

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6  
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

---

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England  
Don Mills, Ontario • Sydney • Mexico City  
Madrid • Amsterdam



# 26 Hochverfügbare Firewalls

Wenn Sie verschiedene Firewalls geplant und implementiert haben, werden Sie irgendwann an den Punkt geraten, wo Ihre erste Firewall hochverfügbar sein soll. Das bedeutet, dass die Internetverbindung trotz eines Ausfalls des Firewall-Systems weiter gewährleistet sein muss. Dieses Kapitel zeigt Ihnen verschiedene Wege, das Ziel zu erreichen. Dabei betrachte ich auch die ganz neue und noch nicht ganz fertige Zustandssynchronisation über `ct_sync`.

## 26.1 Was ist Hochverfügbarkeit, und wo ist das Problem?

Hochverfügbarkeit ist eine heute übliche Anforderung an geschäftskritische Systeme. In vielen Unternehmen zählt auch die Firewall hierzu. Ein Ausfall der Firewall oder der Internetverbindung verursacht hohe Kosten. Potenzielle Kunden können nicht auf die Website gelangen, und wichtige E-Mails erreichen nicht ihren Empfänger. Wenn eine Firewall hochverfügbar ist, bedeutet das, dass ein Ausfall der Funktion sehr selten ist.

Die Verfügbarkeit wird in Prozent gemessen. Wenn Ihre Firewall eine 99%ige Verfügbarkeit aufweisen soll, bedeutet das, dass sie nur 1% im Jahr ausfallen darf. Bei 365 Tagen sind das 3 Tage, 15 Stunden und 36 Minuten. Dies können Sie vielleicht noch mit einem einfachen System gewährleisten. Bei einem Ausfall pro Jahr haben Sie sicherlich innerhalb von 1 bis 2 Tagen Ersatz geschaffen. Meist wird aber eine Verfügbarkeit von 99,9% oder sogar 99,99% gefordert. Das bedeutet, dass das System nur 0,1% (8 Stunden, 46 Minuten) oder gar 0,01% (52 Minuten) im Jahr ausfallen darf. Ein einfacher Stromausfall, Upgrade des Betriebssystems oder Reboot bereitet da unter Umständen schon Kopfschmerzen.

Falls Sie eine derartige Verfügbarkeit garantieren müssen, können Sie das nur leisten, wenn im Falle des Ausfalls des primären Systems die Funktion durch ein zweites Backup-System aufrechterhalten wird. Daher werden derartige Lösungen meist als Cluster aus zwei Knoten implementiert, die sich gegenseitig ersetzen können.

Bei einem einfachen Hochverfügbarkeits-Cluster ist einer der beiden Knoten aktiv (Master) und stellt die Funktion zur Verfügung, während der zweite Knoten passiv als Hot-Standby nur im Fehlerfall die Funktion des Masters übernimmt (Slave, Hot-Standby). Dies nennt man auch einen Hot-Standby-Cluster. Fortgeschrittene Cluster können die Hochverfügbarkeit auch mit zwei aktiven Knoten realisieren. Solange beide Knoten zur Verfügung stehen, wird die Funktion über beide Knoten realisiert.

Sobald ein Knoten ausfällt, übernimmt der zweite Knoten die komplette Funktion. In dieser Konstellation ist die Backup-Hardware nicht die meiste Zeit ungenutzt, sondern wird sinnvoll eingesetzt. Diese Cluster werden als Active-Active-Cluster bezeichnet.

Das wesentliche Problem bei einem Cluster ist die gemeinsame Nutzung der identischen Daten für die Bereitstellung der Funktion. Ein Datenbank- oder Webserver-Cluster muss sowohl auf dem Master als auch auf dem Hot-Standby-Knoten über identische Webseiten oder Datenbanken verfügen. Handelt es sich um statische Daten, so können diese einfach einmalig synchronisiert werden. Werden diese Daten jedoch bei der Verwendung des Systems dynamisch modifiziert, weil zum Beispiel neue Einträge in der Datenbank vorgenommen werden oder vorhandene Einträge geändert werden, so müssen diese Änderungen auch sofort auf den Hot-Standby übertragen (synchronisiert) werden. Ansonsten sind die Informationen bei einem anschließenden Ausfall des Masters verloren. Bei Datenbanken und Webservern wird dies häufig durch ein gemeinsam genutztes Dateisystem (Shared Storage) zum Beispiel in einem Storage-Area-Network (SAN) realisiert.

Noch komplizierter wird die Realisierung eines Active-Active-Clusters, beim theoretisch auf beiden Knoten gleichzeitig ein Schreibzugriff auf das identische Feld in einer Datenbank erfolgen kann. Hier sind sehr intelligente Mechanismen erforderlich, um die Datenkonsistenz zu gewährleisten.

Wo ist nun das Problem bei Firewalls? Eine Firewall verfügt doch über einen statischen Regelsatz, der sehr einfach über alle Knoten in einem Cluster synchronisiert werden kann und nur selten verändert wird. Da die Modifikationen des Regelwerks durch einen geschulten Administrator vorgenommen werden, ist es sehr unwahrscheinlich, dass dieser die Regeln gleichzeitig auf zwei Systemen verändern wird. Ein Konflikt kann daher nicht entstehen. Dennoch tauchen Probleme auf, denn moderne Firewalls wie Iptables/Netfilter sind zustandsorientierte Firewalls. Diese Firewalls werten nicht nur ihre statischen Regeln aus, sondern auch eine Zustandstabelle, die sich in Abhängigkeit von den bereits betrachteten Paketen ändert. Die statischen Firewall-Regeln entscheiden nicht allein, ob ein Paket die Firewall passieren darf oder verworfen wird. Damit beim Ausfall des Masters der Hot-Standby die Funktion komplett übernehmen kann, müssten auch die Zustandsinformationen zwischen den Knoten synchronisiert werden.

## 26.2 Einfache Hochverfügbarkeit

Eine einfache Hochverfügbarkeit kann gewährleistet werden, wenn die Firewall die Zustände der Verbindungen nicht überwacht. Das bedeutet, dass Sie sowohl Regeln für eingehende Pakete als auch für ausgehende Pakete definieren müssen und kein NAT oder Masquerading verwenden dürfen. Sobald Sie diese Voraussetzungen gewährleisten können, ist es recht einfach, eine hochverfügbare Firewall aufzubauen. Sie müssen dann nur mit einer der verfügbaren Clusterlösungen die Ausfallsicherheit der Systeme herstellen und bei einem Ausfall die Übernahme der IP-Adressen von dem ausgefallenen System auf das Ersatzsystem veranlassen. Wenn auf bei-

den Systemen identische Regelsätze zum Einsatz kommen, ist die Funktionalität sofort wiederhergestellt. Da keine Zustandsinformationen gespeichert werden, müssen diese auch nicht zwischen den Firewalls synchronisiert werden. Sämtliche Verbindungen laufen ohne Unterbrechung weiter.

Nun ist dieses Buch aber kein Buch über `ipchains` (eine zustandslose Firewall), sondern betrachtet mit `Netfilter/Iptables` eine zustandsorientierte Firewall. Sicherlich kann man auch mit `Iptables` zustandslose Firewalls konfigurieren, allerdings ist das unter sicherheitstechnischen Gesichtspunkten nicht sinnvoll. Daher werde ich diesen Punkt hier nicht weiter ausführen und direkt zur Hochverfügbarkeit bei einer zustandsorientierten Firewall übergehen.

## 26.3 Hochverfügbarkeit bei zustandsorientierten Firewalls

Natürlich können Sie auch bei einer zustandsorientierten Firewall denselben Ansatz wählen, wie ich ihn gerade bei der zustandslosen Firewall beschrieben habe. Sie wählen eine Cluster-Software Ihrer Wahl (zum Beispiel `Heartbeat` oder `KeepAlived`) und konfigurieren diese so, dass sie bei Ausfall des Master-Nodes automatisch ein Fail-Over zum Hot-Standby durchführt. Bei diesem Fail-Over werden automatisch von der Cluster-Software auch die IP-Adressen übertragen. Verfügt nun der Hot-Standby-Node über ein identisches Regelwerk, ist eine gewisse Funktionalität sofort wiederhergestellt. Sie können sofort wieder neue Verbindungen über die Firewall aufbauen. Leider sind alle zum Zeitpunkt des Fail-Overs aufgebauten Verbindungen nun tot. Die Information, welche Verbindungen von der ausgefallenen Firewall zugelassen wurden, befand sich in deren Zustandstabelle. Deren Inhalt ist verloren und steht der neuen Firewall nicht zur Verfügung. Daher kennt diese nicht die vor dem Fail-Over existenten Verbindungen und wird diese nicht erlauben.

Wenn Sie den Verlust der aufgebauten Verbindungen beim Fail-Over akzeptieren können, ist die Realisierung dieser Hochverfügbarkeitslösung die einfachste und eine sehr unproblematische Variante.

### Tipp: Automatische Erkennung aufgebauter Verbindungen nach dem Fail-Over



Sie können mit dieser Lösung, wenn Sie kein NAT einsetzen, sogar in vielen Fällen echte Hochverfügbarkeit garantieren. Dafür müssen Sie jedoch Ihre Regeln unter Umständen anpassen.

Die Wiedererkennung aufgebauter Verbindungen nach einem Fail-Over ist insbesondere für Langzeit-TCP-Verbindungen wichtig. Das bedeutet, dass der Hot-Standby nach einem Fail-Over besonders diese Verbindungen erkennen und wieder in seine Zustandstabelle aufnehmen muss. Der `Connection-Tracking-Code` von `Netfilter` ist hierzu in der Lage. Bei dem Protokoll `TCP` werden die folgenden zwei Pakete als neue Pakete (State: `NEW`) akzeptiert:

- TCP-SYN-Pakete. Diese Pakete bauen normalerweise eine Verbindung auf.
- TCP-ACK-Pakete. Diese Pakete werden nach dem Aufbau einer Verbindung ausgetauscht. Dieses Paket überführt eine unbekannte Verbindung sofort in den Zustand `ESTABLISHED`.

Wenn Sie folgende Regeln verwenden, kann Ihre Firewall nach einem Fail-Over vorher vorhandene Verbindungen erkennen und automatisch wieder erlauben:

```
$INTDEV=eth0
$EXTDEV=eth1
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 22 -m state \
  --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state ESTABLISHED,RELATED -j ACCEPT
```

Wenn vor dem Fail-Over nun ein interner Client eine SSH-Verbindung zu einem Server in dem Internet aufbaut, wird er den kompletten TCP-Handshake durchlaufen, und die Firewall wird die Verbindung in Ihrer Zustandstabelle eintragen. Alle weiteren Pakete, die der Client nun mit dem Server austauscht, tragen bis zur Beendigung der Verbindung lediglich das TCP-ACK-Bit. Kommt es nun zum Fail-Over, so verfügt der Hot-Standby über den identischen Regelsatz, aber nicht über die Zustandstabelle. Die Verbindung ist daher unbekannt. Wenn nun der Server ein weiteres TCP-ACK-Paket an den Client sendet, wird die Firewall dieses Paket nicht zulassen, da sie die Verbindung nicht kennt. Sendet der Client jedoch ein TCP-ACK-Paket an den Server, so akzeptiert die Firewall auch dieses Paket als Verbindungsaufbau und trägt die Verbindung sofort wieder in der Zustandstabelle mit dem Zustand `Established` ein. Alle weiteren Pakete dieser Verbindung, des Clients und des Servers, dürfen dann auch den Hot-Standby passieren.

Diese Funktionalität ist nicht gegeben, wenn Sie in den Regeln zusätzlich die Option `--syn` angeben:

```
$INTDEV=eth0
$EXTDEV=eth1
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 22 \
  --syn -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state ESTABLISHED,RELATED -j ACCEPT
```

Nun wird nur dann ein Paket als Verbindungsaufbau akzeptiert, wenn es sich tatsächlich um ein TCP-SYN-Paket handelt. Eine Wiederaufnahme einer alten Verbindung nach einem Fail-Over oder

auch einem Neustart der Firewall ist nicht mehr möglich. Die Verbindungen »hängen« und müssen komplett neu aufgebaut werden.

Auch wenn Sie den Parameter `/proc/sys/net/ipv4/netfilter/ip_conntrack_tcp_loose=0` setzen, funktioniert dies nicht mehr (siehe Abschnitt 19.4.8).

Außerdem funktioniert die Wiederaufnahme von alten Verbindungen nicht richtig, wenn die Firewall zusätzlich noch ein Masquering oder ein N:1-NAT durchführt.

## 26.4 Praktische Implementierung mit KeepAlived

Um nun eine derartige hochverfügbare Firewall aus zwei Knoten zu implementieren, benötigen Sie eine Software, mit der Sie die Verfügbarkeit prüfen und einen Fail-Over durchführen können. Ein Kandidat für diese Aufgabe ist KeepAlived. Diese Software von dem Linux-Virtual-Server-Projekt (LVS) ist ein Userspace-Daemon, der die Gesundheit der Cluster-Knoten überwacht und einen Fail-Over durchführen kann. Sie können die KeepAlived-Software von der Homepage <http://www.keepalived.org> herunterladen. Die KeepAlived-Software ist aber auch Bestandteil von vielen Distributionen. Prüfen Sie zunächst, ob Ihre Distribution ein Paket zur Verfügung stellt.

Nach der Installation müssen Sie KeepAlived konfigurieren. KeepAlived soll eine hochverfügbare Firewall realisieren. Wenn wir für einen Moment die Firewall-Regeln vernachlässigen, handelt es sich bei der Funktion, die wir hochverfügbar anbieten möchten, um die Routing-Funktionalität der Firewall. Dass die Firewall nebenbei auch noch filtert, ist ja nur ein zusätzliches Bonbon. Speziell für diesen Zweck implementiert KeepAlived das Virtual Router Redundancy Protocol (VRRP, RFC2338).

### Tipp



Das VRRP-Protokoll ist möglicherweise durch Patente geschützt. Sowohl Cisco (<http://www.ietf.org/ietf/IPR/VRRP-CISCO>) als auch IBM (<http://www.ietf.org/ietf/IPR/NAT-VRRP-IBM>) behaupten, dass das VRRP-Protokoll ihre Patente verletzen würde. Deshalb hat das OpenBSD-Projekt mit dem Common Address Redundant Protocol (CARP) ein patentfreies und sicheres Protokoll geschaffen. Mit UCARP, einem Userspace-Daemon (<http://www.ucarp.org>), kann dieses Protokoll auf Linux und BSD benutzt werden. Ich verwende hier den KeepAlived mit VRRP, da dies auch die bevorzugte Variante des Netfilter-Teams ist.

So können Sie zwei oder mehr redundante Router zusammenfassen, so dass sie sich als ein virtueller Router dem lokalen Netz präsentieren. Hierzu beobachtet KeepAlived mit dem VRRP-Protokoll die redundanten Router und deren Gesundheit und wählt einen Master-Router, der zusätzlich zu seiner realen IP-Adresse die

virtuelle Adresse des virtuellen Routers erhält. Alle weiteren Router werden als Slave bezeichnet. Diese kontrollieren die Gesundheit des Masters und übernehmen bei dessen Ausfall die virtuelle Router-IP-Adresse. Ist der Master später wieder verfügbar, so übergibt der Slave die virtuelle Router-IP-Adresse wieder an den Master.

### Tipp



Das VRRP-Protokoll ist von der Internet Engineering Task Force (IETF) spezifiziert worden und wird von vielen kommerziellen Routern ebenfalls unterstützt.

Die Konfiguration des KeepAlived-Daemons erfolgt in der Datei `/etc/keepalived/keepalived.conf`. Eine Beispieldatei für einen Master könnte folgendermaßen aussehen:

```
vrrp_sync_group VG1 {
    group {
        eth0
        eth1
    }
}
! Interne virtuelle IP-Adresse
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 1
    priority 100
    authentication {
        auth_type AH
        auth_pass password
    }
    virtual_ipaddress {
        192.168.1.1/24 brd 192.168.1.255 dev eth0
    }
}
! Externe virtuelle IP-Adresse
vrrp_instance VE_1 {
    state MASTER
    interface eth1
    lvs_sync_daemon_interface eth0
    virtual_router_id 2
    priority 100
    authentication {
        auth_type AH
```

```
    auth_pass password
  }
  virtual_ipaddress {
    192.168.2.1/24 brd 192.168.1.255 dev eth1
  }
}
```

Die entsprechende Datei für den Slave hat folgenden Inhalt:

```
vrrp_sync_group VG1 {
  group {
    eth0
    eth1
  }
}
vrrp_instance VI_1 {
  state BACKUP
  interface eth0
  virtual_router_id 1
  priority 50
  authentication {
    auth_type AH
    auth_pass password
  }
  virtual_ipaddress {
    192.168.1.1/24 brd 192.168.1.255 dev eth0
  }
}
vrrp_instance VE_1 {
  state BACKUP
  interface eth1
  lvs_sync_daemon_interface eth0
  virtual_router_id 2
  priority 50
  authentication {
    auth_type AH
    auth_pass password
  }
  virtual_ipaddress {
    192.168.2.1/24 brd 192.168.1.255 dev eth1
  }
}
```

Wichtig ist, dass sowohl der Slave als auch der Master die identische `virtual_router_id` für die jeweilige Instanz besitzen. Jede Instanz ist für eine virtuelle IP-Adresse des Routers verantwortlich. Damit diese IP-Adressen beim Ausfall

einer IP-Adresse auch beide gemeinsam übertragen werden, werden sie in einer `vrrp_sync_group` zusammengefasst. Die vergebene Priorität entscheidet, welches der beiden Systeme der Master ist. Bei mehreren Slaves variieren Sie die Priorität leicht zwischen den verschiedenen Slaves (50, 49, 48 etc.). Mit dem Parameter `state` können Sie entscheiden, in welchem Zustand der KeepAlived-Daemon auf diesem System starten soll: `MASTER` oder `BACKUP`. Die `virtual_ipaddress` ist einmal die interne und einmal die externe IP-Adresse des virtuellen Routers. Bei der Authentifizierung sollten Sie darauf achten, dass Sie AH wählen, da die PASS-Authentifizierung das Kennwort im Klartext austauscht. Der Parameter `lvs_sync_daemon_interface` definiert die Netzwerkkarte, über die die Kommunikation zwischen den KeepAlived-Daemons für diese Instanz erfolgen soll. Auf einer Firewall ist es sinnvoll, diese Informationen in dem internen Netz zu halten oder sogar auf ein eigenes getrenntes Netz auszuweichen.

Wenn Sie nun den KeepAlived-Daemon starten, wird das erste System sich als Master konfigurieren und die virtuelle IP-Adresse übernehmen. Sobald der Master ausfällt, wird der Slave die Funktion übernehmen. Sie müssen nun nur allen internen Clients als Standard-Gateway die virtuelle IP-Adresse des Routers zuweisen: `192.168.1.1/24`. Dann können Sie Ihre neue hochverfügbare Firewall nutzen!

## 26.5 Hochverfügbarkeit und Masquerading/NAT

Sobald Sie SNAT oder Masquerading nutzen, wird Ihr hochverfügbarer Firewall-Cluster nicht mehr richtig funktionieren. Um das zu verstehen, müssen Sie sich vor Augen führen, dass eine Firewall bei einem Masquerading oder N:1-SNAT<sup>1</sup> die Firewall häufig nicht nur die IP-Adresse, sondern auch den TCP- oder UDP-Port austauscht. Stellen Sie sich vor, ein Client1 baut durch Ihre Firewall eine Verbindung zu einem Webserver in dem Internet auf. Der Client1 verwendet den Quellport 1027. Dieser Port ist jedoch auf der Firewall bereits durch eine andere genattete Verbindung belegt. Dann wird die Firewall beim NAT nicht nur die IP-Adresse, sondern auch den Quellport modifizieren. Hierfür wählt sie den nächsten geeigneten freien Port aus. Dieser ist natürlich stark abhängig von den anderen gerade aktiven und in der Vergangenheit aktiven Verbindungen. In unserem Beispiel (Abbildung 26.1) ist es der Port 1275. Für den Webserver wird die Verbindung nun von `3.0.0.1:1275` aufgebaut.

Nach dem Fail-Over wird der zweite Knoten des Firewall-Clusters die Verbindung wieder aufnehmen. Da sich dieser aber nicht in demselben Zustand befindet wie der alte Master vor dem Fail-Over, wird er bei der Wahl des NAT-Ports entweder den Port gar nicht verändern, da dieser Port 1027 noch frei ist, oder sicherlich einen anderen wählen. Der Webserver wird die Pakete aber nicht zu der Verbindung zählen, da sie über einen falschen Quellport verfügen und die Pakete verwerfen. Die Verbindung hängt bis zum Timeout.

---

<sup>1</sup> Ein N:1-SNAT ist ein Source-NAT, bei dem viele verschiedene Absender-IP-Adressen gegen eine IP-Adresse ausgetauscht werden. Dies ist der häufigste Fall des SNAT. Masquerading bezeichnet unter Linux dann den Umstand, dass diese IP-Adresse dynamisch zugewiesen wird.

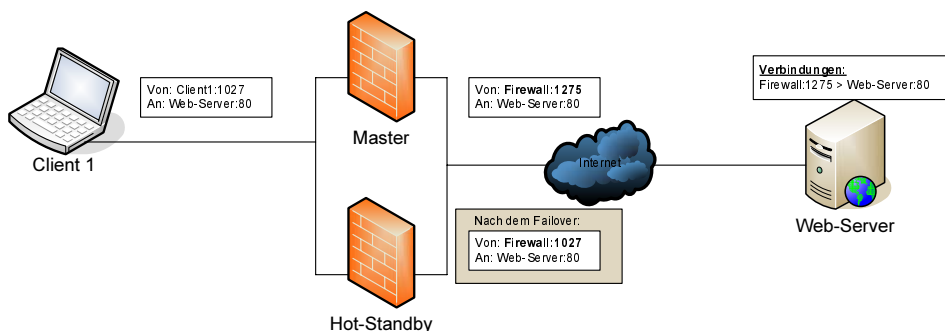


Abbildung 26.1: NAT (Network Address and Port Translation) macht bei einem einfachen Firewall-Cluster einen Strich durch die Rechnung.

Sobald Sie NAT einsetzen, ist es daher zwingend erforderlich, dass die Zustandstabelle von dem Master auf alle Slaves in dem Cluster synchronisiert wird, da die hier gespeicherten Informationen (NAT) für die Modifikation der Pakete beim NAT der Verbindung zwingend erforderlich sind. Diese Zustandssynchronisation kann mit `ct_sync` durchgeführt werden.

## 26.6 Zustandssynchronisation mit ct\_sync

Viele kommerzielle Firewalls verfügen seit einigen Jahren über eine Synchronisation der Zustandstabelle in einem Firewall-Cluster. Selbst OpenBSD hat mit `pfsync` eine seit Jahren stabile Implementierung dieser Funktion. In dem Linux-Kernel ist bis heute keine derartige Lösung integriert. Bisher existiert nur Beta-Code, der sich jedoch mit einigen wenigen Einschränkungen bereits sehr gut einsetzen lässt. Dieser Code ist für den Linux-Kernel 2.4.26 und den Linux-Kernel 2.6.10 verfügbar. Er implementiert mit `ct_sync` ein Connection Tracking-Synchronisationsframework, das den Aufbau eines Hot-Standby-Clusters ermöglicht. Da die Entwicklung hier noch stark im Fluss ist, empfehle ich Ihnen bei Interesse die Netfilter-Failover-Mailingliste zu abonnieren. Das Volumen dieser Mailingliste ist aktuell sehr gering und stellt daher keine Belastung dar. Sie finden die Mailingliste auf <http://lists.netfilter.org/mailman/listinfo/netfilter-failover>.



### Achtung

Der aktuelle Entwicklungsstand (Oktober 2005) erlaubt noch nicht die Verwendung von Conntrack- oder NAT-Helfermodulen. So können FTP-Datenverbindungen, die von dem `ip_conntrack_ftp`-Helfermodul erkannt und zugelassen werden, noch nicht synchronisiert werden. Im Gegenteil, die Anwendung einiger Module hat auch schon in Kombination mit `ct_sync` zur Kernel-Panic geführt.

Wenn Sie auf diese Helfermodule für die Funktionalität Ihrer Firewall angewiesen sind, ist `ct_sync` aktuell für Sie nicht einsetzbar. Sie sollten die Netfilter-Failover-Mailingliste beobachten, um diesbezügliche Änderungen zu erfahren.



### Tipp

Neueste Entwicklungen ermöglichen auch schon den Aufbau eines Active-Active-Clusters. Dieser Code ist zum aktuellen Zeitpunkt (Oktober 2005) aber noch sehr fehlerbehaftet und kann lediglich unter Verzicht auf NAT eingesetzt werden.

Wenn Sie mit dem Code experimentieren möchten, können Sie ihn mit folgendem Befehl aus dem Netfilter-Subversion-Server auschecken:

```
$ svn co http://svn.netfilter.org/netfilter/branches/netfilter-ha/linux-2.6-actact
```

## 26.6.1 Installation

Der Code ist in zwei verschiedenen Versionen für den Linux-Kernel 2.6.10 und den Linux-Kernel 2.4.26 verfügbar. Die aktuelle Entwicklung findet im Moment in dem Linux-Kernel 2.6.10 statt. Sie können den Code mit den folgenden Befehlen aus dem Netfilter-Subversion-Server auschecken:

```
$ svn co http://svn.netfilter.org/netfilter/branches/netfilter-ha/linux-2.6
$ svn co http://svn.netfilter.org/netfilter/trunk/netfilter-ha
```

Um über Updates auf dem Laufenden zu bleiben, sollten Sie in regelmäßigen Abständen in die angelegten Verzeichnisse wechseln und dort den Befehl `svn update` aufrufen:

```
$ cd linux-2.6
$ svn update
Revision 4351.
```

Sollte ein Update des Codes erfolgt sein, werden Ihnen die betroffenen Dateien beim Update angezeigt.

Nun müssen Sie Hand an Ihren Kernel legen, um `ct_sync` zu übersetzen. Zunächst benötigen Sie einen Vanilla-Kernel von <http://www.kernel.org>. Entweder wählen Sie die Version 2.4.26 oder 2.6.10. Wenn Sie dieses Buch lesen, werden möglicherweise auch schon weitere Kernel unterstützt. Entpacken Sie den Kernel, und wechseln Sie in das entstandene Verzeichnis. Dann kopieren Sie aus dem `ct_sync`-Quelltext das Verzeichnis `./patches` in den Kernel-Quelltextbaum. Außerdem kopieren Sie einige Dateien aus dem Verzeichnis `./ct_sync` in den Kernel-Quelltextbaum:

```
$ tar xjf linux-2.6.10.tar.bz2
$ cd linux-2.6.10
$ cp -r ../svn/linux-2.6/patches .
$ cp ../svn/linux-2.6/ct_sync/*.h include/linux/netfilter_ipv4/
$ cp ../svn/linux-2.6/ct_sync/*.c net/ipv4/netfilter/
```

Nun müssen Sie die kopierten Patches anwenden. Hierfür benötigen Sie den Befehl `quilt`. Falls dieser Befehl nicht Bestandteil Ihrer Linux-Distribution ist, können Sie ihn von <http://savannah.nongnu.org/projects/quilt> herunterladen und installieren. Um die Patches anzuzeigen und anschließend anzuwenden, verwenden Sie die folgenden Befehle:

```
$ quilt unapplied
$ quilt push -a
```

Nun können Sie den Kernel wie gewohnt konfigurieren und übersetzen. Wenn Sie bereits einen Kernel mit derselben Version verwenden, editieren Sie bitte die Datei `Makefile` im Wurzelverzeichnis des Kernel-Quelltextbaums und tragen dort eine `EXTRAVERSION` ein, um spätere Konflikte der Kernel untereinander zu vermeiden:

```
EXTRAVERSION = -ha
```

Für die Konfiguration und Übersetzung verwenden Sie die folgenden Befehle:

```
$ make xconfig
$ make dep
$ make bzImage
$ make modules
$ make modules_install
$ make install
```

Bei der Konfiguration sollten Sie dann die Synchronisation der Zustandstabelle auswählen (Abbildung 26.3). Wenn Sie den Active-Active-Cluster konfigurieren, achten Sie bitte auf Abbildung 26.2. Diese Einstellungen finden Sie am Ende unter *Device Drivers/Networking Support/Networking Options/Network Paket Filtering (replaces ipchains)/Netfilter Configuration*. Damit Sie die Optionen für den Active-Active-Cluster angezeigt bekommen, müssen Sie NAT deaktivieren!

Nach dem Reboot des neuen Kernels steht Ihnen die Zustandssynchronisation zur Verfügung.

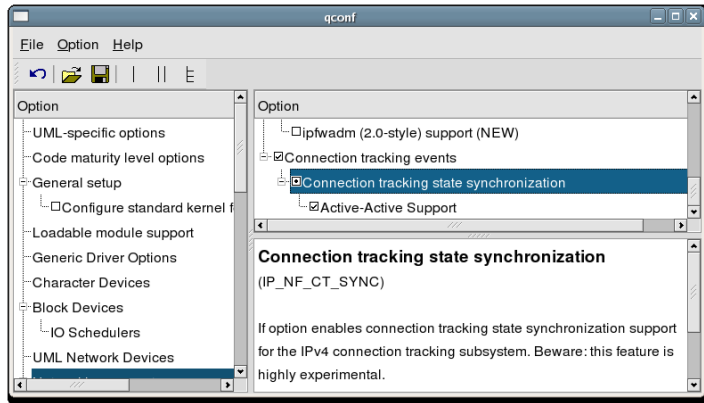


Abbildung 26.2: Für den Active-Active-Cluster muss NAT deaktiviert sein.

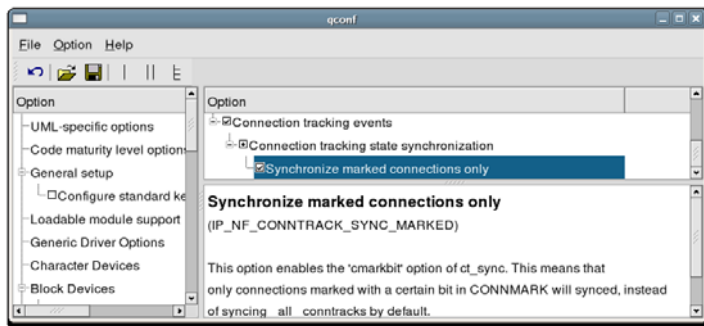


Abbildung 26.3: Bei dem Hot-Standby-Cluster können Sie die zu synchronisierenden Verbindungen durch Markierungen auswählen.

## 26.6.2 Funktion von ct\_sync

Die Synchronisation der Zustandstabelle läuft in mehreren Schritten ab. Der aktive Master analysiert jedes Paket, ermittelt die Zustandsinformationen des Pakets und aktualisiert die Zustandstabelle. Diese Aktualisierung löst einen Connection Tracking-Event aus, der in einer Event-Queue gespeichert wird. Diese Queue wird von dem `ct_sync`-Sender-Thread abgearbeitet. Das maximale Alter der Events in dieser Queue und die maximale Größe der Queue, bevor sie abgearbeitet werden muss, kann konfiguriert werden. Der `ct_sync`-Sender-Thread verpackt bis zu 4 Nachrichten in einem Paket und überträgt dieses Paket mit einem Multicast-Protokoll an alle Slaves. Auf dem Slave werden die Pakete entgegengenommen, von der Zustandsüberwachung ignoriert (Option `notrack=1`) und an den `ct_sync`-Receive-Thread übergeben. Der extrahiert die Meldungen und aktualisiert die Zustandstabelle. Auch im `ct_sync`-Receive-Thread existiert eine Queue, deren maximale Größe eingestellt werden kann.

Für die Übertragung der Pakete wird ein UDP-basiertes Multicast-Protokoll verwendet, das das Netfilter-Team speziell für diesen Zweck entwickelt hat. Dieses Protokoll bietet keinerlei Authentifizierung oder Verschlüsselung und ist auf ein sicheres Medium, zum Beispiel ein dediziertes Netzwerk (Cross-Over Kabel), angewiesen. Als Schutz vor Paketverlust versieht der Master jedes Paket mit einer Sequenznummer, so dass der Slave die Vollständigkeit kontrollieren und ein verlorenes Paket neu anfordern kann.

Das Protokoll unterstützt außerdem die initiale Synchronisation eines Slaves mit einem Master. Dies ist erforderlich, da nach einem Neustart eines Slaves, zum Beispiel aus Wartungsgründen, der Slave zwar Synchronisationsmeldungen erhält, es sich dabei aber nur um Änderungen der Zustandstabelle handelt. Der Slave besitzt aber nach dem Neustart eine leere Zustandstabelle. Um diese zu füllen, müssen sämtliche Informationen des Masters übertragen werden. Diese Komplettsynchronisation wird auch durchgeführt, wenn zu viele Pakete zwischen dem Master und dem Slave verloren gegangen sind oder der Master ein verlorenes Paket nicht mehr in seinem Backlog findet. Die Anzahl der verlorenen Pakete, ab der eine Neusynchronisation auftritt, ist ebenfalls konfigurierbar.

Bei dem Laden des `ct_sync`-Kernel-Moduls können Sie vier Parameter angeben:

- `syncdev=ethX`: Die Angabe der Netzwerkkarte für die Synchronisation ist erforderlich. Es sollte sich dabei um eine Netzwerkkarte handeln, die über ein dediziertes Netzwerk mit den anderen Knoten in dem Cluster verbunden ist, denn das Protokoll bietet keinen Schutz vor Angriffen. Ein serielles Kabel genügt nicht. Eine Resynchronisation erfordert hohe Bandbreiten von mindestens 1,5 MBit/s. In vielen Fällen genügt hier ein Cross-Over-Kabel. Es kann keine zweite Netzwerkkarte genutzt werden, um für Ausfallsicherheit zu sorgen.
- `l2drop=0|1`: Hiermit geben Sie an, ob der Slave einen Layer-2-Drop durchführen soll. Damit ist es möglich, sämtliche Knoten bereits mit identischer IP-Adresskonfiguration auszustatten. In einem normalen Setup führt dies zu Adresskonflikten. Der Layer-2-Drop verwirft jedoch bereits auf der Schicht 2 sämtliche Pakete auf allen Netzwerkkarten außer dem `syncdev`. So können Sie schon die IP-Adressen aktivieren, Dienste starten und alle Knoten identisch konfigurieren. Der jeweilige Master deaktiviert den Layer-2-Drop und bietet seine Dienste an.
- `notrack=0|1`: Mit dieser Option ignoriert der lokale Connection-Tracking-Code sämtliche Pakete, die über die `syncdev`-Schnittstelle transportiert werden. Die Synchronisationspakete lösen dadurch nicht selbst auch noch Änderungen der Zustandstabelle aus.
- `cmarkbit=0-31`: Diese Option steht zur Verfügung, wenn Sie bei der Übersetzung des Kernel die Unterstützung für das `CONNMARK`-Target in Kombination mit der Synchronisation aktiviert haben. Dann können Sie durch Markierung der Verbindungen die Verbindungen auswählen, die synchronisiert werden sollen. Das ist nützlich, da häufig die Synchronisation von zum Beispiel DNS-Verkehr und ICMP-PING-Verkehr unsinnig ist. Diese »Verbindungen« bestehen nur aus ei-

nem Request- und einem Response-Paket. Bei einem Fail-Over kann der Client das Request-Paket auch erneut senden. Häufig tritt durch den Fail-Over sowieso eine Verzögerung auf, die bei diesen Verbindungen zu einem Timeout führen kann.



### Achtung

Ich beschreibe hier die Funktionen des 2.6.10-Kernel-Patches. Der 2.4.26-Kernel-Patch verwendet teilweise andere Optionen und verfügt noch nicht über das mächtige `/proc`-Interface, das im Weiteren beschrieben wird. Möglicherweise ist der 2.4.26-Code, wenn Sie dieses Buch lesen, bereits aktualisiert worden. Ansonsten möchte ich Sie auf die dem Code beiliegende README-Datei verweisen.

Sobald Sie das Modul mit den geeigneten Optionen geladen haben, müssen Sie dem Modul mitteilen, in welchem Zustand sich das System befindet. Hierfür können Sie in dem `/proc`-Verzeichnis die Datei `/proc/sys/net/ipv4/netfilter/ct_sync/state` verwenden. Nach dem Laden des Moduls befindet sich in dieser Datei eine `0`. Dadurch wird effektiv das Modul deaktiviert. Durch das Schreiben von `1` wird der Knoten zum Slave. Eine `2` macht den Knoten zum Master<sup>2</sup>. Zusätzlich befinden sich in dem Verzeichnis `/proc/sys/net/ipv4/netfilter/ct_sync` noch weitere Dateien, deren Inhalt Sie modifizieren und so das Verhalten von `ct_sync` anpassen können:

- `maxage`: Sobald ein Event in der Sender-Event-Queue dieses Alter erreicht, muss er übertragen werden.
- `send_burst`: Sobald die Sender-Event-Queue diese Anzahl an Meldungen enthält, muss sie verarbeitet werden.
- `recv_burst`: Sobald der Slave diese Anzahl an Paketen erhalten hat, muss er die Meldungen extrahieren und die Zustandstabelle aktualisieren.
- `recovery_threshold`: Beim Verlust von Paketen wird ab diesem Schwellenwert eine komplette Neusynchronisation durchgeführt.



### Tipp

Laut Aussagen des Netfilter-Teams von Sommer 2005 wird diese `/proc`-Schnittstelle in der nächsten Zeit unverändert bleiben.

<sup>2</sup>Achtung, auch dies hat sich gegenüber dem 2.4.26-Kernel-Patch geändert. Dort bedeutete eine `0` Slave und eine `1` Master!

**Achtung**

Wenn Sie das `ct_sync`-Modul auf dem Linux-Kernel 2.6.10 oder neuer verwenden, müssen Sie das seit 2.6.9 in dem Kernel befindliche TCP-Window-Tracking durch Netfilter auf dem Slave abschalten. Dies kollidiert aktuell mit der Synchronisation. Das können Sie sehr einfach erreichen, indem Sie im `/proc`-Verzeichnis die folgende Datei schreiben:

```
echo 1 > /proc/sys/net/ipv4/netfilter/ip_conntrack_tcp_be_liberal
```

**26.6.3 Aufbau des ct\_sync-Clusters**

Um nun einen `ct_sync`-Cluster aufzubauen, können Sie zwei unterschiedliche Wege einschlagen. Entweder Sie beginnen mit dem im Abschnitt 26.4 beschriebenen Cluster und erweitern diesen um die `ct_sync`-Funktionalität, oder Sie beginnen mit zwei identisch konfigurierten Maschinen, bei denen Sie zum Test zunächst manuell `ct_sync` bedienen, und erweitern diese anschließend um eine Heartbeat-Software wie KeepAlived oder Linux-HA-Heartbeat (<http://www.linux-ha.org>). Ich werde hier den zweiten Weg einschlagen.

Beim Einsatz des Layer-2-Drops ist bei einem Fail-Over der Take-Over der IP-Adresse nicht mehr zwingend erforderlich. Daher können Sie auch andere Lösungen als KeepAlived einsetzen! Die eingesetzte Software muss lediglich in der Lage sein, die Gesundheit des Masters zu überwachen und bei einem Ausfall auf dem Slave ein Skript aufzurufen. Dieses Skript aktiviert den Slave zum Master. Dabei versendet der neue Master über das Synchronisationsprotokoll Nachrichten, die den alten Master automatisch zum Slave degradieren. Dieses Skript darf sehr einfach sein:

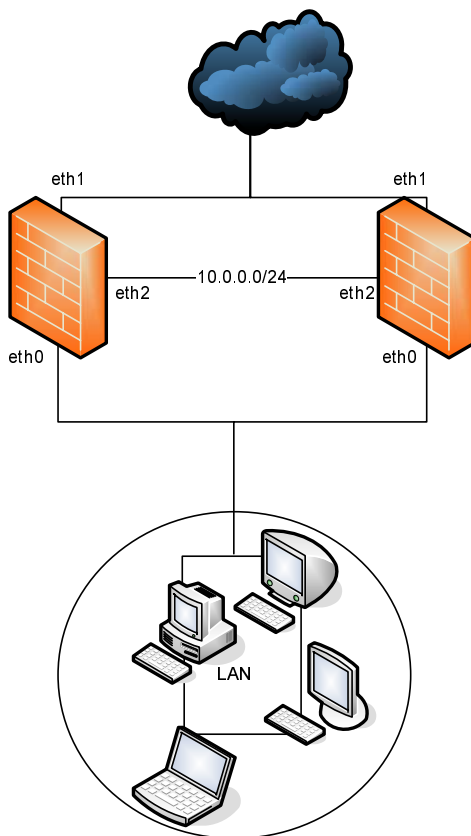
*Listing 26.1: Das Skript `script_master.sh` befördert einen Slave.*

```
#!/bin/sh
/usr/bin/logger -p kernel.crit "Slave zum Master aktiviert."
/bin/echo 2 > /proc/sys/net/ipv4/netfilter/ct_sync/state
```

Bauen Sie zunächst Ihren Firewall-Cluster wie in Abbildung 26.4 auf. Achten Sie darauf, dass Ihre Cluster-Knoten über eine dedizierte Netzwerkkarte miteinander verbunden sind. Aktivieren Sie nun die Netzwerkkarte zur Synchronisation und weisen Sie ihr eindeutige Adressen zum Beispiel in dem Netzwerk 10.0.0.0/24 zu. Sie sollten nun die anderen Knoten in dem Synchronisationsnetzwerk pingen können. Laden Sie nun auf allen Knoten das `ct_sync`-Modul, bevor Sie die weiteren Netzwerkkarten aktivieren:

```
# modprobe ct_sync syncdev=eth2 l2drop=1 notrack=1 cmarkbit=30
```

Nun deklarieren Sie einen Knoten zum Master und die restlichen Knoten jeweils zum Slave. Geben Sie auf dem Master hierzu folgenden Befehl ein:



**Abbildung 26.4:** Der Firewall-Cluster besteht aus zwei Knoten, die sich gegenseitig ersetzen.

```
# echo 2 > /proc/sys/net/ipv4/netfilter/ct_sync/state
```

Auf dem Slave geben Sie den folgenden Befehl ein:

```
# echo 1 > /proc/sys/net/ipv4/netfilter/ct_sync/state
```

Konfigurieren Sie nun die IP-Adressen auf den internen und externen Netzwerkkarten auf allen Knoten identisch. Nun müssen Sie die Firewall-Regeln erzeugen und auf allen Knoten verteilen. Für einen ersten Test mögen die folgenden Regeln genügen:

```
INTDEV=eth0  
EXTDEV=eth1  
SYNCDEV=eth2
```

```
IPTABLES=/usr/sbin/iptables
```

```

SYSCTL=/usr/sbin/sysctl

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -F -t mangle

$IPTABLES -A INPUT -i $SYNCDEV -j ACCEPT
$IPTABLES -A OUTPUT -o $SYNCDEV -j ACCEPT
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -m state --state NEW -j ACCEPT

$IPTABLES -t mangle -A PREROUTING -p udp --dport 53 -j ACCEPT
$IPTABLES -t mangle -A PREROUTING -p icmp -j ACCEPT

$IPTABLES -t mangle -A PREROUTING -m state --state NEW -j CONNMARK --mark
    0x40000000

$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j MASQUERADE

```

Die letzte Regel markiert sämtliche Verbindungen mit der Zahl 0x4000000. Bei dieser Zahl ist als Connmark-Bit das Bit 30 gesetzt.

Wenn Sie nun Verbindungen über den Master aufbauen, sollten Sie die Verbindungen auf dem Master in der Verbindungstabelle beobachten können. Diese Verbindungen sollten auch auf dem Slave in der Verbindungstabelle auftauchen. Für Troubleshooting-Zwecke können Sie den Netzwerkverkehr zwischen den Knoten mit Ethereal überwachen. Hierfür ist in dem `ct_sync`-Quelltext auch ein Ethereal-Plugin vorhanden, das Sie in den Ethereal-Quelltext patchen können. Dieses Plugin kann das Protokoll analysieren und verständlich anzeigen (Abbildung 26.5).

### Tipp



Benutzen Sie Verbindungen, die über eine lange Zeit offen bleiben. Telnet- und SSH-Verbindungen haben sich hier für den Test bewährt. Eine HTTP-Verbindung wird nach dem Datentransport meist direkt wieder von dem Server geschlossen.

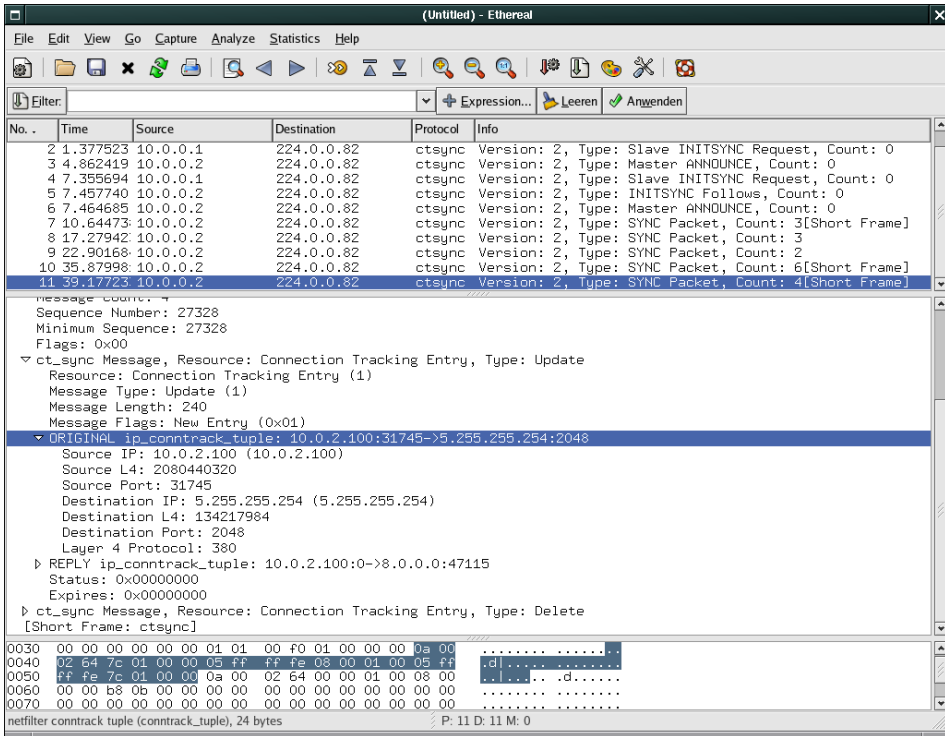


Abbildung 26.5: Ethereal kann mit einem Patch das ct\_sync-Multisync-Protokoll parsen.

Zur weiteren Fehleranalyse befindet sich in dem ct\_sync-Quelltext auch ein Werkzeug `cts_dump`, mit dem Sie ebenfalls die Nachrichten des Synchronisationsprotokolls anzeigen können.

Wenn Sie die Synchronisation einige Zeit betrachtet haben und sich von der Funktion überzeugt haben, können Sie manuell auf dem Slave das Skript aufrufen, das den Slave zum Master befördert. Der alte Master sollte den Layer2Drop aktivieren, und der Slave sollte sichtbar werden.

Wahrscheinlich werden die vorher von Ihnen aufgebauten Netzwerkverbindungen für eine gewisse Zeit »hängen«. Dies hängt mit der MAC-Adresse des neuen Masters zusammen. Der neue Master besitzt zwar die richtige IP-Adresse, aber die »falsche« MAC-Adresse. Erst nach Ablauf des ARP-Caches der Clients werden diese einen neuen ARP-Request aussenden, der von dem neuen Master beantwortet werden wird. Sie können das beschleunigen, indem Sie auf dem neuen Master den Befehl `arping -A <eigene IP-Adresse>` aufrufen. Dies ist ein *gratious ARP*, der die ARP-Caches der benachbarten Rechner aktualisiert.

**Tipp**

Wenn Sie den KeepAlived-Daemon einsetzen, versendet er selbstständig den gratuitous ARP.

Wenn der Fail-Over zu Ihrer Zufriedenheit manuell funktioniert, können Sie eine Software installieren, mit der Sie die Gesundheit des Firewall-Clusters überwachen und bei einem erforderlichen Fail-Over das Skript für die Beförderung des Slaves zum Master aufrufen. Wenn Sie das mit KeepAlived erledigen möchten, achten Sie darauf, dass beim Einsatz des Layer-2-Drops der KeepAlived seine Nachrichten über das Synchronisationsnetzwerk versenden muss. In dem oben gewählten Szenario ist dies die Netzwerkkarte `eth2`. Um dies zu erreichen, müssen Sie in beiden Instanzen die folgende Zeile einfügen:

```
lvs_sync_daemon_interface eth2
```

Damit der KeepAlived-Daemon auch den Slave zum Master befördert, tragen Sie das Skript ebenfalls in der Konfiguration ein. Die komplette Konfiguration sieht dann für den initialen Master so aus:

```
vrp_sync_group VG1 {
  group {
    eth0
    eth1
  }
  notify_master /etc/keepalived/script_master.sh
  notify_backup /etc/keepalived/script_slave.sh
}
! Interne virtuelle IP-Adresse
vrp_instance VI_1 {
  state MASTER
  interface eth0
  lvs_sync_daemon_interface eth2
  virtual_router_id 1
  priority 100
  authentication {
    auth_type AH
    auth_pass password
  }
  virtual_ipaddress {
    192.168.1.1/24 brd 192.168.1.255 dev eth0
  }
}
```

```
}  
! Externe virtuelle IP-Adresse  
vrrp_instance VE_1 {  
  state MASTER  
  interface eth1  
  lvs_sync_daemon_interface eth2  
  virtual_router_id 2  
  priority 100  
  authentication {  
    auth_type AH  
    auth_pass password  
  }  
  virtual_ipaddress {  
    192.168.2.1/24 brd 192.168.1.255 dev eth1  
  }  
}
```

Das Skript `script_slave.sh` ist für die Funktion nicht zwingend erforderlich, da `ct_sync` es nicht erlaubt, einen aktiven Master über die `/proc`-Schnittstelle zu degradieren. Hier wird es eingesetzt, um eine Protokollierung des Vorgangs durchzuführen:

*Listing 26.2: Das Skript `script_slave.sh` protokolliert die Degradierung.*

```
#!/bin/sh  
/usr/bin/logger -p kernel.crit "Master zum Slave degradiert."
```

Achten Sie darauf, dass beim Einsatz von KeepAlived nun wieder eine virtuelle IP-Adresse für die Firewall verwendet wird. Die Netzwerkkarten müssen nun mit anderen IP-Adressen oder ohne IP-Adresse konfiguriert und aktiviert werden! Die Netzwerkkarte für das Synchronisationsnetzwerk benötigt natürlich eine IP-Adresse und muss auch die anderen Knoten erreichen können.

#### Hinweis



Ich habe bereits einige dieser Firewall-Cluster implementiert, die sich auch jetzt noch im täglichen Einsatz befinden. Außerdem kenne ich Firewall-Cluster, die mehrere 100 Mbit/s Verkehr abwickeln. Wenn Sie auf die Conntrack- und NAT-Helfermodule, die von der Synchronisation noch nicht unterstützt werden, verzichten können, steht Ihnen mit `ct_sync` ein mächtiger Firewall-Cluster zur Verfügung.