Virtual Network User Mode Linux - VNUML 1.6

Studienarbeit
4. Oktober 2006

Nadia Ettaous
Fachbereich Informatik
Universität Koblenz-Landau

Betreuer:
Prof. Dr. Ch. Steigner
Dipl.-Inf. Harald Dickel

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Diese Studienarbeit soll eine Einführung in die Arbeit mit virtual network user mode linux (VNUML) geben. Mit Hilfe dieser Arbeit möchte ich speziell die Version VNUML 1.6 näher bringen und die wesentlichen Unterschiede, Vor- und Nachteile zur Version 1.5 zeigen.
In den nächsten zwei Kapiteln wird auf das Thema VNUML und UML oberflächlich eingegangen. Das darauffolgende Kapitel befasst sich mit der Installation von VNUML 1.6, der Vorraussetzung und den möglichen Fehlermeldungen. Wenn dies abgeschlossen ist, wird VNUML 1.6 mit eigenen Beispielen ausfürlich, praktisch und theoretisch vorgestellt. Danach werden die wesentlichen Unterschiede von VNUML 1.5 zu VNUML 1.6 beschrieben.Zum Abschluss sind noch ein Kapitel mit kurzen Begriffsdefinitionen und der Anhang mit allen XML-Dateien zu finden.
Auf den Aufbau einer XML-Datei möchte ich in meiner Arbeit nicht weiter eingehen. Dazu verweise ich auf die Arbeit von Thomas Chmielowiec [Tom06] und Tim Keupen [Tim06]. In diesen Arbeiten sind die XML-Syntax und Semantik ausfürlich beschrieben.

2 VNUML allgemein

2.1 Definition

VNUML (oder Virtual Network User Mode Linux) ist ein Tool das entwickelt wurde um (komplexe) Netzwerksimulationen zu kreieren, welches auf UML basiert. Durch die Verwendung von UML (User Mode Linux) können beliebig viele virtuelle Rechner in einem Szenario, auf einem einzigen physischen Linux Hostrechner, erzeugt werden, die nahezu alle Eigenschaften eines realen Linux Rechners haben. Dabei wird jede einzelne VM (Virtelle Maschine) frei konfiguriert und als Prozess auf einem Rechner ausgeführt. Somit ist es möglich beliebig grosse Netzwerktopologien virtuell auf zu bauen und zu testen.

"Die Simulation besteht aus einer Reihe von Kommandos, die auf jedem virtuellen Rechner beim Start ausgeführt werden, und wird in quasi Echtzeit auf dem Host durchgeführt, nachdem das Szenario in diesem aufgebaut wurde. Da es sich bei den virtuellen Rechnern um Prozesse mit einem eigenen Kernel, Dateisystem und Arbeitsspeicher handelt, ist es möglich nahezu beliebige Programme (ohne besondere Hardwareaufforderungen) auf ihnen zu installieren, die dann im Laufe der Simulation ausgeführt werden können. Somit eignet sich VNUML hervorragend zum Testen von verteilter Software oder Netzwerkprotokollen."[AndTim05]

Zur Konfiguration eines Szenarios dient eine XML-Datei. In dieser Datei, die unter Verwendung der Sprache XML beschrieben wird, sind Informationen zu den virtuellen Rechnern und der eigentlichen Netzwerktopologie enthalten. Somit ist XML die VNUML-Sprache. Damit das Szenario aufgebaut und simuliert werden kann, wird ein Parser, der aus einem Perl-Skript besteht, benötigt. Dieser ist dann der sogenannte vnuml-Parser.

2.2 Historie

"VNUML wurde von Fermin Galan und David Fernandez am Telematics Engineering Department (DIT) der technischen Universität von Madrid entwickelt. Entstanden ist VNUML dabei aus dem Ero6IX-Projekt1, welches sich mit der Einführung von Ipv6 in Europa beschäftigt. VNUML wurde entwickelt um Ipv6-Szenarien auf Linux-Rechnern mit Zebra/Quagga Routing-Deamons zu testen. Die erste öffentliche Version des Programms (vnumlparser-1.2.1 und DTD1.2) wurde am 29. Juli 2003 veröffentlicht." [AndTim05]
Mittlerweile steht auf der offiziellen Projekthomepage , unter http://jungla.dit.upm.es/ vnuml/, eine Testversion zu VNUML Version 1.7 zu Verfügung.

3 User Mode Linux

User Mode Linux, bzw UML, ist ein Linux-Kern, der nicht wie sonst direkt auf die Hardware aufsetzt, sondern als Prozess in einem Wirt-Linux-System (Hostsystem) läuft. Zielgruppe sind hierbei hauptsächlich Entwickler oder fortgeschrittene System-/Netzwerkadministratoren.
Vieles ist durch die Anwendung von UML einfacher umzusetzen und handhabbar. Netzwerkdienste z.B. können in einer UML Umgebung komplett isoliert vom Hauptsystem ablaufen. Oft wird UML auch benutzt um einen "Honeypot"aufzusetzen, mit dem man dann die Sicherheit eines Computers oder Netzwerks testen kann. UML kann auch eingesetzt werden , um neue Software einschliesslich Linux-Kernels zu debuggen oder zu testen , ohne das Hauptsystem zu beeinflussen. [Wiki06]

4 Installation von VNUML 1.6

4.1 Voraussetzungen

Die wohl wichtigste Voraussetzung für die Installation von vnuml ist genügend Speicherplatz im Hauptspeicher (256MB+) und ein einigermassen schneller Prozessor (ca. 500MHz+). Soviel Speicher ist vorallem deswegen notwendig,

weil vnuml für jeden Teilnehmer im virtuellen Netzwerk eine virtuelle Maschine mit einem echten Linux-Kernel und einem eigenen Filesystem in den Hauptspeicher lädt.

Es gibt mehrere Möglichkeiten vnuml zu installieren, wobei bei der Wahl wichtig ist die vorhandenen Systemvoraussetzungen zu beachten. Zum einen steht eine vnuml-Live DVD, die an der Universität Koblenz entwickelt wurde, zum download bereit. Durch diese live-DVD wird ein erster Einblick in die Welt der vnuml gewährt. So kann vnuml von der bootbaren Knoppix live-DVD gestartet werden. Auch wenn eine feste Installation hier nicht nötig ist, brauch der Rechner genügend Arbeitsspeicher.
Die andere Variante ist die statische Installation, wobei man zwischen einer langwierigen und fehleranfälligen manuellen Installation wählen kann, oder sich für die automatische Installation durch den vnuml-Installer entscheidet. Diese Variante, die Installation per Offline-Installer, ist die wohl bequemste und hilfreicheste Variante die geboten wird. Voraussetzung für eine statische Installation ist eine lauffähige Linux-Distribution. Ein weiteres Muss ist das vorhandensein folgender Linux-Pakete, die entweder manuell vor der Installation oder während vom Offline-Installer geprüft wird:

- gcc (C-Compiler)
- make (GNU Make Befehl)
- perl (Perl Interpreter)
- tar (Packprogramm tar)

Fehlende Pakete sollten dann entweder aus dem Internet geladen und installiert werden, oder mit Hilfe eines Paketmanagers passieren. Unter Suse Linux geschieht dies mit dem Paketmanager Yast und unter Debian mit dem Befehl apt-get install namedespakets.

In meinem Fall liegt die Linux-Distribution Suse 10.0 vor. Deswegen habe ich mich für die komfortable statische Installation mit Hilfe des Offline Installers entschieden, der hier Ädresse-smallßur Verfügung gestellt wird.

4.2 Installation

VNUML setzt auf eine grosse Anzahl von Programmpaketen und Modulen auf. Diese werden alle gesammelt und im Installationspaket VNUML Offline Installer gebündelt. Damit wurde das langwierige suchen erleichtert. Die Installation und Konfiguration der ganzen Pakete in der richtigen Reihenfolge wurde zusätzlich durch ein shell-Skript (bash) übernommen, der ebenfalls im Offline Installer vorhanden ist.

Nachdem alle notwendigen Voraussetzungen erfüllt sind , kommt es nun zur eigentlichen Installation. Als erstes sollte der VNUML-Offline-small-Installer runtergeladen und im Verzeichnis ihrer Wahl gespeichert werden. Danach sollte die Datei VNUML-Offline-small.zip entpackt werden, dies geschieht eventuell in einer Linux shell mit dem Befehl bzip2 -d Dateiname.bz2. Nach dem erfolgreichen Entpacken kommt es zum nächsten Schritt. Da ich die aktuelle Version VNUML 1.6 benötige, müssen zusätzlich ein Filesystem und ein UML Kernel, die notwendig für die Konfiguration der VMs sind, unter der offiziellen Homepage http://jungla.dit.upm.es/ vnuml/ runtergeladen werden. Auch VNUML (Current release) sollte runter geladen und im Ordner VNUML-Offline-small gespeichert und entpackt werden. Archiv-Dateien können unter anderem mit dem Befehl gzip -d Dateiname.tar.gz im shell entpackt werden.

Bevor ich mit der Installation fortfahre, bietet es sich an die uml-utilities, unter dem Ordner Pakete, manuell über Yast zu installieren. Da die häufigsten Fehlermeldungen dies bemängeln.

Als nächstes wird eine Linux-shell , entweder über Konsole oder Eingabeaufforderung, geöffnet und als root angemeldet. Das Anmelden als root passiert mit dem Befehl su und dem darauffolgenden Passwort. Dann sollte mit dem Befehl cd in das Verzeichnis gewechselt werden, indem sich der Ordner Offline-Installer befindet. Bsp.: cd /home/projekt/ettaous/VNUML-Offline-small.
Nun wird durch das eingeben von ./install das Installationsskript gestartet. Sollten bei der Installation Problem auftreten, wird man vom Installationsprogramm durch eine Fehlermeldung darauf hingewiesen. Ebenfalls erscheinen Handlungsanweisungen um den Fehler zu beheben. Da das Installationsprogramm selbstständig abläuft und der genaue Verlauf zu schnell geschieht, kann man dies mit dem Befehl ./install –debug verhindern und jeden Schritt einsehen. Somit stoppt das Programm nach jedem Schritt, damit man die Bildschirmausgabe besser verfolgen kann. Sollten Fehler auftreten, müssen diese erst behoben und danach die Installation mit ./install wieder gestartet werden.

Das "install"-Paket macht alles automatisch:

- Dateisystem und Kernel kopieren
- Programmpakete entpacken und installieren (uml utilities, bridge utilities...)
- Perl Module entpacken und installieren
- "vnuml"entpacken und installieren

Nach der erfolgreichen Installation geht es nun weiter zur Anwendung von VNUML 1.6.

4.3 Mögliche Fehlermeldungen

Fehler	**Lösungen**
Uml-utilities	Manuelle Installation - Yast: gewünschte Suse 10.0 CD einlegen und laden - apt: apt-get install uml-utilities
Start von vnumlparser.pl ergibt "Binary missing"	Manuelle Installation des Pakets bridge-utilities
VMs können nicht mit einander kommunizieren	Firewall deaktivieren; Bsp. Suse Linux über Yast → Sicherheit und Benutzer → Firewall

5 VNUML 1.6

5.1 Einleitung

Die ältere VNUML Version 1.5 war sehr root-fixiert, was sich aber in der aktuellen Version ändern soll. Die neue Version unterstützt daher auch "normale"User, ohne root-Privilegien. Dabei sind drei Möglichkeiten, VNUML zu benutzen, zu unterscheiden.

1. Limited User Privileges
2. User Privileges
3. Full Root Privileges

Jedes dieser Privilegien möchte ich einmal theoretisch und anhand eines Beispiels in den nächsten Kapiteln erläutern.

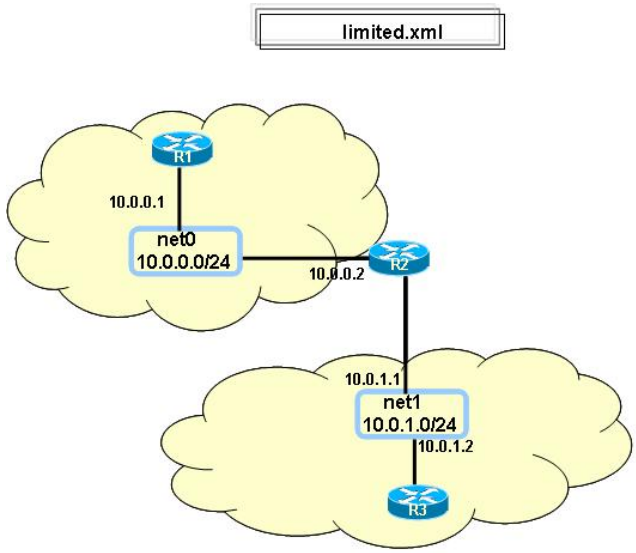
5.2 Limited User Privileges

Mit dem Limited User Privileg kann VNUML ohne root Rechte benutzt werden. Leider besteht dadurch kein Netzwerkzugang zu den einzelnen VMs vom host, und auch keine Verbindung vom externen Netzwerk. Die einzige Möglichkeit mit den VMs zu kommunizieren entsteht durch direktes login. Zur Beschreibung einer Datei dienen unter anderem diese Befehle:

<**global**> Innerhalb dieses Tags werden Parameter deklariert, die für die ganze Simulation geltend sind. Ein Beispiel wäre die filesystem-Datei für die VMs.

<**net**> Zum kreieren von virtuellen Netzwerken wird dieser Tag benutzt. Jedes Netzwerk erhält einen Namen und wird später, in der Definition der VMs dazu benutzt zu erkennen, an welchem Netzwerk die VMs angebunden sind.
<**vm**> Durch diesen Tag wird eine VM deklariert. Darin definiert man z.B. durch den <if>-Tag zur Beschreibung der Netzwerkschnittstelle oder durch den <root>-Tag die statischen Routen der VM.

Ich möchte nun alle weiteren Anweisungen anhand eines selbstkreierten Szenarios darstellen. Die XML-Datei von diesem und allen weiteren Szenarios in meiner Arbeit befinden sich im Anhang.
Nachdem eine XML-Datei bzw. eine Simulation fertig geschrieben und gespeichert wurde, kann man diese innerhalb einer shell (xterm) hochfahren. Der Befehl dazu lautet -t, Bsp.: vnumlparser.pl -t Dateiname.xml -v
Dadurch wird das Szenario geladen, samt der Netzwerktopologie und allen darin enthaltenen VMs die in der Datei definiert wurden. Durch den Zusatz von -v (verbose) ,kann beobachtet werden was genau bei der Ausführung des Befehls passiert. Diesen Befehl zeige ich anhand meines Beispiels limited.xml, das hier zusätzlich bildlich veranschaulicht wird.




```
projekt@licher: /ettaous/Dateien> vnumlparser.pl -t limited.xml -vB
```

Nach ein paar Augenblicken sollte sich für jede deklarierte VM ein xterm

öffnen. Sollte dies nicht so schnell passieren, ist das kein Grund zur Beunruhigung, denn je nach dem wie umfangreich das Szenario ist, dauert es etwas bis dieses hochgefahren und alle VMs geladen sind. Um danach innerhalb der VM kommunizieren zu können, muss man sich erst in die jeweilige VM einloggen, mit dem Kennwort: *root*und dem Passwort: *xxxx*.

```
R3 login: root
Passwort:
Linux (none) 2.6.12.3-bs9-xt-2m #8 Sat Nov 12 00:55:32 CET 2005 i686
GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with AABSOLULUTELY NO WARRANTY, to the extent
permitted by applicable law.
R3: #
```

Danach können Befehle eingegeben und ausgeführt werden, wie z.B. ls, ifconfig usw.

```
R3: # ifconfig
eth1      Link encap:Ethernet HWaddr FE:FD:00:00:03:01
inet addr:10.0.1.2 Bcast:10.255.255.255 Mask:255.255.255.0
inet6 addr: fe80::fcfd:ff:fe00:301/64 Scope:link
UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0b) TX bytes:278 (278.0b)
Interrupt:5

lo        Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP Loopback RUNNING MTU:16436 Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:560 (560.0b) TX bytes:560 (560.0b)
```

Zur Kontrolle ob eine Verbindung zwischen zwei VMs existiert wird der Befehl ping oder traceroute benutzt. Dazu habe ich ein paar Auszüge aus

dem Szenario limited.xml hinzugefügt.

```
R3: #ping 10.0.0.2
PING 10.0.0.2(10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=31.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.768 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.864 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.869 ms

R3: #traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 38 byte packets
1 10.0.1.1 (10.0.1.1) 1.705 ms 1.173 ms 1.158 ms
2 10.0.0.1 (10.0.0.1) 30.512 ms 2.116 ms 2.300 ms
```

Bei der Ausführung von ifconfig, starten die VMs mit eth1, nicht wie üblich mit eth0, da diese Schnittstelle für den management network reserviert ist. Zum Beenden des Szenarios verwendet man -d: vnumlparser.pl -d limited.xml -v.

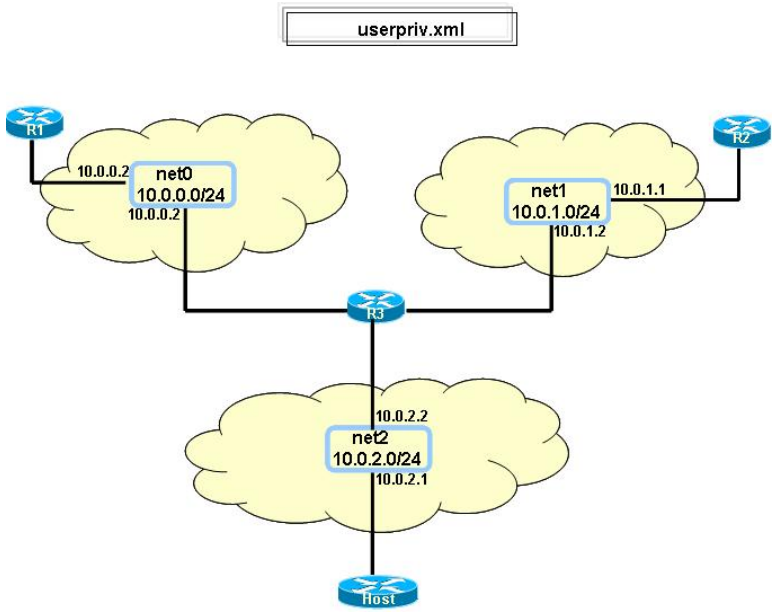
```
projekt@licher: /ettaous/Dateien> vnumlparser.pl -d limited.xml -vB
```

Dadurch erhält jede VM eine CAD (Ctrl+Alt+Delete) und wird runter gefahren. Der Kernel brauch etwas länger und die geöffneten xterms schliessen sich nach und nach. Um das Szenario danach wieder hoch zu fahren wird der Befehl -t, wie bereits erklärt, benutzt.

5.3 User Privileges

5.3.1 Verbindung zum Host

Der vnumlparser.pl läuft auch hier, wie beim limited user mode, ohne root Privileg. Ausser bei der Verwendung des management network und beim Zugriff auf externe Netzwerke sind root Privilegien notwendig. Das heisst die Verbindung zum Host ist möglich, ebenso die Verbindung zum Internet, wenn der Host als Router fungiert.Den Zugriff bzw die Verbindung zum Host möchte ich anahnd des nächsten Beipsiels zeigen.
Hier hat die virtuelle Maschine R3 direkten Zugriff auf den Host über das Netzwerk net2.



Der grosse Unterschied an diesem Beispiel, im Gegensatzzum Beispiel limited.xml, ist das net2-Netzwerk. In der dazugehörigen XML-Datei ist die Verbindung von R3 zum Host durch das Socket beschrieben, das im <net>-Tag definiert wird, und mit der virtuellen Maschine R3 verbunden ist.
Um das Szenario mit User Privilegien zu starten sind davor noch einige Anweisungen notwendig. Diese Anweisungen müssen allerdings als Root erfolgen, damit ein "normaler"User das Szenario mit dem vnumlparser.pl hochfahren kann. In meinem Fall heisst der User "projekt".
Dieser User, projekt, kann entweder über YAST oder über ein Terminal angelegt werden und zur Gruppe vnuml hinzugefügt werden. Um zu überprüfen das der Benutzer projekt tatsächlich existiert, benötigt man nur die Eingabe des Befehls "id projekt"oder schaut sich die Benutzerliste über YAST an. Ebenfalls unter dem Pfad /var/local/run/vnuml werden später die Sockets gespeichert.
Wenn sie sich dafür entscheiden ihren Benutzer über die Konsole anlegen zu wollen, so könnte dies zum Beispiel so aussehen:

1. Sie starten eine Konsole und loggen sich als Root ein:

```
projekt@licher:~> su
Password:
licher:/home/projekt #
```

2. Dann geben Sie diesen Befehl ein:

```
licher:/home/projekt # usermod -G vnuml,
‘groups projekt | sed ’s/^.\ +:[[:space:]]*//;s/[[:space:
]]\\+/,/g’‘ projekt
```

ODER

3. Starten Sie den Paketmanager YAST und verfolgen diesen Pfad um einen Benutzer anzulegen: Yast → Sicherheit → Benutzer → vnuml Nachdem projekt erfolgreich als User unter der Gruppe vnuml angelegt wurde, wird nun, weiterhin als root angemeldet, eine TUN/TAP Vorrichtung erstellt. Diese TUN/TAP Vorrichtunggehört dem User vnuml an und dient als Abbruchstelle für die Host-Verbindungim net2-Netzwerk.

4.
```
licher:/home/projekt # tunctl -u vnuml -t tap0
Set ’tap0’ persistent and owned by uid 1001
```

   Nun wird die Host IP und die dazugehörige statische Route angelegt, mit der der Host und die virtuelle Maschine R3 verbunden sind.

5.
```
licher:/home/projekt # ifconfig tap0 10.0.2.1
netmask 255.255.255.0 up
```

6.
```
licher:/home/projekt # route -A inet add -net
10.0.0.0/16 gw 10.0.2.2
```

   Die folgenden Anweisungen musste ich etwas umschreiben, damit der uml_switch gestartet werden kann.
   ORIGINAL-Anweisung lautet:

```
licher:/home/projekt # su -pc
’uml_switch -tap tap0 -unix /var/local/run/vnuml/net2.ctl
< /dev/null > /dev/null \&’ vnuml
```

7.
```
licher:/home/projekt # $uml_switch -tap tap0 -unix
/var/local/run/vnuml/net2.ctl < /dev/null > /dev/null \&
[1] 5790
```

   Der Grund weshalb meine Anweisung so aussieht ist der, weil unter dem von mir benutzten System der Benutzer vnuml keinerlei Rechte zum ausführen hat.

Als nächstes wird der vnuml Gruppe Schreib- und Leseprivilegien über den verwendeten Socket erteilt. Auch hier musste ich die Originalanweisungen umgehe, damit die Datei net2.ctlan ihrme Bestimmungsort landet. Die dazugehörigen Anweisungen sollten,laut Tutorial aus Madrid, so aussehen:

```
licher:/home/projekt # chmod  g+rw /var/local/run/vnuml
/net2.ctl
```

Damit ich problemlos weiter arbeiten konnte, habe ich diese Reihe von Anweisungen angegeben:

8.
```
licher:/home/projekt # ls -l /var/local/run/vnuml
/net2.ctl
srwxr-xr-x  1 root root 0 2006-09-11 13:50 /var/local/run/vnuml/net2.ctl
```

9.
```
licher:/home/projekt # chown vnuml:vnuml /var/local
/run/vnuml/net2.ctl
```

10.
```
licher:/home/projekt # ls -l /var/local/run
/vnuml/net2.ctl
srwxr-xr-x  1 vnuml vnuml 0 2006-09-11 13:50 /var/local/run
/vnuml/net2.ctl
```

Dann kann der eigentliche Befehl eingegeben werden:

11.
```
licher:/home/projekt # chmod  g+rw /var/local/run/vnuml
/net2.ctl
```

12.
```
licher:/home/projekt # ls -l /var/local/run/vnuml
/net2.ctl
srwxrwxr-x  1 vnuml vnuml 0 2006-09-11 13:50 /var/local
/run/vnuml/net2.ctl
```

Die Datei net2.ctl existiert nun unter dem Pfad /var/local/run/vnuml/net2.ctl und kann vom User projekt gelesen und beschrieben werden. Mit dieser Voraussetzung ist es jetzt möglich das Szenario unter dem Benutzer projekt laufen zu lassen.˜
Um das Szenario userpriv.xml zu starten, wird zunächst eine neue Konsole geöffnet und als Benutzer projekt angemeldet. Dann sollte man den Pfad eingeben, wo sich das Szenario befindet, und anschliessend diese Kommandozeile eingeben.

13.
```
projekt\@licher:˜> cd /home/projekt/ettaous/Dateien/
```

14. ```
    projekt\@licher:~/ettaous/Dateien> vnumlparser.pl -t
    userpriv.xml -vB
    ...R1 sshd is ready (socket style): 10.0.0.1 (if id=1)
    R2 sshd is ready (socket style): 10.0.1.1 (if id=1)
    R3 sshd is ready (socket style): 10.0.0.2 (if id=1)
    host> /bin/rm -f /home/projekt/.vnuml/LOCK
    Total time elapsed: 628 seconds
    projekt\@licher:~/ettaous/Dateien>
    ```

Nach dem erfolgreichen Hochfahren des Szenarios lassen sich die einzelnen Verbindungen zwischen den VM's und dem Host mit traceroute überprüfen.

```
R1:~# traceroute -n 10.0.2.1
traceroute to 10.0.2.1 (10.0.2.1), 30 hops max, 38 byte packets
  1 10.0.0.2 29.281 ms 1.918 ms 1.615 ms
  2 10.0.2.1 5.573 ms 2.131 ms 2.239 ms
```

```
projekt@licher:~/ettaous/Dateien> ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=30.9 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=63 time=1.28 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=63 time=1.31 ms
64 bytes from 10.0.1.1: icmp_seq=4 ttl=63 time=1.40 ms
```

Sollte die Verbindung zum Host gesperrt sein, so muss man lediglich die Firewall ausschalten, und es dann erneut versuchen.

Das Szenario kann später ganz gewöhnlich unter projekt heruntergefahren werden.

1. ```
   projekt@licher:~/ettaous/Dateien> vnumlparser.pl -d
   userpriv.xml -vB
   ```

   Da aber noch der uml_switch Prozess für net2, die tap0 Vorrichtung und das Gateway existieren, müssen diese erst noch beendet werden. Dies geschieht alles unter Root. Zunächst wird der uml_switch Prozess beendet und das Socket gelöscht.

2. ```
   licher:/home/projekt # kill ‘ lsof -t /var/local/run
   /vnuml/net2.ctl‘
   ```

3. ```
   licher:/home/projekt # rm /var/local/run/vnuml/net2.ctl
   ~
   ```

   Danach wird die tap0 Vorrichtung dekonfiguriert.

4. `licher:/home/projekt # ifconfig tap0 down`

5. `licher:/home/projekt # tunctl -d tap0`
   `Set 'tap0' nonpersistent`

5.3.2 SSH-Key

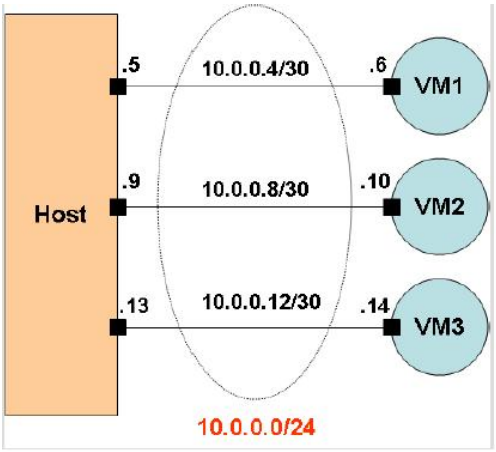
SSH öffnet und managet UML's, und damit man die Passwort-Nachfrage vermeidet, wird der public-key im ¡ssh_key¿-tag generiert. Um Kommando Sequenzen an die einzelnen virtuellen Maschinen zu senden verwendet VNUML *ssh*, der dazu notwendige ssh-key muss vorher allerdings noch generiert werden. Um den ssh-key zu generieren, gibt man diese Zeile ein: ssh-keygen -t rsa1. Dieser Schlüssel öffnet und managet die UMLs, der vom vnumlparser benutzt wird. Dadurch wird vermieden nach Passwörtern gefragt zu werden.

```
licher:/home/projekt # ssh-keygen -t rsa1
Generating public/private rsa1 key pair.
Enter file in which to save the key (/root/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/identity.
Your public key has been saved in /root/.ssh/identity.pub.
The key fingerprint is:
37:2c:1e:76:40:c0:39:29:d8:fb:b0:5b:b4:62:67:f7 root@licher
```

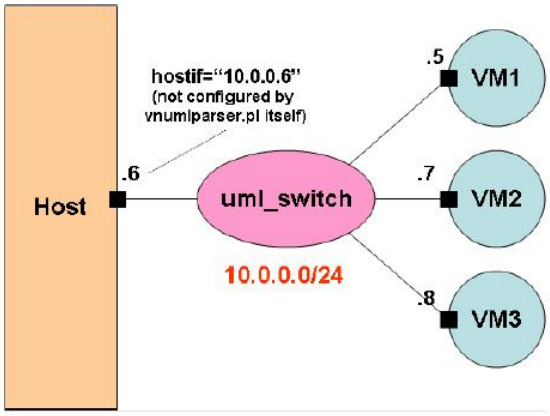
Bei der ersten Verwendung von ssh sollten die Anfragen mit yes beantwortet werden, dies geschieht normalerweise aber nur einmal.
Durch die Verwendung von ssh ist es nicht mehr nötig für jede einzelne VM ein xterm zu öffnen. Das ist vor allem dann ein Vorteil wenn in der Simulation viele VMs vorhanden sind.

5.4 Management Networks

Es gibt zwei Arten von management networks. Zum einen das *private-management* und zum anderen das *net-management*. Durch das private-management entsteht eine peer-to-peer Verbindung zwischen dem Host und jeder virtuellen Maschine, dazu muss der parser allerdings als root laufen.



Das net-management wird wie ein uml_switch network benutzt. Net wird durch type im <vm_mgmt>-tag deklariert. Bei dieser Variante spezifizieren die network und mask Attribute eine Netzwerk-Adresskette die der parser bei der Adressbezeichnung der virtuellen Maschinen benutzen wird.



Als nächstes wird das Szenario hochgefahren, wobei vorher noch über den Root die notwenigen Konfigurationen getätigt werden müsse. Da ich sie im vorherigen Kapitel ausführlich erklärt habe, möchte ich nun nur noch die

Kommandozeilen ohne weiteren Kommentar auflisten.

```
licher:/home/projekt # tunctl -u vnuml -t tap0
Set \'tap0\' persistent and owned by uid 1001
licher:/home/projekt # ifconfig tap0 10.250.0.1 netmask
255.255.255.0 up
licher:/home/projekt # uml_switch -tap tap0 -unix\\ /var
/local/run/vnuml/Mgmt_net.ctl < /dev/null > /dev/null &
[1] 7449
licher:/home/projekt # chown vnuml:vnuml /var/local/run/vnuml
/Mgmt_net.ctl
licher:/home/projekt # chmod g+rw /var/local/run/vnuml
/Mgmt_net.ctl
```

Nun möchte ich das Szenario userpriv2.xml hochfahren.
Nach dem erfolgreichen Hochfahren ist das einloggen in jede virtuelle Maschine vom host aus durch ssh möglich.

```
projekt@licher:~/ettaous/Dateien> vnumlparser.pl -t
userpriv2.xml -vB
main::main (506): No interface on the host is configured for
VM management with address 10.250.0.1/24
```

Leider war es mir nicht möglich weder mein eigenes Szenario, noch das vom Tutorial aus Madrid erfolgreich zu starten (siehe Fehlermeldung). Mir ist es ebenfalls nicht gelungen den Fehler zu beheben oder zu umgehen.
Um die Simulation zu beenden geht man wie üblich vor: vnumlparser.pl -d Dateiname.xml -v , wobei uml_switch, socket, tap0 und dessen gateway immer noch existieren. Deshalb müssen uml_switch und socket gelöscht werden, und tap0 dekonfiguriert.

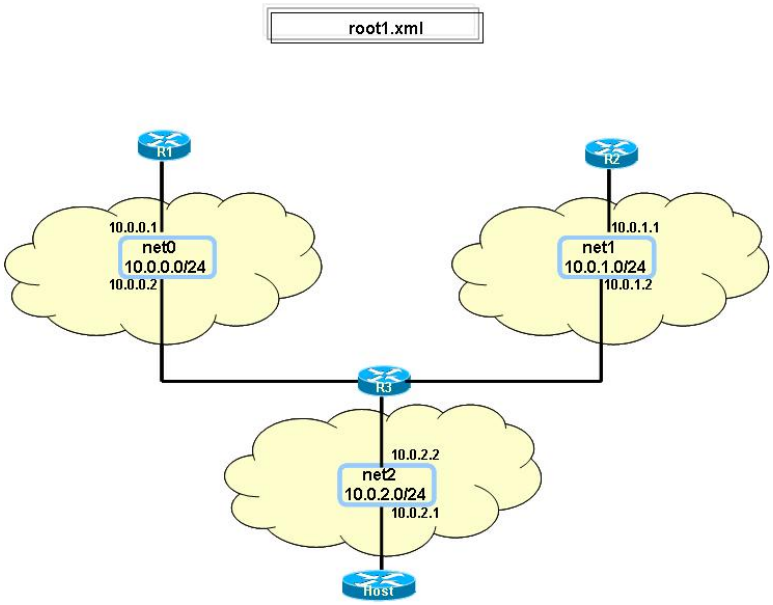
```
licher:/home/projekt # fuser -k /var/local/run/vnuml
/Mgmt_net.ctl
/var/local/run/vnuml/Mgmt_net.ctl:  7449
[1]+  GetÃ¶tet                uml_switch -tap tap0 -unix\\
/var/local/run/vnuml/Mgmt_net.ctl </dev/null >/dev/null
licher:/home/projekt # rm /var/local/run/vnuml/Mgmt_net.ctl
licher:/home/projekt # ifconfig tap0 down
```

```
licher:/home/projekt # tunctl -d tap0
Set 'tap0' nonpersistent
```

```
projekt@licher:~/ettaous/Dateien> vnumlparser.pl -P
userpriv2.xml -vB
```

5.5 Full Root Privileges

Nun kommen wir zu den root Rechten. Der <host>-tag kann nur benutzt werden wenn der parser vom root gelaufen wird. Deshalb reicht hier eine Konsole, unter der man sich als root anmeldet, mit Hilfe von su und dem Passwort. Danach sollte der Pfad eingegeben werden, indem sich das Szenario befindet, welches gestartet werden soll. Jetzt kann das Szenario root1.xml hochgefahren werden.




```
projekt@licher:~/ettaous/Dateien> su
Password:
licher:/home/projekt/ettaous/Dateien # vnumlparser.pl -t
root1.xml -vB
```

```
main::main (409): vnuml_dir /root/.vnuml does not exist or is not
readable/executable (user vnuml)

licher:/home/projekt/ettaous/Dateien # vnumlparser.pl -t
root1.xml -vB -u
root
(...)

R1 sshd is ready (socket style): 10.0.0.1 (if id=1)
R2 sshd is ready (socket style): 10.0.1.1 (if id=1)
R3 sshd is ready (socket style): 10.0.0.2 (if id=1)
host> /bin/rm -f /root/.vnuml/LOCK
Total time elapsed: 181 seconds
licher:/home/projekt/ettaous/Dateien #
```

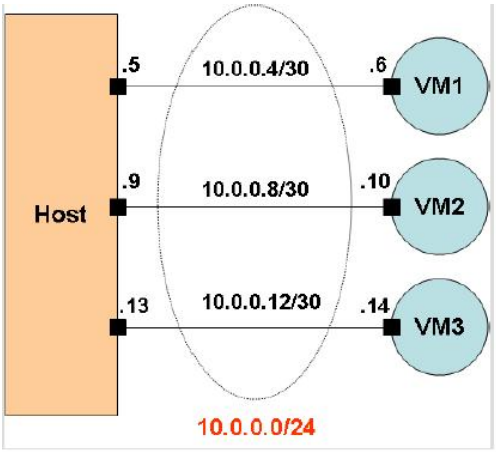
Zur Überprüfung der Verbindung zu den einzelnen VM's benutze ich traceroute.

```
licher:/home/projekt/ettaous/Dateien # traceroute -n 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.2.2  0.000 ms   0.000 ms   0.000 ms
 2  10.0.0.1  0.000 ms   0.000 ms   0.000 ms

licher:/home/projekt/ettaous/Dateien # traceroute -n 10.0.1.2
traceroute to 10.0.1.2 (10.0.1.2), 30 hops max, 40 byte packets
 1  10.0.1.2  1.638 ms   0.000 ms   0.000 ms
```

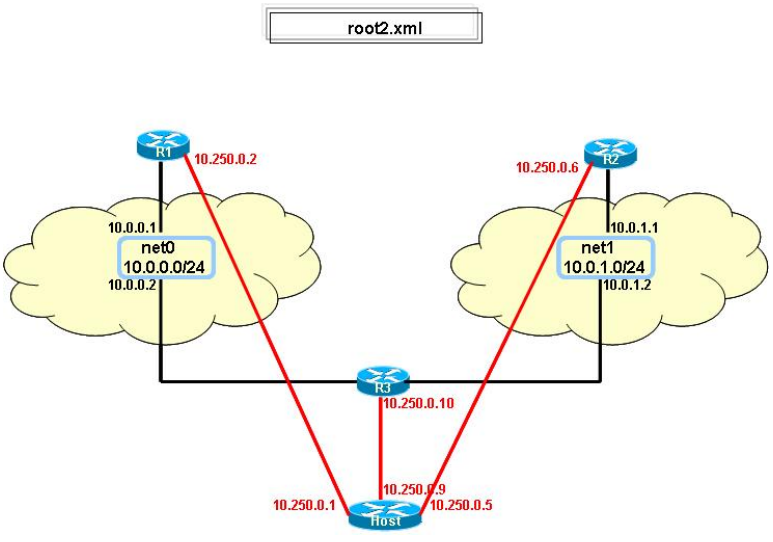
Das Szenario kann dann später wieder wie üblichmit -d heruntergefahren werden.

```
licher:/home/projekt/ettaous/Dateien # vnumlparser.pl -d
root1.xml -vB -u
root
```

Nun kommen wir zum private Management. Durch type=private im <vm_mgmt>-tag entsteht eine peer-to-peer Verbindung vom host zu jeder virtuellen Maschine (/30 network).



Bevor die Simulation gestartet wird muss allerdings noch ein ssh-key für den root generiert werden, wie zuvor beschrieben, weil ansonsten Beschwerden auftreten. Das nächste Szenario das benutzt wird heisst root2.xml, und ist wie alle anderen Beispiele als XML-Datei im Anhang zu finden.




```
licher:/home/projekt/ettaous/Dateien # vnumlparser.pl -t
root2.xml -vB -u
root
.
.
```

```
.
R1 sshd is ready (socket style): 10.250.0.2 (mng_if)
R2 sshd is ready (socket style): 10.250.0.6 (mng_if)
R3 sshd is ready (socket style): 10.250.0.10 (mng_if)
host> /bin/rm -f /root/.vnuml/LOCK
Total time elapsed: 147 seconds
licher:/home/projekt/ettaous/Dateien #
```

Nun logge ich mich vom, Host direkt in die vrituelle Maschine R1 ein und überprüfe einige Verbindungen.

```
licher:/home/projekt/ettaous/Dateien # ssh R1
WARNING: RSA1 key found for host r1
in /root/.ssh/known_hosts:13
RSA1 key fingerprint 7e:3d:7e:73:22:85:b8:03:ff:0f:b0:07:5b:
41:d8:ba.
The authenticity of host 'r1 (10.250.0.2)' can't be established
but keys of different type are already known for this host.
RSA key fingerprint is ad:98:21:6e:10:e2:50:51:4b:b8:4a:99:ad:
ac:b9:0d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'r1,10.250.0.2' (RSA) to the list of
known hosts.
Password:
R1:~#

R1:~# ping 10.250.0.1
PING 10.250.0.1 (10.250.0.1) 56(84) bytes of data.
64 bytes from 10.250.0.1: icmp_seq=1 ttl=64 time=0.552 ms
64 bytes from 10.250.0.1: icmp_seq=2 ttl=64 time=0.378 ms
64 bytes from 10.250.0.1: icmp_seq=3 ttl=64 time=0.378 ms
64 bytes from 10.250.0.1: icmp_seq=4 ttl=64 time=0.383 ms

R1:~# traceroute 10.250.0.1
traceroute to 10.250.0.1 (10.250.0.1), 30 hops max, 38 byte
packets
 1  10.250.0.1 (10.250.0.1)  1.068 ms  0.835 ms  0.836 ms

R1:~# traceroute 10.0.1.1
traceroute to 10.0.1.1 (10.0.1.1), 30 hops max, 38 byte
packets
 1  10.0.0.2 (10.0.0.2)  38.114 ms  3.511 ms  3.462 ms
 2  10.0.1.1 (10.0.1.1)  31.500 ms  4.978 ms  5.354 ms
```

Hier sieht man ganz deutlich das eine direkte Verbindung vom Host zu jeder virtuellen Maschine möglich ist.
Um eine virtuelle Maschine zu verlassen gibt man den Befehl exit ein.

```
R1:~# exit
logout
Connection to R1 closed.
licher:/home/projekt/ettaous/Dateien # cat /etc/hosts
#
# hosts         This file describes a number of hostname-to-
#               address mappings for the TCP/IP subsystem.
#               It is mostly used at boot time, when no name
#               servers are running. On small systems, this
#               file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost

# special IPv6 addresses
::1             localhost ipv6-localhost ipv6-loopback

fe00::0         ipv6-localnet

ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
127.0.0.2       linux.site linux
# VNUML BEGIN -- DO NOT EDIT!!!

# BEGIN: tutorial
# topology destroyed: Mi Mai 17 15:43:39 CEST 2006
# END: tutorial

# BEGIN: tutorial-r2
# topology destroyed: Mi Aug  9 12:45:24 CEST 2006
# END: tutorial-r2
# BEGIN: root2
# topology built: Mo Sep 11 16:09:03 CEST 2006
10.250.0.2 R1
```

```
10.250.0.6 R2
10.250.0.10 R3
# END: root2
# VNUML END
licher:/home/projekt/ettaous/Dateien #
```

Mit dem Befehl

```
cat /etc/hosts
```

werden die Szenarien angeben die ordnungsgemäss hoch- und runtergefahren wurden, und den Zustand des aktuellen Szenarios wieder gegeben. In diesem Beispiel verwende ich

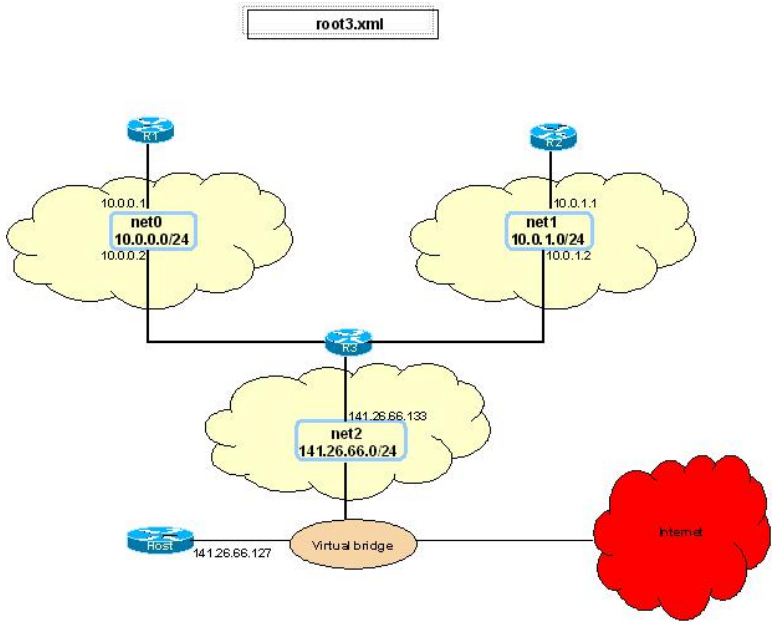
```
cat /etc/hosts
```

während das Szenario root2.xml läuft. Dabei wird das Datum und die Uhrzeit angegeben, an der ich das Szenario gestartet habe, und zusätzlich die im Szenario verwendeten virtuelle Maschinen.
Um das Szenario wieder herunterzufahren gebe ich diese Zeile ein.

```
licher:/home/projekt/ettaous/Dateien # vnumlparser.pl -d
root2.xml -vB -u
root
...
licher:/home/projekt/ettaous/Dateien # cat /etc/hosts
#
# hosts         This file describes a number of hostname-to-
#               address mappings for the TCP/IP subsystem.
#               It is mostly used at boot time, when no name
#               servers are running. On small systems, this
#               file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#

127.0.0.1       localhost

# special IPv6 addresses
::1             localhost ipv6-localhost ipv6-loopback

fe00::0         ipv6-localnet
```

```
ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes
ff02::2         ipv6-allrouters
ff02::3         ipv6-allhosts
127.0.0.2       linux.site linux
# VNUML BEGIN -- DO NOT EDIT!!!

# BEGIN: tutorial
# topology destroyed: Mi Mai 17 15:43:39 CEST 2006
# END: tutorial

# BEGIN: tutorial-r2
# topology destroyed: Mi Aug  9 12:45:24 CEST 2006
# END: tutorial-r2
# BEGIN: root2
# topology destroyed: Mo Sep 11 16:16:54 CEST 2006
# END: root2
# VNUML END
licher:/home/projekt/ettaous/Dateien #
```

Es ist möglich das ein VM direkt mit dem Host zum externen Netzwerk verbunden werden kann. Mit Hilfe des Attributs type="virtual bridge"im <net>-tag ist dies möglich. Dies möchte ich noch als letztes, die Layer 2 Interconnection bei vollen Root Rechten, vorstellen. Dazu habe ich ein eigenes Programm geschrieben, namens root3.xml. In diesem Szenario sind die Maschinen R1 und R2 sind direkt mit R3 verbunden, und die virtuelle Maschine R3 ermöglicht durch eine "virtual_bridge"Verbindung zum Internet. Die dazugehörige XML-Datei befindet sich im Anhang.



Nach dem erfolgreichen hochfahren kann durch ping getestet werden ob Verbindung zum Host und nach "Draussen"besteht. Zunächst wird die Verbindung von R3 zu R1 getestet:

```
licher:/home/projekt/ettaous/Dateien # ssh -1 10.0.1.2
The authenticity of host '10.0.1.2 (10.0.1.2)' can't be
established.
RSA1 key fingerprint is 7e:3d:7e:73:22:85:b8:03:ff:0f:b0:07:5b
:41:d8:ba.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.1.2' (RSA1) to the list of
known hosts.
Password:
Response:
R3:~# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.01 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.804 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.703 ms

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2126ms
rtt min/avg/max/mdev = 0.703/0.840/1.014/0.131 ms
```

Dieser Test verlief erfolgreich!
Dann wurde die Verbindung von R3 zum Host getestet:

```
R3:~# ping 141.26.66.127
PING 141.26.66.127 (141.26.66.127) 56(84) bytes of data.
64 bytes from 141.26.66.127: icmp_seq=1 ttl=64 time=0.564 ms
64 bytes from 141.26.66.127: icmp_seq=2 ttl=64 time=0.519 ms
64 bytes from 141.26.66.127: icmp_seq=3 ttl=64 time=0.516 ms

--- 141.26.66.127 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.516/0.533/0.564/0.021 ms
```

Auch hier besteht die verbindung.
Hier noch den Nachweis der Verbindung vom Host zur VM R2 und zu VM R3:

```
projekt@licher:~/ettaous/Dateien> ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=29.0 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=63 time=1.07 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=63 time=1.26 ms
64 bytes from 10.0.1.1: icmp_seq=4 ttl=63 time=1.13 ms
64 bytes from 10.0.1.1: icmp_seq=5 ttl=63 time=1.11 ms
^C64 bytes from 10.0.1.1: icmp_seq=6 ttl=63 time=1.19 ms
64 bytes from 10.0.1.1: icmp_seq=7 ttl=63 time=1.09 ms

--- 10.0.1.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6023ms
rtt min/avg/max/mdev = 1.073/5.132/29.059/9.768 ms
projekt@licher:~/ettaous/Dateien> ping 141.26.66.133
PING 141.26.66.133 (141.26.66.133) 56(84) bytes of data.
64 bytes from 141.26.66.133: icmp_seq=1 ttl=64 time=0.767 ms
64 bytes from 141.26.66.133: icmp_seq=2 ttl=64 time=0.500 ms
64 bytes from 141.26.66.133: icmp_seq=3 ttl=64 time=0.471 ms

--- 141.26.66.133 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.471/0.579/0.767/0.134 ms
```

Leider konnte ich die Verbindung von R3 zum Host nicht vorstellen, da ich bei meinen Tests zu keinem erfolgreichen Ergebnis gekommen bin.

6 Unterschiede von 1.5 zu 1.6

Seit der Version vnuml 1.5 hat sich einiges geändert. Der wohl gravierendste Unterschied von 1.5 zu 1.6 sind die drei wählbaren Benutzergruppen, während wir in der Version 1.5 nur mit vollen Root-Rechten arbeiten konnten. Somit ist auch anderen Benutzen, ohne vollen Zugang, die Möglichkeit geboten Szenarios hochzufahren und damit zu arbeiten.

- Limited User Privileges
- User Privileges
- Root Privileges

Welche Funktionen mit den verschiedenen Rechten verbunden sind, habe ich in den vorherigen Kapiteln veranschaulicht.
Der nächste bedeutende Unterschied ist das **Management**. Die management interfaces werden unter dem "structural tag"<vm_mgmt> konfiguriert. Die Attribute innerhalb des <vm_mgmt>-tags lauten: private, net oder none. Beispiele und die Unterschiede der Attribute dazu sind in meiner Arbeit bildlich veranschaulicht und dadurch gut nach zu voll ziehen. Weitere Tags wie zum Beispiel <host_mapping> und <mgmt_net> werden innerhalb des <vm_mgmt>tags beschrieben. Beim <host_mapping> werden die Namen der virtuellen Maschinen auf die entsprechenden management-device-IPs gemapped. Während beim mgmt_net zusätzliche Angaben wie sock und hostip hinzukommen, wobei sock den Pfad zum Socket angibt und hostip die IP-Adresse angibt, womit der uml_switch Prozess verbunden ist. Der boot Tag dient dazu, dass bei Verwendung beim Hochfahren eines Szenarios für jede einzelne virtuelle Maschine ein Terminal geöffnet wird. Der Befehl dazu lautet:

```
<boot>
     <con0>xterm</con0>
     <xterm>gnome-terminal,-t,-x</xterm>
</boot>
```

Neu ist unter anderem auch dass unter dem "structural tag"<vm> der Befehl zum starten oder beenden eines Szenarios nicht mehr <start> oder <stop> heisst, sondern nun durch <exec seq="start"> und <exec seq="stop"> ersetzt wurde.
Beispiel:

```
<exec seq="start" type="verbatim">nohup /usr/bin/hello
&lt;/dev/null &gt;/dev/null
2&gt;&amp;1 &amp; </exec>
<exec seq="stop" type="verbatim">killall hello</exec>
```

Dann ist es noch möglich innerhalb des "structural tags" <vm> weitere user mit Gruppenzugehörigkeit zu definieren.

Abschließend möchte ich alle Unterschiede in einer Tabelle gegenüberstellen.

VNUML 1.5	**VNUML 1.6**
	Neue tags: <vm_mgmt> <mgmt_net> <user> <group>
<ip_offset>	network, mask und offset innerhalb des <vm_mgmt>-tags
<ssh_key>	<ssh_version>
<global> → <host_mapping>	<vm_mgmt> → <host_mapping>

7 Ausblick

Durch die Version 1.6 sind einige neue Möglichkeiten gegeben, die vorher nicht geboten waren, wie zum Beispiel das Einbinden von Benutzern ohne Root-Rechten. Aber auch das Management-Network ist einer der Veränderungen der Version 1.6. VNUML wird auch in Zukunft immer weiter Benutzerfreundlich optimiert. Die Entwicklung dazu sind genaustens unter der VNUML-Homepage zu verfolgen.
Seit dem 27.Juli 2006 besteht die Version 1.7 auf der offiziellen VNUML-Homepage http://jungla.dit.upm.es/ṽnuml/doc/current/tutorial/index.html#tutorial_requirements. Welch Unterschiede und Neuheiten es in der aktuellsten Version im Gegensatz zur Version 1.6 gibt, können mit Hilfe des Tutorials und der Beipiele nachvollzogen werden.

8 Begriffsdefinitionen

boot erlaubt Angabe von Boot Parameternn für den Kernel. Parameter müssen sinnvoll gegeben werden, weil sie nicht vom Parser kontrolliert werden. Innerhalb des tags kann mit <con0> die Konsole auf dem Host angegeben werden, auf der der virtuelle Rechner seine Ausgaben schreiben soll. Die Ausgabe auf eine x-Konsole ist auch möglich (xterm).

chown Der Befehl wird verwendet, um den Inhaber einer Datei oder eines Verzeichnisses zu ändern.

socket Ein Socket (wörtlich übersetzt "Sockel"oder "Steckverbindungen") ist eine bi-direktionale Software-Schnittstelle zur Interprozess- (IPC)

oder Netzwerk-Kommunikation. Sockets sind die vollduplexfähige Alternative zu Pipes, FiFos oder Shared Memory.

tap Tap ist ein "virtual ethernet network device"und ist auf Schicht 2 des OSI-Modells implementiert.

tun Tun ist ein "virtual point-to-point network device"und ist auf Schicht 3 des OSI-Modells implementiert.

uml_switch Im Hintergrund läuft ein uml_switch-Prozess der die virtuellen Netzwerke implementiert und verwaltet. Möchte man einen Hub anstatt eines Switches verwenden, nimmt man zusätzlich die Angabe hub="yes". Für den uml_switch-Prozess sind nur User-Rechte notwendig.

virtual_bridge Es wird eine virtuelle Brücke zum Verbinden der Rechner verwendet. Über die Angabe von "external"lässt sich eien virtuelle Schnittstelle mit einer echten Schnittstelle des Hosts verbinden, und so eine Verbindung zum Internet realisieren.

9 Anhang

9.1 limited.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">

<vnuml>
  <global>
    <version>1.6</version>
    <simulation_name>limited</simulation_name>
    <automac/>
    <vm_mgmt type="none" />
    <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
    <default_kernel>/usr/local/share/vnuml/kernels/linux
    </default_kernel>
  </global>
  <net name="net0" mode="uml_switch" />
  <net name="net1" mode="uml_switch" />
  <vm name="R1">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
```

```
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.2">10.0.1.0/24</route>
  </vm>
  <vm name="R2">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <if id="2" net="net1">
        <ipv4>10.0.1.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.2">10.0.0.0/24</route>
    <route type="inet" gw="10.0.1.1">10.0.1.0/24</route>
    <forwarding type="ip"/>
  </vm>
  <vm name="R3">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net1">
      <ipv4>10.0.1.2</ipv4>
    </if>
    <route type="inet" gw="10.0.1.1">10.0.0.0/24</route>
    <forwarding type="ip" />
  </vm>
</vnuml>
```

9.2 userpriv.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.6</version>
    <simulation_name>userpriv</simulation_name>
    <automac/>
    <vm_mgmt type="none" />
```

```
    <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
    <default_kernel>/usr/local/share/vnuml/kernels/linux
    </default_kernel>
  </global>
  <net name="net0" mode="uml_switch" />
  <net name="net1" mode="uml_switch" />
  <net name="net2" mode="uml_switch" sock="/var/local/run/vnuml/net2.ctl" />

  <vm name="R1">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.2">default</route>
  </vm>
  <vm name="R2">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net1">
      <ipv4>10.0.1.1</ipv4>
    </if>
    <route type="inet" gw="10.0.1.2">default</route>
  </vm>
  <vm name="R3">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <if id="2" net="net1">
      <ipv4>10.0.1.2</ipv4>
    </if>
    <if id="3" net="net2">
      <ipv4>10.0.2.2</ipv4>
    </if>
```

```
    <!--<route type="inet" gw="10.0.1.2">10.0.2.0/24</route>-->
    <forwarding type="ip" />
  </vm>
 </vnuml>
```

9.3 userpriv2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.6</version>
    <simulation_name>userpriv2</simulation_name>
    <automac/>
    <vm_mgmt type="net" network="10.250.0.0" mask="24">
            <mgmt_net sock="/var/local/run/vnuml/Mgmt_net.ctl"
            hostip="10.250.0.1"/>
    </vm_mgmt>
    <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
    <default_kernel>/usr/local/share/vnuml/kernels/linux
    </default_kernel>
  </global>
  <net name="net0" mode="uml_switch" />
  <net name="net1" mode="uml_switch" />
  <net name="net2" mode="uml_switch" />

  <vm name="R1">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.2">default</route>
  </vm>
  <vm name="R2">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
```

```
    <if id="1" net="net1">
      <ipv4>10.0.1.1</ipv4>
    </if>
    <route type="inet" gw="10.0.1.2">default</route>
  </vm>
  <vm name="R3">
    <boot>
      <con0>xterm</con0>
      <!--xterm>gnome-terminal,-t,-x</xterm-->
    </boot>
    <if id="1" net="net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <if id="2" net="net1">
      <ipv4>10.0.1.2</ipv4>
    </if>
    <if id="3" net="net2">
      <ipv4>10.0.2.2</ipv4>
    </if>
    <!--<route type="inet" gw="10.0.1.2">10.0.2.0/24</route>-->
    <forwarding type="ip" />
  </vm>
 </vnuml>
```

9.4 root1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.6</version>
    <simulation_name>root1</simulation_name>
    <automac/>
    <vm_mgmt type="none" />
    <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
    <default_kernel>/usr/local/share/vnuml/kernels/linux
    </default_kernel>
  </global>
  <net name="net0" mode="uml_switch" />
  <net name="net1" mode="uml_switch" />
  <net name="net2" mode="uml_switch" />
```

```
<vm name="R1">
  <boot>
    <con0>xterm</con0>
    <!--xterm>gnome-terminal,-t,-x</xterm-->
  </boot>
  <if id="1" net="net0">
    <ipv4>10.0.0.1</ipv4>
  </if>
  <route type="inet" gw="10.0.0.2">default</route>
</vm>
<vm name="R2">
  <boot>
    <con0>xterm</con0>
    <!--xterm>gnome-terminal,-t,-x</xterm-->
  </boot>
  <if id="1" net="net1">
    <ipv4>10.0.1.1</ipv4>
  </if>
  <route type="inet" gw="10.0.1.2">default</route>
</vm>
<vm name="R3">
  <boot>
    <con0>xterm</con0>
    <!--xterm>gnome-terminal,-t,-x</xterm-->
  </boot>
  <if id="1" net="net0">
    <ipv4>10.0.0.2</ipv4>
  </if>
  <if id="2" net="net1">
    <ipv4>10.0.1.2</ipv4>
  </if>
  <if id="3" net="net2">
    <ipv4>10.0.2.2</ipv4>
  </if>
  <!--<route type="inet" gw="10.0.1.2">10.0.2.0/24</route>-->
  <forwarding type="ip" />
</vm>
<host>
        <hostif net="net2">
                <ipv4>10.0.2.1</ipv4>
        </hostif>
        <route type="inet" gw="10.0.2.2">10.0.0.0/16</route>
</host>
</vnuml>
```

9.5 root2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">

<vnuml>
        <global>
                <version>1.6</version>
                <simulation_name>root2</simulation_name>
                <ssh_version>1</ssh_version>
                <ssh_key>~/.ssh/identity.pub</ssh_key>
                <automac/>
                <vm_mgmt type="private" network="10.250.0.0" mask="24">
                        <host_mapping />
                </vm_mgmt>
                <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
                <default_kernel>/usr/local/share/vnuml/kernels/linux
                </default_kernel>
        </global>
        <net name="net0" mode="uml_switch" />
        <net name="net1" mode="uml_switch" />

        <vm name="R1">
                <!--<boot>
                        <con0>xterm</con0>
                        <xterm>gnome-terminal,-t,-x</xterm>
                </boot>-->
                <if id="1" net="net0">
                        <ipv4>10.0.0.1</ipv4>
                </if>
                <route type="inet" gw="10.0.0.2">10.0.1.0/24</route>
                <exec seq="start" type="verbatim">nohup /usr/bin/hello
                &lt;/dev/null &gt;/dev/null
2&gt;&amp;1 &amp; </exec>
                <exec seq="stop" type="verbatim">killall hello</exec>
        </vm>
        <vm name="R2">
                <!--<boot>
                        <con0>xterm</con0>
                        <xterm>gnome-terminal,-t,-x</xterm>
                </boot>-->
                <if id="1" net="net1">
```

```
                        <ipv4>10.0.1.1</ipv4>
                </if>
                <route type="inet" gw="10.0.1.2">10.0.0.0/24</route>
        </vm>
        <vm name="R3">
                <!--<boot>
                        <con0>xterm</con0>
                        <xterm>gnome-terminal,-t,-x</xterm>
                </boot>-->
                <if id="1" net="net0">
                        <ipv4>10.0.0.2</ipv4>
                </if>
                <if id="2" net="net1">
                        <ipv4>10.0.1.2</ipv4>
                </if>
                <route type="inet" gw="10.0.1.2">10.0.1.0/24</route>
                <route type="inet" gw="10.0.0.2">10.0.0.0/24</route>
                <forwarding type="ip"/>
        </vm>
</vnuml>
```

9.6 root3.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">

<vnuml>
  <global>
    <version>1.6</version>
    <simulation_name>root3</simulation_name>
    <automac/>
    <vm_mgmt type="none" />
    <default_filesystem
type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial
</default_filesystem>
    <default_kernel>/usr/local/share/vnuml/kernels/linux
    </default_kernel>
  </global>
  <net name="net0" mode="uml_switch" />
  <net name="net1" mode="uml_switch" />
  <net name="net2" mode="virtual_bridge" external="eth0"/>
  <vm name="R1">
    <!--<boot>
      <con0>xterm</con0>
```

```
    </boot>-->
    <if id="1" net="net0">
      <ipv4>10.0.0.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.2">141.26.66.0/24</route>
    <route type="inet" gw="10.0.0.2">10.0.1.0/24</route>
  </vm>
  <vm name="R2">
<!--<boot>
      <con0>xterm</con0>
    </boot>-->
    <if id="1" net="net1">
      <ipv4>10.0.1.1</ipv4>
    </if>
    <route type="inet" gw="10.0.1.2">141.26.66.0/24</route>
    <route type="inet" gw="10.0.1.2">10.0.0.0/24</route>
  </vm>
  <vm name="R3">
        <!--<boot>
      <con0>xterm</con0>
    </boot>-->
    <if id="1" net="net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <if id="2" net="net1">
      <ipv4>10.0.1.2</ipv4>
    </if>
    <if id="3" net="net2">
      <ipv4>141.26.66.133</ipv4>
    </if>
    <route type="inet" gw="141.26.66.127">141.26.66.0/24</route>
    <forwarding type="ip" />
  </vm>
  <host>
    <hostif net="net2">
      <ipv4>141.26.66.127</ipv4>
    </hostif>
    <physicalif name="eth0" ip="141.26.66.127" mask="255.255.248.0"
    gw="141.26.66.1" />
    <route type="inet" gw="141.26.66.133">10.0.0.0/24</route>
    <route type="inet" gw="141.26.66.133">10.0.1.0/24</route>

  </host>
</vnuml>
```

10 Quellenverzeichnis

Literatur

[AndTim05] André Volk und Tim Keupen.
Anleitung zur Installation des Netzwerk-Simulators VNUML
2005, pp 4-5

[Wiki06] *de.wikipedia.de*

[Mueller05] Markus Müller.
Simulation mit User Mode Linux
VNUML-Einführung in die Simulation von Rechnernetzen
WS 05/06

[VNUML05] Fermin Galan, David Fernandez *VNUML Tutorial Version 1.6*
http:'
'jungla.dit.upm.es/ vnuml/doc/1.6/tutorial
2005

[Tim05] Tim Keupen.
User Mode Linux
2005, pp 10-13

[Tim06] Tim Keupen
Verteilte Simulationen und externe Verbindungen mit vnuml
2006

[Tom06] Thomas Chmielowiec
VNUML-Versionen 1.5 und 1.6
2006