

Photographing Wildlife

How digital cameras came to the rescue of Zebra routers

Remco van Mook
Virtu Secure Webservices
remco@virtu.nl

Achtergrond

In 2000 moest binnen mijn bedrijf een router aangeschaft worden. Het bedrijf was net gestart, had een stevig groeiscenario voor ogen en (wat achteraf gezien ongebruikelijk bleek) had de hand redelijk op de knip. Investeren was geen probleem, maar dan wel investeren in zaken die bij doorgroeien niet versneld uitgefaseerd zouden moeten worden.

Kort gezegd: wat voor router moeten we kopen? Een die onze geprognosticeerde groei over een periode van 3 jaar zou kunnen bijhouden? Dat was nogal een aanslag op de kas; zeker omdat het apparaat gedurende het grootste deel van zijn loopbaan overgedimensioneerd zou zijn. Een router die nu op maat is dan? Ook die was bepaald niet goedkoop en zou waarschijnlijk binnen afzienbare tijd toch vervangen moeten worden. Een apparaat dat kon meegroeien leek niet voorhanden.

Het alternatief in een kennisintensief bedrijf: we bouwen er zelf een. Gedegen kennis van zowel hardware als software als protocollen was aanwezig; de kosten leken mee te vallen. Aldus werden 2 (N+1) systemen aangeschaft, volgestopt met netwerkkaarten, Debian Linux en voorzien van routing software (Zebra).

Probleemstelling

Naast de routers bleek het vrij simpel om andere netwerkgerelateerde zaken op een dergelijke manier op te lossen. Een firewall voor onze klanten, een IDS, nog een router. Allemaal machines waar een harddisk in zat die warm werd, storingen kon veroorzaken en vooral eigenlijk weinig tot niets toe te voegen had aan het uitvoeren van de kerntaak van het systeem. Allemaal machines waar in feite een full-blown OS op stond dat weinig tot niets toe te voegen had aan het functioneren van het systeem. Een OS dat beheerd moet worden, filesystems die kapot kunnen gaan bij een crash. Het was duidelijk dat de medaille van goedkope systemen voor netwerk taken een keerzijde had: een uit de hand lopende beheersinspanning. Met dit probleem onder de arm gingen we aan de slag.

Werken aan een oplossing

De harddisk

Allereerst: de alternatieven voor een harddisk. Booten vanaf een server op een netwerk kan, maar is niet direct wenselijk voor een router (die dat netwerk misschien wel in de lucht houdt). Booten vanaf floppy levert een misschien wel te minimale omgeving op om praktisch mee te kunnen werken; bovendien staan floppies niet bekend om hun betrouwbaarheid ten opzichte van harddisks. Booten vanaf CD zou kunnen, maar levert een wel heel statische configuratie op. Solid state disks (flash) zouden de beste oplossing zijn, maar bleken naast duur ook nog beperkt leverbaar te zijn.

De oplossing kwam toen iemand zijn nieuwe digitale camera kwam laten zien, met een Compact Flash kaartje erin om de foto's op te slaan. Het kaartje werd uit het toestel verwijderd, in een PCMCIA-adapter geschoven en in een laptop herkend als IDE-disk. Interessant.



Compact Flash blijkt, na wat korte studie, zich gewoon aan te laten sluiten en te gedragen als een IDE disk met een FAT filesystem. Verder bleken er meerdere leveranciers van adapters te bestaan waarmee Compact Flash kaartjes zich direct lieten aansluiten op de IDE bus van een PC.

Compact Flash heeft echter een aantal belangrijke nadelen: snelheid en levensduur. De Compact Flash die wij gebruiken heeft een leesnelheid van ongeveer 1 MB/sec., niet bepaald snel. Schrijfsnelheid ligt nog een factor 2 lager; hooguit 500K/sec. Daarnaast heeft flash geheugen een beperkte levensduur: na een paar duizend keer schrijven (hoeveel precies is niet duidelijk, conservatief is 1000, 10000 is meer reëel), raken er delen defect. Al met al is een Compact Flash disk niet de optimale plek om een 'live' filesystem neer te zetten. Wat natuurlijk wel kan is Compact Flash gebruiken om een kernel en een filesystem image als RAM-disk te laden, en dat is de manier die we gekozen hebben.



Het Operating system

Vervolgens werd de aandacht verlegd naar het operating system. Het eisenpakket was niet eenvoudig; men was zodanig verwend met de complete UNIX-omgeving op de systemen dat op functionaliteit niet al te veel ingeleverd mocht worden. Het was dus taak om de bestaande omgeving 'uit te kleden'. In eerste instantie werd gesnoeid in de beschikbare packages: geen maildaemon meer, geen compilers, text formatting tools. Wat aan daemons overbleef was in feite ssh, ntp, zebra (routing) en syslog. Daarna werden zelden tot nooit gebruikte gegevens onder handen genomen: termcaps, locales, timezones. Voor het verwijderen (en blijvend verwijderd houden!) van locales heeft Debian Linux het package `localepurge`, wat hier dankbaar voor werd gebruikt. Daarna werd alle documentatie verwijderd. Geïnspireerd door het `localepurge` package heb ik hiervoor het package 'docspurge'¹ gemaakt: alle info, manpages, documentatie van packages et cetera wordt hierdoor verwijderd.

Ten slotte, en dat deed pijn, moest `apt` (de package manager) er aan geloven. Met de geminimaliseerde set bleek dat de databestanden van de package manager inmiddels goed waren voor een kleine 25% van de totale benodigde diskruimte! Om het leed te verzachten is de 'lokale' `apt` vervangen door een perlscript dat de package manager van een ander systeem leent².

Inmiddels is het de inhoud van het complete filesystem nog slechts een kleine 30MB. Om nog wat bewegingsruimte te hebben op het filesystem is dit 48MB groot. Na comprimeren is dit nog een RAM-disk image van ongeveer 10MB.

Rest nog de vraag, wat te doen met het filesystem. Is het belangrijker om altijd een werkend filesystem te hebben of is het belangrijker om altijd alle configuratiewijzigingen bij te houden? Bij het beantwoorden van die vraag is met een schuin oog gekeken naar 'echte' routers. Die hanteren doorgaans 2 configuraties: een *running-config* en een *startup-config*. Een van de voordelen die je hierbij hebt is dat de wijzigingen die worden doorgevoerd in de *running-config* (de op dat moment draaiende configuratie) geen invloed hebben op de configuratie die geladen wordt als de machine op dat moment opnieuw zou starten (de *startup-config*). Met een commando kunnen de 2 gesynchroniseerd worden (bijvoorbeeld als de *startup-config* vervangen moet worden door de huidige *running-config*, omgekeerd kan natuurlijk ook).

Door die analogie door te trekken komt het er op neer dat we op zoek zijn naar 2 keer alle configuratiebestanden op een Debian Linux systeem; 2 keer de `/etc` directory dus. Wijzigingen horen zich niet buiten de `/etc` directory voor te doen, dus wijzigingen daarbuiten mogen, in principe, verloren gaan. Al met al zijn er dus een aantal plaatsen waar een configuratie kan staan:

1. De filesystem image zelf (zeg maar de 'default' configuratie).
2. De *startup-config* (een tar.gz file met daarin een `/etc` tree).

3. De *running-config* (de /etc directory op het 'live' filesystem).
4. De *old-config* (een tar.gz met daarin de vorige startup-config).

Om met deze configuraties goed om te kunnen gaan zijn er een aantal zaken aangepast:³

1. Aan de kernel kan een optie meegegeven worden waardoor de *startup-config* niet geladen wordt bij het starten van het systeem.
2. Als allereerste na het starten van de kernel en init wordt de *startup-config* over de /etc directory op de RAM-disk geschreven (tenzij punt 1, natuurlijk).
3. Er is een shellsript `commit` dat de *running-config* kopieert naar de *startup-config* en de oude *startup-config* verplaatst naar *old-config*.
4. Er is een shellsript `rollback` dat de *startup-config* vervangt door de *old-config*.
5. Er is een shellsript `reconfigure` dat de *running-config* vervangt door de *startup-config* (in feite hetzelfde als bij 2. gebeurt).

De Kernel

Omdat de kernel buiten de filesystem image een eigen plaats krijgt op de Compact Flash, is uit overweging van beheer ervoor gekozen om een monolithic kernel, dus zonder modules of ondersteuning voor modules te maken. We gebruiken zelf een aantal standaardcomponenten, dus het aantal verschillende drivers wat in de kernel moet komen is beperkt. Wat wel belangrijk is, is dat de kernel `initrd` (RAM-disks bij het booten) ondersteunt en dat de standaard maat van een RAM-disk groot genoeg is om het filesystem image in te kunnen passen (standaard is 8MB, we gebruiken 48MB).

Geen swap

Omdat effectief onze enige disk een RAM-disk is, is het er op na houden van swap een beetje zinloos. Bovendien zijn de applicaties die nog wel gebruikt worden dermate kritisch dat ze niet uitgeswapt kunnen worden; een van de dingen die we geleerd hebben van het draaien met harddisk (en dus wel met swap) is dat bijvoorbeeld BGP sessies kapot gaan als de `bgp` daemon door geheugentekort gaat swappen. Bovendien is geheugen momenteel goedkoop, 'overkill' schaadt hier nadrukkelijk niet.

Putting it all together

En uiteindelijk worden alle onderdelen bij elkaar gestopt. De Compact Flash verdwijnt zoals hiernaast gefotografeerd staat in een systeem, wordt uitgerust met een bootloader ('Grub' in dit geval) en voorzien van een kernel en een filesystem image. De Compact Flash bevat nu het volgende:

<code>/boot/</code>	<i>directory voor Grub, bootloaders, configuratie bootloader</i>
<code>/kernel-2.4.19-static</code>	<i>monolithic kernel</i>
<code>/rootimg.gz</code>	<i>compressed ramdisk image</i>

De harddisk is uit het zicht verdwenen, het operating system is ingekrompen.



We kunnen het zoeken naar een andere oplossing voorlopig weer even opschuiven.

¹ http://slashme.org/flashrouters/docspurge_0.0.1_all.deb

² <http://slashme.org/flashrouters/perl-apt/>

³ Alle scripts zijn beschikbaar op <http://slashme.org/flashrouters/scripts/>