Simulation von Netzwerken in virtualisierten Umgebungen

Jörg Möllenkamp
Oracle B.V.& Co. KG
Hamburg

Schlüsselworte:

Virtualisierung,Netzwerke,Konsolidierung

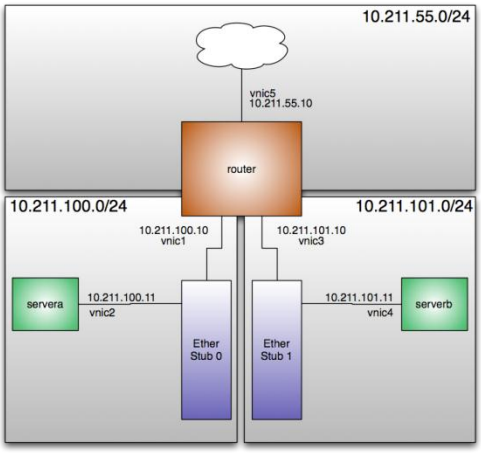
Einleitung

Virtualisiert man mehr als nur einige wenige lose zusammenhängende Systeme, so ergibt sich schnell die Frage, wie man die Infrastruktur, die die Systeme zusammenhält, virtualisiert. Insbesondere das IP-Netzwerk sticht hier hervor. Nicht immer macht es die bestehende Installation dem Planer einfach, mal verstecken sich in einer Umgebung fest einkodierte IP-Adressen, die sich nicht so einfach entfernen lassen, mal geben architekturelle Vorgaben eine Netzwerkgestaltung vor, die nicht einfach nur durch das Hinzufügen eines virtualisierten Switches ausgestaltet werden kann.

Im Rahmen dieses Artikels möchte ich auf die besonderen Gestaltungsmöglichkeiten, die Solaris 11 Express bietet, hinweisen. Obschon die darin enthaltenen Solaris Container das Betriebssystem und nicht die Hardware virtualisieren, vermögen die darin enthaltenen Mechanismen auch komplexe Netzwerke zu simulieren. Weiterhin soll dieser Artikel auch zeigen, wie viele Features von Solaris nahtlos ineinandergreifen.

Aufbau eines Netzwerkes mittlerer Komplexität in Solaris 11 Express

Nehmen wir an, es soll ein etwas komplizierteres Netzwerk in einer virtualisierten Umgebung dargestellt werden. Natürlich macht es zunächst Sinn sich über die Struktur einen Überblick zu verschaffen und sich die Umgebung bildlich zu vergegenwärtigen.



Daraus ergibt sich eine Reihe von Systembestandteilen:

- `simulation` - der Server, der das Netzwerk simulieren soll (er ist nicht in der obigen Graphik berücksichtigt)

- `router` – eine Netzwerkkomponente, die zwischen den Netzwerken routen soll.
- `Servera, serverb` – Server im Rahmen des simulierten Netzwerkes.
- Zwei Switche

Wie man aus Infrastrukturübersicht ersehen kann, benötigen wir zunächst einmal die Switche. Die virtuellen Switche in der Netzwerkvirtualisierung von Solaris werden Etherstubs genannt.

```
root@simulation:~# dladm create-etherstub etherstub0
root@simulation:~# dladm create-etherstub etherstub1
```

Nun muss es natürlich Ethernetkarten geben, die ebenso virtuell wie die Switche sind, um sie später in den virtuellen Instanzen des Betriebssystems zu nutzen. Diese müssen wiederum mit den Switches verbunden werden.

```
root@simulation:~# dladm create-vnic -l etherstub0 vnic1
root@simulation:~# dladm create-vnic -l etherstub0 vnic2
```

Mit diesen beiden Befehlen werden zwei diese virtuellen Ethernetkarten konfiguriert. In der Sprache von Solaris heißen diese „virtual NICs“. Die gebräuchliche Abkürzung ist VNICs. Was passiert hierbei zum Beispiel bei der ersten Kommandozeile: Es wird ein virtueller NIC `vnic1` konfiguriert, dessen virtuelles Kabel in den virtuellen Switch `etherstub0` gesteckt wird

Für den anderen Switch müssen wir natürlich entsprechendes durchführen:

```
root@simulation:~# dladm create-vnic -l etherstub1 vnic3
root@simulation:~# dladm create-vnic -l etherstub1 vnic4
```

Wie in der obigen Graphik zu sehen, muss nun auch noch das Interface angelegt werden, mit dem das emulierte Netzwerk an die reale angeschlossen wird. Hierzu nutzen binden wir dieses Netzwerk nicht an einen emulierten Switch, sondern direkt an eine Netzwerkkarte.

```
root@simulation:~# dladm create-vnic –l e1000g0 vnic5
```

Überprüfen kann man die Konfiguration wie folgt. Hierbei stehen die virtuellen NICs gleichberechtigt neben den physikalisch vorhandenen Karten.

```
root@simulation:~# dladm show-link
LINK        CLASS     MTU    STATE    BRIDGE     OVER
e1000g0     phys      1500   up       --         --
etherstub0  etherstub 9000   unknown  --         --
etherstub1  etherstub 9000   unknown  --         --
vnic1       vnic      9000   up       --         etherstub0
vnic2       vnic      9000   up       --         etherstub0
vnic3       vnic      9000   up       --         etherstub1
vnic4       vnic      9000   up       --         etherstub1
vnic5       vnic      1500   up       --         e1000g0
```

Die Konfiguration des virtuellen Netzwerkes ist damit abgeschlossen. Mehr ist hier nicht zu tun. Allerdings ist ein Netzwerk ohne Server sinnlos und so möchte ich im Folgenden darstellen, wie man diese Server als Solaris Zonen bereitstellen kann.

Die ersten Schritte sind nicht wirklich notwendig, um die virtuellen Instanzen bereitzustellen, allerdings beschleunigen sie nachher die Bereitstellung wesentlich. Wir nutzen hierbei das Cloning-Feature der Solaris Zonen. Wir kreieren also zunächst einmal eine Schablone, dstonie als Vorlage für unsere spätere Zone dient. Die eigentlich später benutzen Zonen sind dann nur noch Kopien dieser Vorlage, oder genau genommen Klone.

Legt man eine Zone in ZFS-Filesystem, kann Solaris später einfach einen Zustand der Schablone einfrieren und an einer anderen Stelle beschreibbar machen, ohne dass das Original verändert wird. Statt jede Datei zu kopieren müssen nur einige Blöcke geschrieben werden, um die Verwaltungsinformationen des Clones auf die Platte zu schreiben. Das geht naturgemäß sehr viel schneller als eine volle Kopie. Die eigentlichen Datenblöcke sind dieselben. Eigene Datenblöcke erhält der Klon erst, wenn sich ein Block ändert und dann auch nur für die veränderten Blöcke.

```
root@simulation:~# zfs create rpool/zones
root@simulation:~# zfs set mountpoint=/zones rpool/zones
```

Damit habe ich ein Filesystem angelegt, in das nun die Zonen gelegt werden. Nun kann ich die Zone konfigurieren und installieren. Hilfsweise sollte man sich hier eine Steuerdatei anlegen, in der die entsprechenden Kommandos stehen:

```
/root/template:
create -b
set zonepath=/zones/template
set ip-type=exclusive
set autoboot=false
commit
```

Die Datei `/root/template` nutze ich nun um die Konfiguration der Templatezone zu steuern. Durch das –f wird hierbei das File template gelesen. Die darin enthaltenen Befehle steuern dann den eigentlichen Konfigurationsvorgang.

```
root@simulation:~# zonecfg -z template -f /root/template
```

Die Konfiguration ist durchgeführt, die Zone muss nun installiert werden. Dieser Vorgang kann in Abhängigkeit vom Plattensubsystem, auf denen die Zonen liegen und der Geschwindigkeit mit der auf ein Solaris-Paketrepository zugegriffen werden kann, durchaus den Genuss einer oder zwei Tassen Kaffee zulassen.

```
root@simulation:~# zoneadm -z template install
A ZFS file system has been created for this zone.
   Publisher: Using solaris (http://pkg.oracle.com/solaris/release/ ).
       Image: Preparing at /zones/template/root.
Sanity Check: Looking for 'entire' incorporation.
  Installing: Core System (output follows)
------------------------------------------------------------
Package:                     pkg://solaris/consolidation/osnet/osnet-incorporation@0.5.11,5.11-
0.151.0.1:20101104T230646Z
License: usr/src/pkg/license_files/lic_OTN

Oracle Technology Network Developer License Agreement
[…]
Oracle may contact you to ask if you had a satisfactory experience
installing and using this OTN software download.



              Packages to install:     1
           Create boot environment:    No
DOWNLOAD                                   PKGS       FILES    XFER (MB)
Completed                                   1/1         1/1      0.0/0.0

PHASE                                        ACTIONS
Install Phase                                  11/11

PHASE                                        ITEMS
Package State Update Phase                     1/1
```

```
Image State Update Phase                         2/2
              Packages to install:    45
           Create boot environment:    No
              Services to restart:     3
DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                                45/45 12511/12511    89.1/89.1

PHASE                                        ACTIONS
Install Phase                            17953/17953

PHASE                                          ITEMS
Package State Update Phase                     45/45
Image State Update Phase                         2/2
  Installing: Additional Packages (output follows)
              Packages to install:    46
           Create boot environment:    No
              Services to restart:     2
DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                                46/46   4498/4498    26.5/26.5

PHASE                                        ACTIONS
Install Phase                            6139/6139

PHASE                                          ITEMS
Package State Update Phase                     46/46
Image State Update Phase                         2/2

        Note: Man pages can be obtained by installing SUNWman
 Postinstall: Copying SMF seed repository ... done.
 Postinstall: Applying workarounds.
        Done: Installation completed in 395.966 seconds.

  Next Steps: Boot the zone, then log into the zone console (zlogin -C)
              to complete the configuration process.
```

Diese Zone wird nicht gestartet, sondern dient nur als Vorlage für alle weiteren Zonen. Im Rahmen der Installation wird automatisch eine Reihe von ZFS-Filesystemen angelegt.

```
root@simulation:~# zfs list
NAME                            USED  AVAIL  REFER  MOUNTPOINT
[..]
rpool/zones/template            341M  10.2G    32K  /zones/template
rpool/zones/template/ROOT       341M  10.2G    31K  legacy
rpool/zones/template/ROOT/zbe   341M  10.2G   335M  legacy
```

Für die Simulation des Netzwerkes mit drei Servern benötigen wir drei Zonen, dementsprechend benötigen wir auch drei Konfigurationsfiles:

/root/servera:
```
create -b
set zonepath=/zones/serverA
set ip-type=exclusive
set autoboot=false
add net
set physical=vnic2
end
commit
```

/root/serverb:
```
create -b
set zonepath=/zones/serverB
set ip-type=exclusive
set autoboot=false
add net
set physical=vnic4
end
commit
```

/root/router:
```
create -b
set zonepath=/zones/router
set ip-type=exclusive
set autoboot=false
add net
set physical=vnic5
end
add net
set physical=vnic1
end
add net
set physical=vnic3
end
commit
```

Damit man die Basis-Konfiguration der einzelnen Zonen nach dem Booten nicht mehr manuell durchführen muss, kann man sich des sysidcfg-Mechanismus bedienen. Liegt beim ersten Booten eines Systems oder eben einer Zone ein solches File an einer bestimmten Stelle im Filesystem der Zone, so werden die darin erhaltenen Daten verwendet um beispielsweise das Netzwerk zu konfigurieren oder um ein root-Passwort[1] zu vergeben.

/root/sysidcfg_router:
```
locale=C
terminal=vt100
name_service=none
network_interface=vnic5 {primary hostname=router ip_address=10.211.55.10 netmask=255.255.255.0
protocol_ipv6=no default_route=10.211.55.1}
network_interface=vnic1   {hostname=router-a   ip_address=10.211.100.10   netmask=255.255.255.0
protocol_ipv6=no default_route=NONE}
network_interface=vnic3   {hostname=router-b   ip_address=10.211.101.10   netmask=255.255.255.0
protocol_ipv6=no default_route=NONE}
nfs4_domain=dynamic
root_password=cmuL.HSJtwJ.I
security_policy=none
timeserver=localhost
timezone=US/Central
```

/root/sysidcfg_servera:
```
system_locale=C
terminal=vt100
name_service=none
network_interface=vnic2   {hostname=servera   ip_address=10.211.100.11   netmask=255.255.255.0
protocol_ipv6=no default_route=NONE}
nfs4_domain=dynamic
root_password=cmuL.HSJtwJ.I
security_policy=none
timeserver=localhost
timezone=US/Central
```

/root/sysidcfg_serverb:
```
system_locale=C
terminal=vt100
name_service=none
network_interface=vnic4   {hostname=serverb   ip_address=10.211.101.11   netmask=255.255.255.0
protocol_ipv6=no default_route=NONE}
nfs4_domain=dynamic
root_password=cmuL.HSJtwJ.I
security_policy=none
timeserver=localhost
timezone=US/Central
```

An dieser Stelle können wir nun die vorbereiteten Files nutzen, um die drei Virtuellen Systeme zu kloniert, zu konfigurieren und zu starten. Vorher möchte ich aber noch auf die Besonderheit hinweisen, dass die Konfiguration explizit keine Defaultroute enthält. Bitte halten Sie das für einen Moment im Hinterkopf.

```
root@simulation:~# zonecfg -z router -f router
root@simulation:~# zoneadm -z router clone template
root@simulation:~# cp /root/sysidcfg_router /zones/router/root/etc/sysidcfg
root@simulation:~# one
```

Die Zone wird nun nicht mehr installiert, sondern das ZFS snapshot/clone-Feature genutzt, um eine Klon der Zone anzulegen. Wichtig ist hier zu wissen, dass jede Zone nur den Speicherplatz belegt, in dem sie sich vom Template unterscheidet. Des Weiteren geht der Klonierungsvorgang so schnell, das kaum der Gedanke an Kaffee gefasst werden kann.

```
root@simulation:~# zfs list
```

[1] Das root-Password für alle Zonen ist newroot

```
NAME                            USED  AVAIL  REFER  MOUNTPOINT
[..]
rpool/zones/router             5.89M  10.2G    34K  /zones/router
rpool/zones/router/ROOT        5.86M  10.2G    31K  legacy
rpool/zones/router/ROOT/zbe    5.83M  10.2G   340M  legacy
```

Die Klonierung für die beiden anderen Zonen führen wir auf die gleiche Art und Weise durch.

```
root@simulation:~# zonecfg -z servera -f servera
root@simulation:~# zoneadm -z servera clone template
root@simulation:~# cp /root/sysidcfg_servera /zones/serverA/root/etc/sysidcfg
root@simulation:~# zoneadm -z servera boot

root@simulation:~# zonecfg -z serverb -f serverb
root@simulation:~# zoneadm -z serverb clone template
root@simulation:~# cp /root/sysidcfg_serverb /zones/serverB/root/etc/sysidcfg
root@simulation:~# zoneadm -z serverb boot
```

Da wie bei unvirtualisierten Systemen nach dem ersten Booten eine Reihe von Schritten ablaufen (beispielsweise das Einlesen der Konfigurationsinformationen der Service Management Facility), dauert es einen Moment, bis alle Zonen gestartet sind. Jedoch kann man jetzt schon den Erfolg obiger Konfiguration testen.

```
root@simulation:/home/jmoekamp# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              ipkg     shared
   6 router           running    /zones/router                  ipkg     excl
   8 servera          running    /zones/serverA                 ipkg     excl
  10 serverb          running    /zones/serverB                 ipkg     excl
```

Ein erster Test besteht natürlich darin, in der globalen Zones des Servers, der die Zonen betreibt, zu versuchen, das Netzwerkinterface zu nutzen.

```
root@simulation:~# ifconfig vnic2 plumbroute
vnic2 is used by non-globalzone : server
```

Korrekterweise verweigert Solaris dies. Das Interface ist schließlich schon in Benutzung. Nun will man natürlich auch testen, wie sich die Umgebung hinsichtlich des simulierten Netzwerkes verhält. Allerdings muss man hier vorher der Router-Zone auch das Routen erlauben.

```
root@router:~# routeadm -e ipv4-forwarding
root@router:~# routeadm -e ipv4-routing
root@router:~# routeadm -u
root@router:~# routeadm
             Configuration   Current             Current
                     Option   Configuration        System State
---------------------------------------------------------------
               IPv4 routing   enabled              enabled
               IPv6 routing   disabled             disabled
             IPv4 forwarding   enabled              enabled
             IPv6 forwarding   disabled             disabled

           Routing services   "route:default ripng:default"

Routing daemons:

                      STATE   FMRI
                   disabled   svc:/network/routing/legacy-routing:ipv6
                   disabled   svc:/network/routing/legacy-routing:ipv4
                   disabled   svc:/network/routing/ripng:default
                     online   svc:/network/routing/route:default
                   disabled   svc:/network/routing/rdisc:default
                     online   svc:/network/routing/ndp:default
root@router:~#
```

Dieses Dokument umfasst nur einen kleinen Teil der Routingfähigkeiten von Solaris, wie man aber leicht anhand obiger Übersicht sehen kann, unterstützt Solaris 11 Express eine Reihe von Routingprotokollen. Viele der möglichen Protokolle sind in diesem System noch gar nicht installiert Diese zu behandeln würde aber den Rahmen dieses Vortrags vollkommen sprengen.

Meldet man sich auf dem Node `servera` an, so findet man folgende Routingtabelle an.

```
root@servera:~# netstat -nr

Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              10.211.100.10        UG        2          4 vnic2
10.211.100.0         10.211.100.11        U         3          1 vnic2
127.0.0.1            127.0.0.1            UH        2          4 lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2       0 lo0
```

Ich habe sie vorhin gebeten, im Hinterkopf zu halten, dass wir keine Defaultroute angegeben haben. Woher kommt aber nun die in der `netstat -nr` Ausgabe angezeigte Defaultroute?

Es handelt sich hierbei um eine dokumentierte, aber wenig bekannte Eigenschaft von Solaris. Wird keine Defaultroute angegeben, so wird in Abhängigkeit der Anzahl von Netzwerkkarten ein Mechanismus zur Findung einer Defaultroute gewählt. Ist nur eine Netzwerkkarte im System, so wird versucht mit RDISC[2] eine Defaultroute zu finden. Ist mehr als eine Karte verfügbar, wird versucht, diese Information über das RIP[3] zu erlangen.

Zurück zu unserer Umgebung: Melde ich mich auf `serverb` an, und benutze `traceroute` zur Ermittlung der Routingwege, so sehe ich, das hier tatsächlich über die router-Zone geroutet worden ist.

```
root@serverb:~# ping 10.211.100.11
10.211.100.11 is alive
root@serverb:~# root@serverb:~# traceroute 10.211.100.11
traceroute to 10.211.100.11 (10.211.100.11), 30 hops max, 40 byte packets
 1  10.211.101.10 (10.211.101.10)  0.138 ms  0.076 ms  0.039 ms
 2  10.211.100.11 (10.211.100.11)  0.077 ms  0.059 ms  0.041 ms
root@serverb:~#
```

Und was ist mit Fehlern im Netz?

Nun sind Netzwerke insbesondere wenn sie in den Weitbereich greifen selten immer Fehlerfrei. Es kann hier sehr interessant sein, zu testen, wie sich eine Applikation im Fehlerfalle verhält. Auch hier kann Solaris 11 Express ein sehr nützliches Werkzeug sein, auch wenn das hierzu verwendete Tool nicht offiziell Bestandteil eines von Oracle ausgelieferten Produkts ist. Auf der Webseite des Opensolaris-Projekts kann ein Treiber herunter geladen werden, der zur Emulation von WAN-Umgebungen genutzt werden können. Dieser Treiber fängt die Pakete, die über ihn laufen ab und führt eine Reihe von Manipulationen an diesen durch. Heruntergeladen werden kann dieser Treiber unter

[2] Router Discovery
[3] Router Identification Protocol

http://hub.opensolaris.org/bin/download/Community+Group+networking/WebHome/hxbt.tar.gz . Die Fehlerfälle, die dieser Treiber simulieren kann sind vielfältig:

- Neusortierung von Paketen
- Verwerfen von Paketen
- Verlängerung der Paketlatenz
- Bandbreiteneinschränkung[4]
- Paketcorruption

Ich möchte an einem Beispiel zeigen, wie dieser genutzt werden kann. Zunächst muss man den hxbt-Treiber an die richtige Stelle im System kopieren. Davon ausgehend, das man das .tar.gz-Archiv in das momentane Verzeichnis ausgepackt hat, muss man auf einem SPARC-System den Treiber wie folgt kopieren.

```
jmoekamp@server:~# cp ./hxbt/onnv/usr/src/uts/sparc/hxbt/obj64/hxbt /kernel/drv/sparcv9
```

Auf einem x86-basierten System sieht die Sequenz leicht anders aus:

```
jmoekamp@server:~# cp ./hxbt/onnv/usr/src/uts/intel/hxbt/obj32/hxbt /kernel/drv
jmoekamp@server:~# cp ./hxbt/onnv/usr/src/uts/intel/hxbt/obj64/hxbt /kernel/drv/amd64
```

Unabhängig von der Architektur muss immer die datei hxbt.conf an die selbe Stelle kopiert werden.

```
jmoekamp@server:~$ cp ./hxbt/onnv/usr/src/uts/common/inet/hxbt/hxbt.conf /kernel/drv
```

Nun muss das Modul geladen werden. Üblicherweise sind an ein Netzwerkinterface der ip-Treiber sowie der Treiber der Netzwerkkarte gebunden.

```
jmoekamp@server:~# ifconfig e1000g0 modlist
0 ip
1 e1000g
```

An die Stelle zwischen den beiden Treibern muiss nun der hxbt-Treiber eingefügt werden.

```
jmoekamp@server:~# ifconfig e1000g0 modinsert hxbt@1
```

Danach sollte man kurz kontrollieren, ob der Treiber wirklich geladen worden ist. Das geht mit einem kurzen `modinfo`-Befehl:

```
moekamp@server:~# modinfo
 Id         Loadaddr   Size Info Rev Module Name
  0 ffffffffb800000 1cb9fa   -   0  unix ()
[...]
260 fffffffff7aa5000   22c8   -   1  hxbt (hxbt stream module v1.1)
260 fffffffff7aa5000   22c8 292   1  hxbt (hxbt stream driver v1.1)
```

In einem ersten Test fügen wir eine künstliche Latenz von 100ms ein.

```
jmoekamp@server:~# ifhit 192.168.2.200 -l 100
(::ffff:192.168.2.200)
  Delay=100 ms
```

[4] Wobei dies durch die Bandbreitenlimitierung, die im aktuellen IP-Stack von Solaris 11 eingebaut ist, nicht mehr ganz so interessant ist.

Mit einem Ping kann man den Erfolg schnell testen:

```
jmoekamp@server:~# ping -s 192.168.2.200
PING 192.168.2.200: 56 data bytes
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=0. time=93.191 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=1. time=104.319 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=2. time=102.006 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=3. time=103.962 ms
^C
----192.168.2.200 PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev=93.191/100.870/104.319/5.219
```

Wie man sieht, haben sämtliche Pakete eine Latenz von knapp über 100ms. Um zu zeigen, dass es sich um die Auswirkung des Treibers und nicht einfach nur um ein langsames Netzwerk handelt, entfernen wir diese konfiguration nun wieder.

```
jmoekamp@server:~# ifhit 192.168.2.200 -z
(::ffff:192.168.2.200)
is turned off
```

Eine kurze Überprüfung ergibt nun wesentlich kürzere Ping-Zeiten.

```
jmoekamp@server:~# ping -s 192.168.2.200
PING 192.168.2.200: 56 data bytes
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=0. time=6.024 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=1. time=4.099 ms
^C
----192.168.2.200 PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev=4.099/5.061/6.024/1.361
jmoekamp@server:~#
```

Man kann aber auch Paketverluste simulieren:

```
jmoekamp@server:~# ifhit 192.168.2.200 -i 5 -n 2 -o 1
(::ffff:192.168.2.200)
  Drop interval=5 pkts, drop length=2 pkts, drop gap=1 pkts
```

Diese Konfiguration simuliert ein Netzwerk in dem alle fünf Pakete zwei Pakete verworfen warden, allerdings sind diese zwei Pakete nicht aufeinanderfolgend, sondern ein Paket wird wiederum zwischen den beiden paketen wird durchgelassen.

```
jmoekamp@server:~# ping -s 192.168.2.200
PING 192.168.2.200: 56 data bytes
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=0. time=4.060 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=1. time=3.953 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=2. time=4.776 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=3. time=4.264 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=4. time=7.889 ms
(icmp seq=5 fehlt)
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=6. time=2.406 ms
(icmp seq=7 fehlt)
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=8. time=4.824 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=9. time=3.739 ms
64 bytes from Server.home.moellenkamp.org (192.168.2.200): icmp_seq=10. time=3.593 ms
^C
----192.168.2.200 PING Statistics----
11 packets transmitted, 9 packets received, 18% packet loss
round-trip (ms)  min/avg/max/stddev=2.406/4.389/7.889/1.495
```

Eben genau dieses Pattern kann man nun in der Ausgabe eines Ping sehen.

Leider verfügt `hxbt` über keine manpage, auf die ich im Rahmen dieses Artikels verweisen könnte, deswegen möchte ich zur weiteren Verdeutlichung der Fähigkeiten auf die –h funktion des ifhit-Befehls hinweisen.

Darüber hinaus

Eine so kurze Darstellung kann natürlich nicht umfassend all die Funktionen behandeln, die zur Netzwerkvirtualisierung genutzt werden können. So ist der IP-Stack von Solaris in der Lage, Bandbreitenlimitierung und Paketpriorisierung auf Basis von Regeln durchzuführen. Auch kann man leicht in den einzelnen Zonen die in Solaris verfügbaren Firewallfunktionen nutzen.

So lässt sich mit Solaris einfach auch ein verteiltes System mittels eingebauter Features simulieren, ohne das der Aufwand eine Vielzahl physikalisch getrennter Komponenten oder einer Hardwarevirtualisierung getrieben werden muss. Denn schlussendlich: Auch bei der Simulation eines komplexen Netzwerkes läuft weiterhin nur ein Betriebsystemkernel[5] ohne die Zwischenschicht eines Hypervisors, und kann so die Virtualisierung sehr effizient durchführen.

Anhänge
Prüfen Sie vor Rückfragen bitte http://www.c0t0d0s0.org/pages/appendix_doag2011_1.html auf etwaige Erweiterungen oder Korrekturen zu diesem Text.

Kontaktadresse:

Jörg Möllenkamp
Oracle B.V. & Co. KG
Kühnehöfe 5
D-22761 Hamburg

Telefon: +49 172 8318433
E-Mail joerg.moellenkamp@oracle.com
Internet: http://www.oracle.com

[5] Naturgemäß gerät dieser Ansatz dann ins Schwanken, wenn man gerade mehrere unterschiedliche Betriebssysteme nutzen will, aber dafür gibt es dann aber ja eben Hardwarevirtualisierung.