preferable as it is a good practise to monitor the daemon behaviour. But the `init` script is in now way usable for the heartbeat cluster (no status), so one would have to write a improved version. During my experiments I had `contrackd` also sometimes crashed leaving the PID file locked and thus preventing it from startig again. So a good `init` script would have to check the existance of `contrackd` in the process table and, in case of failure, delete the PID lock. Until that work is done, you have to check from time to time, if the `contrackd` is still running on both systems. Anyway, `contrackd` is not really essential for the function of the firewall. It is nice not to loose alle session during a failover.

5 Management with fwbuilder

To manage the firewall cluster with `fwbuilder` you'll need version 3 of the software. The project page⁵ provides source code of the software as well as binary packages for the most distributions. After download you would install it to your admin PC, not necessarily to one of the firewall nodes. `fwbuilder` has a function to transfer and install the new build policy. Create two new firewalls according to your two nodes. Define

⁴<http://contrack-tools.netfilter.org/manual.html>

⁵<http://www.fwbuilder.org>

all interfaces (eth0, ...) and both dedicated and cluster IP addresses on both nodes. According to table 1 eth0 of node 1 would get a IP address of 10.0.0.2 (marked dedicated) and the cluster IP address 10.0.0.1 (marked cluster) in figure 4. It shows the definition of the firewall nodes. Both nodes have defined their dedicated IP addresses and the cluster addresses in their interfaces.

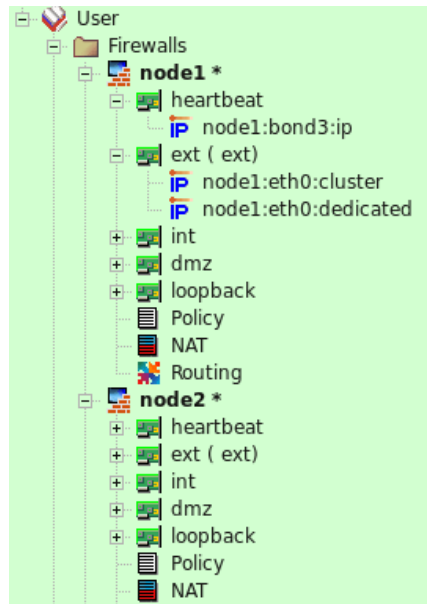


Figure 4: Entering two firewall nodes into fwbuilder. Both nodes have their dedicated and the cluster IP addresses defined.

First of all we need the local policies for the two nodes. Put all rules into these policies that touch the nodes itself. I took the sample from the firewall templates. So all rules until the stealth rule go in here.

Now add one rule allowing the heartbeat (udp/694) and conntrackd (udp/3780) communication on the heartbeat interface. I added two upd services to define the rule. Please see this rule marked yellow in figure 5.

Please also add a rule diverting all further traffic to a common cluster policy as the last rule in this policy. It does not matter under which node you define that cluster policy. The only restriction is that no dedicated node appears in that policy. Please be sure that the node policy is the top one.

	Source	Destination	Service	Interface	Direction	Action	Time	Options	Comment
0	node1 dmz heartbeat internal	Any	Any	ext			Any		anti spoofing rule
1	Any	Any	Any	loopback			Any		
2	internal	node1	ssh	All			Any		SSH Access to firewall is permitted
3	Any	Any	heartbeat contrackd	heartbeat			Any		heartbeat and contrackd between nodes
4	node1	internal server	DNS	All			Any		Firewall uses one of the machines
5	Any	node1	Any	All			Any		All other attempts to connect to
6	Any	Any	Any	All			Any		Chain to the cluster policy

Figure 5: Local ruleset of node1 on the firewall cluster. The yellow marked rule allows heartbeat and contrackd communication and the last rule diverts to the cluster policy.

The rest of the Policy is defined the the cluster ruleset. Please take care that this ruleset does not contain any reference to a dedicated node.

5.1 Firewall settings

Since the IP addresses are managed by the cluster software they do not have to be set in the firewall script. So disable the setting of the addresses and the address verification in the properties of both firewall nodes. You also have to adjust some other settings in the properties dialog: node1 -> Edit -> Firewall Settings > Compiler:

- Compiler options
 - Disable “Firewall is part of any”: You can leave that turned on, if you like. This is just a preference of me.
- Script options
 - Disable “Verify interfaces”
 - Disable “Configure interfaces of the firewall”
 - Disable “Add virtual addresses for NAT”
- Host OS Settings -> Options -> IPv4 Packet forwarding -> No change.

Since we have to disable the NAT option, you also will have to add all NAT addresses to the external interface of the cluster setup. This has to be done via the cluster software, because these addresses also have to fail over.

6 Tests

Install the policy and test the setup.

A nice test is logging in to a machine on the other side of the firewall and execute

```
external# while [ 1 ] ; do date ; sleep 1 ; done
```

This displays the date and time every second. Now look which firewall is active and have a look to the internal chach of the connection table, assume fw1 is active:

```
fw1# conntrackd -i
```

If you have a lot of connections you can limit the output appending `| grep "dport=22"` to the command. You should see the SSH connection. Now look to the external cache of the other machine:

```
fw2# conntrackd -e
```

The same connection should be visible here. Now you can cause the failover the switching one machine to the standby mode. It does not matter on which node you enter the command:

```
# crm_standby -v1 -N fw1
```

with

```
# crm_mon
```

you can have a look to the state of the cluster and see the active node switching the standby, shutting down all resources and the cluster starting all resources on the other node. The windows of the remote connection should stop display the time for some seconds but then catch up and continue to show the time. What do `conntrackd -i` and `conntrackd -e` tell you on the nodes?

For a second test get a big but useless file to download, like <http://www.microsoft.com/windows/internet-explorer/beta/worldwide-sites.aspx>. During the switch the active firewall to standby, take it online again and switch the other to standby. The download should survive all failovers.

Congratulations, you are finished with the basic setup of your HA firewall cluster. Please mail me any suggestions, improvements and error corrections to misch@multinet.de.

As I find more time, I intend to write the first sections in more detail.

Have fun!

7 VPN

Now you can add the VPN stuff (OpenVPN and OpenSWAN). Both services provide means of checking the tunnels and reestablishing in case of failover.

7.1 OpenVPN

Look for the keyword `keepalive`.

7.2 OpenSWAN

Look for the keyword "dead peer detection". Config option in `ipsec.conf`:

```
dpdaction=restart
```

8 IPv6

Comming!

9 Appendix

9.1 firewall Script

```
#!/bin/sh #
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
NAME=fwbuilder DESC="Firewall Builder"
DEFAULT=/etc/default/fwbuilder
IPTABLES=/sbin/iptables
test -f $DEFAULT || exit 0
grep -s -q 'START_FWUILDER=yes' $DEFAULT || exit 0
# SCRIPT_DIR=$(grep -s "^[[:space:]]*FWBSCRIPT_DIR" $DEFAULT | cut -d "=" -f 2)
# SCRIPT="$SCRIPT_DIR/$(hostname -s).fw"
SCRIPT=/etc/firewall.fw
NAME=fwbuilder
stopfw() {
    #Set accept for default tables
    # This is definitely NO good policy. DO NOT USE in production environment.
    $IPTABLES -P OUTPUT ACCEPT
    $IPTABLES -P INPUT ACCEPT
    $IPTABLES -P FORWARD ACCEPT
    #Flush tables
    $IPTABLES -F
    $IPTABLES -F -t nat
    $IPTABLES -F -t mangle
    $IPTABLES -X
    $IPTABLES -X -t nat
    $IPTABLES -X -t mangle
}
test -x $SCRIPT || exit 0 test -x $IPTABLES || exit 0
set -e
case "$1" in
    start)
        echo -n "Starting $DESC: "
        /usr/sbin/conntrackd -c
        /usr/sbin/conntrackd -f
        /usr/sbin/conntrackd -R
        /usr/sbin/conntrackd -B
        $SCRIPT 2>/dev/null
        echo "1" > /proc/sys/net/ipv4/ip_forward
        echo "$NAME."
        ;;
    stop)
        echo -n "Stopping $DESC: "
        /usr/sbin/conntrackd -t
        /usr/sbin/conntrackd -n
        echo "0" > /proc/sys/net/ipv4/ip_forward
        # stopfw
        echo "$NAME."
        ;;
    status)
        if [ 'cat /proc/sys/net/ipv4/ip_forward' = "1" ]
        then
            echo "$NAME forwarding"
```

```
        exit 0
    else
        echo "$NAME not forwarding"
        exit 3
    fi
;;
*)
    N=/etc/init.d/$NAME
    echo "Usage: $N {start|stop|restart|status}" >&2
    exit 1
;;
esac
exit 0
```

9.2 Definition of the firewall group in the CIB

Algorithm 1 Internal XML representation of the firewall group in the Cluster Information Base (CIB).

```
<group id="groupFirewall">
  <primitive class="lsb" id="resFirewall" type="firewall">
    <operations id="resFirewall-operations">
      <op id="resFirewall-op-monitor-15" interval="15" name="monitor"
        start-delay="15" timeout="15"/>
    </operations>
    <instance_attributes id="resFirewall-instance_attributes"/>
  </primitive>
  <primitive class="ocf" id="resIPext" provider="heartbeat"
    type="IPAddr2">
    <operations id="resIPext-operations">
      <op id="resIPext-op-monitor-10s" interval="10s"
        name="monitor" start-delay="5s" timeout="20s"
        on_fail="restart"/>
    </operations>
    <meta_attributes id="resIPext-meta_attributes">
      <nvpair id="meta-IPext-thr" name="migration-threshold"
        value="3"/>
    </meta_attributes>
    <instance_attributes id="resIPext-instance_attributes">
      <nvpair id="IPext-ip" name="ip" value="10.0.0.1"/>
      <nvpair id="IPext-nic" name="nic" value="eth0"/>
      <nvpair id="IPext-nm" name="cidr_netmask" value="24"/>
    </instance_attributes>
  </primitive>
  <primitive class="ocf" id="resIPint" provider="heartbeat"
    type="IPAddr2">
    <operations id="resIPint-operations">
      <op id="resIPint-op-monitor-10s" interval="10s"
        name="monitor" start-delay="5s" timeout="20s"
        on_fail="restart"/>
    </operations>
    <meta_attributes id="resIPint-meta_attributes">
      <nvpair id="meta-IPint-thr" name="migration-threshold"
        value="3"/>
    </meta_attributes>
    <instance_attributes id="resIPint-instance_attributes">
      <nvpair id="IPint-ip" name="ip" value="192.168.0.1"/>
      <nvpair id="IPint-nic" name="nic" value="eth1"/>
      <nvpair id="IPint-nm" name="cidr_netmask" value="24"/>
    </instance_attributes>
  </primitive>
  <primitive class="ocf" id="resIPdmz" provider="heartbeat"
    type="IPAddr2">
    <operations id="resIPdmz-operations">
      <op id="resIPint-op-monitor-10s" interval="10s"
        name="monitor" start-delay="5s" timeout="20s"
        on_fail="restart"/>
    </operations>
    <meta_attributes id="resIPdmz-meta_attributes">
      <nvpair id="meta-IPdmz-thr" name="migration-threshold"
        value="3"/>
    </meta_attributes>
    <instance_attributes id="resIPdmzinstance_attributes">
      <nvpair id="IPdmz-ip" name="ip" value="192.168.0.1"/>
      <nvpair id="IPdmz-nic" name="nic" value="eth1"/>
      <nvpair id="IPdmz-nm" name="cidr_netmask" value="24"/>
    </instance_attributes>
  </primitive>
</group>
```