Автор: Administrator
04.01.2010 20:47 - Обновлено 28.05.2010 13:42

Построение виртуального свитча из нескольких сетевых карт с использованием Netgraph.

Итак
Виртуальный свич из нескольких сетевух на нетграфе.
Ситуация такая:
на сервере интерфейс rl0 смотрит наружу, в локалку провайдера 10.*.*.*.
а vr0 и vr1 смотрят во внутреннюю сеть 192.168.0.*
Для удобства хотелось бы чтобы в домашнюю сеть смотрела одна сетевуха, а все компы домашней сети были подключены к серверу через свич. Реализуем задуманное на основе ядерного модуля ng_bridge!

Кому лень читать статью полностью, выкладываю свой скрипт. Скрипт предоставляется "As Is" - что то тут лишнее, что то кривое... Конструктивная критика приветствуется. Наличие необходимых зависимостей подразумевается. #!/bin/sh

```
ngCtl="/usr/sbin/ngctl "
Sleep="/bin/sleep"
PFcmd="/sbin/pfctl"
Head="/usr/bin/head"
Tail="/usr/bin/tail"
Echo="/bin/echo"
Rm="/bin/rm"

PidFile="/var/run/mybridge.pid"

if1="vr0"
if2="vr1"
ifng="ngeth0"
Switch="switch"

case $1 in
 start)
 if [ -s $PidFile ]; then
 ${Echo} Bridge already runned!
 exit 1
 fi
 ${Echo} Starting bridge
 ${ngCtl} debug     2
 ${ngCtl} mkpeer    $if1:          bridge       lower   link0
 ${ngCtl} name      $if1:lower      $Switch
```

```
 ${ngCtl} connect   $if1:          $Switch:     upper   link1
 ${ngCtl} connect   $if2:          $Switch:     lower   link2
 ${ngCtl} connect   $if2:          $Switch:     upper   link3
 ${ngCtl} msg       $if1:          setpromisc   1
 ${ngCtl} msg       $if2:          setpromisc   1
 ${ngCtl} msg       $if1:          setautosrc   0
 ${ngCtl} msg       $if2:          setautosrc   0
 ${ngCtl} mkpeer    $Switch:       eiface       link5   ether
 ${ngCtl} name      $Switch:link5   $ifng
 ${Sleep} 2
 ${Head} -n7 /etc/pf.conf.bak    >  /etc/pf.conf
 ${Echo} 'int_if="'$ifng'"     >>  /etc/pf.conf
 ${Tail} -n79 /etc/pf.conf.bak  >>  /etc/pf.conf

 ifconfig $if1  -alias
 ifconfig $ifng inet 192.168.0.1 netmask 255.255.255.0

 ${PFcmd} -f /etc/pf.conf
 ${Echo} "runned" >> $PidFile

 ;;
 stop)
 if [ ! -s $PidFile ]; then
 ${Echo} Bridge is not runned!
 exit 1
 fi
 ${ngCtl} shutdown $Switch:
 ${ngCtl} shutdown $ifng:
 ${Head} -n7 /etc/pf.conf.bak > /etc/pf.conf
 ${Echo} 'int_if="'$if1'" >> /etc/pf.conf
 ${Tail} -n79 /etc/pf.conf.bak >> /etc/pf.conf
 ifconfig $if1 inet 192.168.0.1 netmask 255.255.255.0

 ${Rm} $PidFile
 ${PFcmd} -f /etc/pf.conf
 ;;
 *)
 if [ -s $PidFile ]; then
 ${Echo} State: Up
 else
 ${Echo} State: Down
 fi
 ;;
esac
```

Нетграф по слухам чертовски производителен и чертовски плохо документирован, особенно на русском, так что если что не так, не обессудьте - в описанной конфигурации у меня все работает уже не один месяц. Также оговорюсь, что все беру с рабочей системы, так что-то может быть лишнее, а чего-то может не хватать.

Начнем с необходимого - с нетграфа. Вот список того, что стоит у меня:# kldstat -v |grep ng

```
 202 ng_socket
 181 ng_car
 201 ng_rfc1490
 200 ng_pptpgre
 177 ng_async
 199 ng_ppp
 180 ng_bridge
 198 ng_one2many
 197 ng_nat
 176 ng_UI
 196 ng_mppc
 195 ng_lmi
 179 ng_bpf
 194 ng_l2tp
 175 ng_netflow
 193 ng_ksocket
 192 ng_ipfw
 191 ng_ip_input
 190 ng_iface
 189 ng_hole
 188 ng_gif_demux
 187 ng_gif
 186 ng_framerelay
 185 ng_ether
 184 ng_echo
 183 ng_deflate
 207 ng_vjc
 206 ng_tty
 205 ng_tee
 204 ng_tcpmss
 203 ng_ng_split
 182 ng_cisco
 2    1 0xc0945000 2fd4     ng_eiface.ko
 1 ng_eiface
# kldstat -v |grep netgraph
 178 netgraph
```

Как видно из всего этого необходим минимум ng_eiface, который я забыл(или поленился)

положить в ядро. Кроме того нам понадобятся модули ng_ether и ng_bridge. Но никто не запрещает побаловаться с ng_tee, ng_ipfw и чем-нибудь подобным для подсчета трафика, шейпинга et cetera...

После загрузки модулей нужных модулей смотрим что мы имеем# ngctl list
There are 6 total nodes:
Name: <unnamed> Type: eiface ID: 0000001e Num hooks: 0
Name: rl0 Type: ether ID: 00000001 Num hooks: 0
Name: vr0 Type: ether ID: 00000002 Num hooks: 0
Name: vr1 Type: ether ID: 00000003 Num hooks: 0
Name: ngeth0 Type: ether ID: 00000003 Num hooks: 0
Name: ngctl38473 Type: socket ID: 0000be26 Num hooks: 0

Мы должны видеть узлы типа ether по числу реальных интерфейсов, виртуальный интерфейс ngeth0 и соответствующий ему и узел socket, который служит для связи ngctl с ядром(это уже мои измышления. Подробнее - man ng_socket && man ngctl). Еще мы видем один безымянный узел - это родственник ngeth0, они создаются и уничтожаются только вместе.

Далее нам надо сделать мост. Но просто так его нельзя создать, его надо на что то прицепить. Прицепим хук link0 к хуку lower сетевухи vr0. Сказано - сделано. Попутно обзовем его каким нибудь нехорошим именем, типа switch# ngctl mkpeer vr0: bridge lower link0
ngctl name vr0:lower switch

Вот тут надо остановиться и поподробнее рассказать о том, что такое хуки(hooks), узлы(nodes), какие они бывают и с чем их едят. Нетграф - это система, которая строит граф, по которому бегают данные. В графе есть узлы и ребра. Иногда узел может просто висеть в воздухе(как например узлы наших сетевух), но чаще чтобы добавить какой то узел надо его к чему ни будь прикрутить. Прикручивание происходит соединением хуков разных узлов. После соединения между хуками появляется ребро по которому могут бегать данные от одного узла к другому. Хуки бывают разные и у каждого типа узла они свои. Какие то пропускают пакеты только в одну сторону, какие то пускают только пакеты верхних сетевых протоколов, подробнее про типы узлов написано в манах, а пока про насущное.
У сетевух(узлов типа ether) есть три хука - upper, lower, и orphans. Грубо говоря, lower работает с нижними протоколами(ethernet), upper - с верхними, а про orphans не помню, но он мне не вроде не подошел. У bridge - до NG_BRIDGE_MAX_LINKS(у меня на 7.2 в src/sys/netgraph/ng_bridge.h:55 говорится про 32 хука) хуков с именами типа link0 link1 link2... У eiface только один хук, - ether - через который бегают все пакеты.
При обращении к локальным узлам пишем двоеточие в конце имени. Обращаться можно как по имени, так и по ID. в последнем случае вместо <node_name>: пишется [0x<node_id>]:

С теорией вроде закончил, перейдем к практике. Продолжаем строит свич: цепляем к мосту остальные хуки сетевушек# ngctl connect vr0: switch: upper link1
ngctl connect vr1: switch: lower link2
ngctl connect vr1: switch: upper link3

Идем далее. Сетевухи имеют свойства выпуская пакет прописывать в нем поле отправитель себя и игнорировать пакеты, предназначенные не ей. Отучаем их от этих нехороших привычек:# ngctl msg vr0: setpromisc 1
ngctl msg vr1: setpromisc 1
ngctl msg vr0: setautosrc 0
ngctl msg vr1: setautosrc 0

Свич почти готов. цепляем к нему виртуальную сетевуху, чтобы общаться с ним
ngctl mkpeer switch: eiface link5 ether
ngctl name switch:link5 ngeth0

И далее самое интересное. снимаем ипы с реальных сетевух(порты свича не имеют ипов) и ставим внутренний ип(у меня это 192.168.0.1) машины на виртуальную сетевуху. Реальные карточки должны быть без ипов только в состоянии UP# ifconfig vr0 up
ifconfig vr1 up
ifconfig vr0 -alias
ifconfig vr1 -alias
ifconfig ngeth0 inet 192.168.0.1 netmask 255.255.255.0

Вроде бы и все.
Смотрим что у нас получилось:# ifconfig
rl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=8<VLAN_MTU>
ether <XXX>
inet 10.XXX.XXX.XXX netmask 0xffffXXX broadcast 10.XXX.XXX.XXX
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
vr0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST>
metric 0 mtu 1500
options=2808<VLAN_MTU,WOL_UCAST,WOL_MAGIC>
ether <XXX>
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
vr1: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST>
metric 0 mtu 1500
options=2808<VLAN_MTU,WOL_UCAST,WOL_MAGIC>
ether <XXX>

```
 media: Ethernet autoselect (100baseTX <full-duplex>)
 status: active
ngeth0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
 ether 00:00:00:00:00:00
 inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
<overquoting deleted>
# ngctl list
There are 17 total nodes:
 Name: <unnamed>       Type: ksocket         ID: 0000be23   Num hooks: 1
 Name: <unnamed>       Type: pptpgre         ID: 0000be22   Num hooks: 2
 Name: <unnamed>       Type: eiface          ID: 0000001e   Num hooks: 1
 Name: ng0             Type: iface           ID: 0000bd94   Num hooks: 1
 Name: rl0             Type: ether           ID: 00000001   Num hooks: 0
 Name: switch          Type: bridge          ID: 00000014   Num hooks: 5
 Name: vr0             Type: ether           ID: 00000002   Num hooks: 2
 Name: vr1             Type: ether           ID: 00000003   Num hooks: 2
 Name: mpd16513-stats  Type: socket          ID: 0000bd9a   Num hooks: 0
 Name: ngctl38842      Type: socket          ID: 0000be2a   Num hooks: 0
 Name: mpd16513-cso    Type: socket          ID: 0000bd92   Num hooks: 0
 Name: mpd16513-eso    Type: socket          ID: 0000bd93   Num hooks: 0
 Name: mpd16513-lso    Type: socket          ID: 0000bd91   Num hooks: 1
 Name: ngeth0          Type: ether           ID: 0000001f   Num hooks: 0
 Name: mpd16513-B1-mss Type: tcpmss          ID: 0000be24   Num hooks: 2
 Name: mpd16513-B1     Type: ppp             ID: 0000bd95   Num hooks: 3
 Name: mpd16513-L1-lt  Type: tee             ID: 0000bd96   Num hooks: 2

# arp -a
? (10.XXX.XXX.XXX) at <XXX_MAC>  on rl0 [ethernet]
------------------------``---------------------------------
? (192.168.0.2) at <XXX_MAC> on ngeth0 [ethernet]
? (192.168.0.3) at <XXX_MAC> on ngeth0 [ethernet]
? (192.168.0.255) at ff:ff:ff:ff:ff:ff on ngeth0 permanent [ethernet]
```

Примечание: записи вида ? (10.XXX.XXX.XXX) at <XXX_MAC> on rl0 [ethernet] - адреса из локалки провайдера)

Как видно, сетевухи vrX смирились со своей ролью в качестве портов свича и на них не висят маки соседих машин. Все выглядит так, будто у нас есть свич, в который воткнуты кабеля машин внутренней сети и кабель от нашего компа с интерфейса ngeth0(любопытно, что ifconfig и arp по разному определяют его мак). самого свича, как и полагается, не видно

Дальше разруливаем все пакеты файером, в нем разрешаем прохождение пакетов через реальные внутренние сетевухи(или шейпим/режем по надобности), ставим нат с внешней сетевухи на ngeth0, раздача инета между сетевухами моста будет

регулироваться нетграфом. В общем ведем себя так, как будто у нас появился свич и исчезли две сетевухи.