Fachbereich 4: Informatik
Arbeitsgruppe Steigner
Universitätsstraße 1
56070 Koblenz

Seminar
"Simulationen mit User Mode Linux"
Wintersemester 2005/2006

VNUML- Einführung in die Simulation von Rechnernetzen

Dozent:
Dipl.-Inform. Harald Dickel

Verfasser:
Markus Müller, Raiffeisenstraße 12, 56294 Münstermaifeld
markus.mueller@uni-koblenz.de

Inhaltsverzeichnis

1 Einleitung

Ziel dieser Seminararbeit ist es, eine Einführung in die Arbeit mit Virtual Network User Mode Linux (VNUML) zu geben.

Neben der Vorstellung von VNUML und dessen Bestandteilen sollen auch weitere, bei der Arbeit mit Rechnernetzen hilfreiche Werkzeuge eingeführt werden. Deren Funktionsweise wird anhand des Beispielnetzes aus dem VNUML-Tutorial [1] erläutert.

1.1 Was ist VNUML?

Virtual Network User Mode Linux (VNUML) ist ein Werkzeug zur Simulation von Rechnernetzen.

Grundlage bildet hierbei die Virtualisierungstechnik User Mode Linux (UML), die es erlaubt, auf einem einzigen physischen Linux-Hostrechner mehrere virtuelle Linux-Rechner zu betreiben. Die einzelnen virtuellen Maschinen (VM) können hierbei relativ frei konfiguriert werden und beanspruchen jeweils einen Teil der Rechenleistung des Gastsystems. Ihre Dateisysteme werden durch Images in Form einer Datei auf dem Wirtsrechner abgebildet.

Die Konfiguration eines Simulationsszenarios geschieht bei VNMUL mittels eines leicht erlernbaren XML-Dialektes, der sogenannten VNUML-Sprache. Die damit erstellte XML-Datei enthält Informationen zu den virtuellen Rechnern sowie der eigentlichen Netztopologie. Das Umsetzen eines solchen Szenarios in eine laufende Simulation geschieht durch den aus einem Perl-Skript bestehenden VNUML-Parser.

Entwickelt wurde VNUML am Telematics Engineering Department (DIT) der Technischen Universität von Madrid im Rahmen der Teilnahme am Euro6IX-Projekt.

Zielsetzung dieses Projektes ist die Verbreitung des IPv6-Protokolls als Nachfolger des bisherigen, mit seinen 32 Bit großen Adressen für die Zukunft des Internet ungeeigneten IPv4-Adressstandards.

Um das bis dato langwierige Erstellen von Testszenarien für IPv6 zu erleichtern, begannen Fermín Galán und David Fernández mit der Entwicklung von VNUML. Dabei ist man inzwischen bei Versionsnummer 1.6.2-1 (22.11.2005) angelangt. Aufgrund der weiten Verbreitung beschäftigt sich diese Seminarbeit jedoch mit der etwas älteren Version 1.5 vom 21. März diesen Jahres.

2 Installation

Die Installation bzw. Inbetriebnahme eines VNUML-Systems kann im wesentlichen auf drei unterschiedliche Arten erfolgen.

Die einfachste Variante besteht hierbei in der Nutzung der im Rahmen eines Projektpraktikums an der Universität Koblenz-Landau entwickelten VNUML-Live-DVD. Diese kann in Form eines ungefähr 1 GB großen Images von [2] heruntergeladen werden. Nach dem Brennen des Images auf DVD ist von dieser das Booten eines voll funktionierenden VNUML-Testsystemes möglich. Grundlage bildet die Linux-Live-Distribution Knoppix in

Version 3.8.2, die wiederum auf Debian basiert. Nachteil der Arbeit mit einer Live-DVD ist die etwas längere Reaktionszeit des Systems, da benötigte Daten vom im Gegensatz zu einer Festplatte langsamen DVD-Laufwerk nachgeladen werden müssen. Des Weiteren wird ein RAM-Laufwerk angelegt, was die zur Verfügung stehende Hauptspeicherkapazität verringert.

Steht bereits eine funktionierende Linux-Installation zur Verfügung, bietet sich eine statische VNUML-Installation an. Hierfür steht zum einen ein auf der offiziellen VNUML-Homepage erhältliches Installationspaket zur Verfügung. Dieses benötigt zur Installation einiger eventuell nicht vorhandener Perl-Pakete eine funktionierende Internetverbindung. Sollte diese nicht vorhanden sein, können die betreffenden Pakete auch manuell nachinstalliert werden, was in der Vergangenheit jedoch vereinzelt zu Problemen führte.

Zum Anderen existiert ein komfortabler Offline-Installer, der wie die Live-DVD im Rahmen des Projektpraktikums "Routingsimulation" an der Universität Koblenz-Landau entwickelt wurde und unter [3] erhältlich ist. Dieser enthält alle zur Einrichtung eines VNUML-Systemes benötigten Pakete und erlaubt somit eine unkomplizierte Installation. Daher ist er dem "offiziellen" Installer vorzuziehen.

Nach dem Entpacken des rund 240 MB großen Offline-Installationspaketes sollte eine Konsole mit Superuser-Rechten gestartet werden (Kommando `su`). Danach wird über den Befehl `./install` das Installationsskript gestartet. Treten bei der Installation Probleme auf, informiert das Installationsprogramm den Anwender mittels ausführlicher Fehlermeldungen. Nach dem Abschluss der Installation erscheint die unten abgebildete Nachricht in der Konsole.

```
******************************
***       Erster Test !     ***
******************************

**** Der VNUML-Parser sollte jetzt Informationen zu
     gültigen Parametern ausgeben.

*** Wenn jetzt keine Fehlermeldung erscheint,
    dann haben Sie es wahrscheinlich geschafft!!
```

Um danach die korrekte Funktionsweise der VNUML-Installation zu testen sollte mit `cd /usr/local/share/vnuml/examples` in das Verzeichnis der VNUML-Installation gewechselt werden. Durch Ausführen des Befehls `vnumlparser.pl -t tutorial.xml -vB` wird das im folgenden näher beschriebene Beispielnetz aus dem VNUML-Tutorial gestartet, das sich anschließend über `vnumlparser.pl -d tutorial.xml -vB` auch wieder Beenden lässt. Eine nähere Erläuterung des Befehls findet sich in Abschnitt 4.

2.1 Probleme bei der Installation/Nutzung

Das wohl am häufigsten während der Installation auftretende Problem dürfte das Fehlen des Pakets `uml-utilities` sein. Abhilfe schafft hier die manuelle Installation mittels eines

Paketmanagers wie beispielsweise YAST (SuSE Linux) oder apt (Debian). Das Paket ist bei den meisten Distributionen bereits auf den Installationsmedien enthalten. Bei der Verwendung von apt sorgt der Befehl `apt-get install uml-utilities` für die Installation. Das entsprechende Kommando für YAST lautet `yast install uml-utilities`.

Selbst bei einer erfolgreichen Installation kann es vorkommen, das der VNUML-Parser `vnumlparser.pl` den Start mit der Fehlermeldung "Binary missing" quittiert. In diesem Fall ist eine manuelle Installation des Paketes `bridge-utils` notwendig, das ebenfalls auf den Installationsmedien der meisten Linux-Distributionen vorhanden ist.

Können die virtuellen Rechner eines VNUML-Szenarios nicht miteinander kommunizieren, liegt dies häufig an einer aktivierten Firewall. Durch Deaktivieren der Firewall kann hier Abhilfe geschaffen werden. Dies geschieht beispielsweise unter SuSE Linux über YAST -> Sicherheit und Benutzer -> Firewall.

3 Die VNUML-Sprache

Virtual Network User Mode Linux besteht im wesentlichen aus zwei Komponenten: der VNUML-Sprache sowie dem VNUML-Parser. Die VNUML-Sprache besteht aus einem XML-Dialekt, über den sich Netzwerktopologien relativ einfach beschreiben lassen. Diese Beschreibungen werden in XML-Dateien gespeichert. Die Auswertung der VNUML-Dateien geschieht dann über den VNUML-Parser, der diese auf syntaktische Korrektheit überprüft und, falls gewünscht, die Simulation startet.

In diesem Kapitel wird die Syntax der VNUML-Sprache in Version 1.5 beschrieben. Das folgende Kapitel 4 erläutert die Funktionsweise des zugehörigen VNUML-Parsers. Mit den Syntax-Unterschieden in den VNUML-Versionen 1.5 und 1.6 beschäftigt sich der zweite Vortrag dieses Seminars.

Allgemein können bei der VNUML-Sprache drei Arten von Tags unterschieden werden [4]:

- Strukturelle Tags
  Dienen der Strukturierung der VNUML-Datei, enthalten jedoch keinerlei Semantik.

- Topologie Tags
  Definieren die Semantik der Netzwerktopologie und werden beim Starten und Beenden einer Simulation ausgewertet.

- Simulations-Tags
  Beinhalten die Semantik der Simulation und werden im Kommandoausführungsmodus ausgewertet (siehe hierzu auch Abschnitt 4).

3.1 Grundgerüst einer VNUML-Datei

Das Grundgerüst einer typischen VNUML-Datei lässt sich wie folgt darstellen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
 <global>
  ...
 </global>
 <net name="Netz 0" />
 ...
 <vm name="Rechner_1">
  ...
 </vm>
 ...
 <host>
  ...
 </host>
</vnuml>
```

Die ersten beiden Zeilen sind Standarddefinitionen, die in jeder XML-Datei vorkommen müssen. In Zeile 1 wird die verwendete XML-Version spezifiziert während in der zweiten Zeile der Dokumententyp festgelegt wird. Außerdem wird hier der Pfad zur verwendeten Dokumenttypdefinition (Document Type Definiton, DTD) angegeben.

Die eigentliche Definition eines Netzwerkszenarios wird mit dem Tag <**vnuml**> eingeleitet und am Ende der Datei auch wieder mit dem entsprechenden End-Tag </**vnuml**> beendet.

3.2 Globale Definitionen: Der Tag <global>

In diesem Abschnitt der VNUML-Datei werden globale Definitionen vorgenommen [5]. Diese Definitionen gelten für das gesamte Szenario und damit insbesondere für alle virtuellen Rechner. Unten abgebildet ist der <**global**>-Abschnitt des Beispielnetzes aus dem online verfügbaren VNUML-Tutorial [1], anhand dessen die wichtigsten Tags erläutert werden.

```
<global>
 <version>1.5.0</version>
 <simulation_name>tutorial</simulation_name>
 <ssh_key>/root/.ssh/identity.pub</ssh_key>
 <automac/>
 <ip_offset>100</ip_offset>
 <host_mapping/>
</global>
```

Zur Angabe der verwendeten VNUML-Version muss der <**version**>-Tag vorkommen, durch <**version**>1.5</**version**> wird diese beispielsweise auf Version 1.5 gesetzt.

Ebenso erforderlich ist die Angabe eines eindeutigen Simulationsnamens mittels des Tags <**simulation_name**>.

Über den Tag <**ssh_key**> erfolgt die Installation des öffentlichen SSH-Schlüssels des Hostrechners auf den virtuellen Maschinen, wodurch die ständige Abfrage des Passwortes entfällt.

Über den Tag <**automac**/> kann die automatische Vergabe von MAC-Adressen an die Netzwerkinterfaces der VMs aktiviert werden.

Die virtuellen Rechner werden standardmäßig über ihre IP-Adresse angesprochen. Um diese auch über ihren Rechnernamen erreichen zu können, fügt man den Tag <**host_mapping**> in den <**global**>-Abschnitt ein.

3.3 Konfiguration virtueller Netze: Der Tag <net>

Über den Tag <**net**> werden die virtuellen Netzwerke des Szenarios konfiguriert. Dabei ist jedem Netzwerk mit Hilfe des name-Attributes ein eindeutiger Name zuzuweisen:

```
<net name="Net0"/>
```

Ein Netzwerk kann unter VNUML in zwei verschiedenen Modi betrieben werden:

- Virtual bridge based networks
  Die einzelnen virtuellen Rechner werden hierbei über eine virtuelle Bridge miteinander verbunden. Über das external-Attribut kann diese virtuelle Bridge mit einem physischen Interface des Hostrechners und somit einem externen Netzwerk wie beispielsweise dem Internet verbunden werden. Dieser Ausführungsmodus benötigt zwingend Superuser-Rechte.

- UML switch based networks
  Hierbei werden die einzelnen Rechner über einen virtuellen Switch miteinander verbunden. Dieser kann mit Hilfe des hub-Attributes in einen Hub verwandelt werden. Ein Anschließen an externe Netzwerke ist nicht möglich. In diesem Modus werden keine Root-Rechte benötigt.

3.4 Konfiguration virtueller Maschinen: Der Tag <vm>

Über den Tag <**vm**> werden die virtuellen Rechner konfiguriert [5]. Dabei stehen zahlreiche Optionen zur Verfügung. In diesem Abschnitt lassen sich insbesondere auch die Netzwerkschnittstellen konfigurieren wodurch die Zuordnung eines virtuellen Rechners zu einem Netzwerk möglich wird. Anhand des unten abgebildeten Auszugs aus der Konfigurationdatei des Beispielnetzes sollen einige der verwendbaren Tags und Konfigurationsmöglichkeiten vorgestellt werden.

```
<vm name="uml2">
    <filesystem type="cow">/usr/local/share/vnuml/
      filesystems/root_fs_tutorial</filesystem>
```

```
    <if id="1" net="Net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <route type="inet" gw="10.0.0.3">default</route>
  </vm>
```

Über das name-Attribut des <**vm**>-Tags wird jedem Rechner ein eindeutiger Name zugewiesen.

Die Angabe des Speicherortes des Dateisystem-Images erfolgt über den <**filesystem**> -Tag.

Im Abschnitt <**vm**> werden die Netzwerkschnittstellen des virtuellen Rechners konfiguriert. Die Angabe der Schnittstellennummer erfolgt über das id-Attribut, im obigen Beispiel hat das Interface die Bezeichnung "eth1". Über das Attribut net wird das Interface einem vorher definierten Netzwerk zugeordnet.

Die Konfiguration einer IPv4-Adresse geschieht mit Hilfe des <**ipv4**>-Tags. Zusätzlich kann auch über <**ipv6**> eine IPv6-Adresse vergeben werden.

Zur Deklaration statischer Routen wird der <**route**>-Tag benutzt. Im Beispiel wird als Standardrouter der Rechner mit der IP-Adresse 10.0.0.3 angegeben, was der VM mit dem Name "UML3" entspricht (siehe hierzu auch Abbildung 1 auf Seite 12).

3.5 Konfiguration des Hostrechners: Der Tag <host>

Soll der Hostrechner Teil des Szenarios sein, wird dieser über den <**host**>-Tag konfiguriert [5]. Dabei stehen im wesentlichen die gleichen Möglichkeiten wie bei der Konfiguration der virtuellen Maschinen im Abschnitt <**vm**> zur Verfügung.

```
<host>
    <hostif net="Net3">
      <ipv4>10.0.3.2</ipv4>
    </hostif>
    <route type="inet" gw="10.0.3.1">10.0.0.0/16</route>
  </host>
```

Zu beachten ist jedoch, dass der Tag zur Konfiguration der Netzwerkschnittstellen hier <**hostif**> und nicht <**if**> heißt. Das Attribut id zur Angabe des Schnittstellennamens existiert nicht, vielmehr wird der Schnittstelle der Name des zugehörigen Netzwerkes zugewiesen. Im obigen Ausschnitt lautet der Name der Schnittstelle also "Net3".

4 Der VNUML-Parser

Das Umsetzen des in der VNUML-Datei definierten Szenarios in eine laufende Simulation erfolgt mittels des VNUML-Parsers `vnumlparser.pl`. Dieses Perl-Skript analysiert die Eingabedatei auf syntaktische Korrektheit und sorgt, je nach Parameter, beispielsweise für

das Booten der virtuellen Maschinen und den Aufbau der virtuellen Netztopologie. Die allgemeine Syntax des Parseraufrufes lässt sich wie folgt beschreiben:

- `vnumlparser.pl -t|-r|-d|-P VNUML-Datei [weitere Parameter]`
- `vnumlparser.pl -x Kommandosequenz@VNUML-Datei [weitere Parameter]`

Der Parser kann also mit fünf verschiedenen Schaltern gestartet werden, wobei jeder Schalter einem bestimmten Ausführungsmodus entspricht [4]:

- `-t`: Aufbau der Netztopologie (build topology)
  Hierüber wird ein in der angegebenen VNUML-Datei spezifiziertes Szenario gestartet. Dabei werden die virtuellen Maschinen gebootet und die Netztopologie entsprechend den Angaben in der VNUML-Datei aufgebaut. Je nach Komplexität des Szenarios und Leistungsfähigkeit des Hostrechners kann dies eine gewisse Zeit in Anspruch nehmen.

- `-r`: Neustarten eines Szenarios (restart scenario)
  Mit Hilfe dieses Schalters wird ein bereits vorher gestartetes Szenario neu gestartet.

- `-d`: Beenden eines Szenarios (destroy scenario)
  Ein vorher mit `-t` gestartetes Szenario wird beendet und alle verwendeten Ressourcen werden wieder freigegeben. Die Dateisysteme der einzelnen virtuellen Maschinen bleiben dabei erhalten und stehen beim nächsten Start des Szenarios wieder unverändert zur Verfügung.

- `-P`: "Gewaltsames" Beenden eines Szenarios
  Lässt sich ein Szenario nicht über den dafür vorgesehenen Schalter `-d` beenden, kann über den Schalter `-P` ein Beenden des Szenarios erzwungen werden. Dabei werden jedoch auch die Dateisysteme der virtuellen Maschinen gelöscht, so dass alle darin gespeicherten Daten verloren gehen.

- `-x`: Ausführen von Kommandos (execute commands)
  In diesem Modus kann dem Parser ein in der VNUML-Datei mittels des <**exec**>-Tag festegelegtes Kommando als weiterer Parameter übergeben werden, das dieser dann ausführt. Hat man etwa mit

  ```
  <exec seq="test" type="verbatim">/usr/local/bin/test1.sh</exec>
  ```

  die Kommandosequenz "test" in der Datei `szenario.xml` festgelegt, erreicht man durch Aufruf des Parsers mit `vnumlparser.pl -x test@szenario.xml` das Ausführen dieser Kommandosequenz und damit das Starten des Skriptes "test1.sh" im Verzeichnis `/usr/local/bin/`.

4.1 Allgemeine Parseroptionen

In diesem Abschnitt werden einige weitere nützliche und in allen Ausführungsmodi nutzbare Parameter vorgestellt. Daneben existieren weitere, für den jeweiligen Ausführungsmodus spezifische Parameter, die in [4] ausführlich beschrieben werden.

- `-v`: verbose mode
  Durch Angabe dieses Parameters wird der Parser angewiesen, die Ergebnisse der von ihm ausgeführten Kommandoaufrufe ausführlich auf dem Bildschirm darzustellen. Diese Funktion kann insbesondere bei Fehlfunktionen nützliche Informationen zu den entsprechenden Hintergründen liefern.

- `-g`: debug mode
  Hiermit wird der Parser angewiesen, sämtliche Kommandos auf der Kommandozeile auszugeben, ohne diese jedoch wirklich auszuführen. Somit ist ein Testen auch ohne tatsächliches Hochfahren eines Szenarios möglich.

- `-B`: blocking mode
  Das Einloggen auf einem virtuellen Rechner geschieht mittels einer SSH-Verbindung, d.h. auf allen virtuellen Rechnern muss ein entsprechender SSH-Server laufen. Wird der Parser mit dem Parameter `-B` gestartet, wartet er beim Booten der virtuellen Maschinen solange, bis auf jeder von ihnen der SSH-Server gestartet und somit ein Verbinden vom Hostrechner aus möglich ist.

- `-i`: interactive mode
  Dieser Parameter wird im Zusammenhang mit dem "verbose mode" benutzt und erzwingt vom Benutzer ein Bestätigen jedes einzelnen, vom Parser auszuführenden Kommandos durch Drücken einer Taste.

5 Nützliche Netzwerk-Werkzeuge

In diesem Abschnitt werden einige bei der Arbeit mit Rechnernetzen hilfreiche Werkzeuge vorgestellt, die bei den meisten Unix/Linux-Distributionen zum Standardumfang zählen.

Mit ihrer Hilfe lassen sich Informationen über wichtige Parameter wie beispielsweise bestehende Verbindungen zwischen einzelnen Rechnern oder Art und Umfang des Netzverkehrs sammeln.

Die Demonstration der verschiedenen Kommandos geschieht anhand des Beispielnetzes aus dem VNUML-Tutorial [1]. Dessen Aufbau wird in der unten dargestellten Abbildung 1[1] beschrieben.

[1] Diese Grafik wurde dem VNUML-Tutorial [1] entnommen.

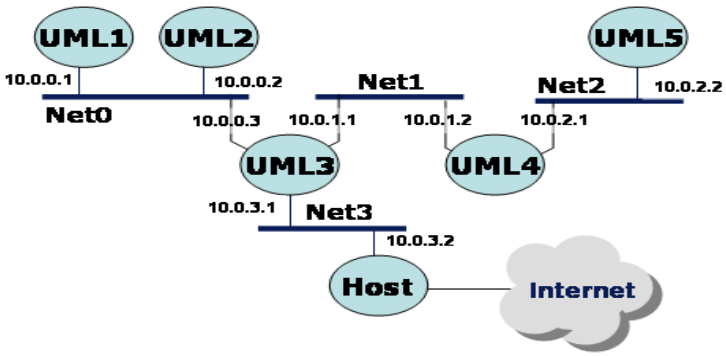

Abbildung 1: Aufbau des Beispielnetzes

5.1 ping

Mit Hilfe des Programmes `ping` lässt sich die Verfügbarkeit eines entfernten Rechners in einem Netzwerk überprüfen. Die IP-Adresse dieses Rechners ist hierbei als Parameter anzugeben. Alternativ kann auch eine URL übergeben werden, die dann vom Betriebssystems in die entsprechende IP-Adresse aufzulösen ist.

Zur Realisierung seiner Funktionalität verwendet `ping` das Internet Control Message Protocol (ICMP), das dem Austausch von Diagnose- und Fehlermeldungen in einem Netzwerk dient. `ping` sendet ein "Echo Request"-Paket an den Zielrechner, der bei korrekter Erreichbarkeit mit einem "Echo Reply"-Paket antwortet. Ist der Zielrechner hingegen nicht erreichbar, antwortet der Router mit einem "Host unreachable"-Paket. Aus dieser Rückmeldung kann jedoch im Allgemeinen nicht eindeutig auf die Unerreichbarkeit des Zielrechners geschlossen werden, da dieser sich beispielsweise hinter einer Firewall befinden könnte, die aus Sicherheitsgründen ankommende ICMP-Pakete verwirft und auch so die "Host unreachable"-Meldung auslöst.

Entwickelt wurde `ping` im Jahr 1983 von Mike Muuss für das Betriebssystem BSD Unix 4.2a. Inzwischen gehört es jedoch zum Standardumfang eines jeden modernen Betriebssystemes. Seinen Namen verdankt das Programm dem Geräusch eines Sonars, mit dessen Hilfe beispielsweise U-Boote geortet werden können.

Unten abgebildet befindet sich die Ausgabe des Ping-Kommandos beim "Anpingen" der virtuellen Maschine "UML2" ausgehend von der VM "UML1" (siehe Abbildung 1).

Neben den Sequenznummmern der ICMP-Pakete wird die Round Trip Time (RTT), also die Zeit, die ein Paket für die Strecke Sender-Empfänger-Sender benötigt, angezeigt. In einer abschließenden Statistik werden verschiedene Daten dargestellt, hierzu zählen die Anzahl der insgesamt übertragenen und empfangenen Pakete, der prozentuale Anteil verlorener Pakete sowie die minimale, durchschnittliche und maximale RTT.

Neben dem manuellen Abbruch der Ausführung von `ping` über die Tastenkombination STRG+c kann mit dem Parameter `-c` *Anzahl* die Anzahl zu sendender Pakete festgelegt werden. Wurde diese gesendet, wird die Ausführung ohne weiteres Zutun beendet.

```
uml1:~# 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=79.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.448 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.438 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.443 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.438/20.311/79.918/34.414 ms
```

5.2 ssh

Das SSH-Protokoll erlaubt den Aufbau authentifizierter und verschlüsselter Verbindungen zwischen zwei Rechnern in einem unsicheren Netzwerk. Auf dem Zielrechner können nach einem erfolgreichen Verbindungsaufbau anschließend beliebige Kommandos in einem Terminal ausgeführt werden.

Zur Verschlüsselung kommt das Public-Key-Verfahren zum Einsatz. Nach der erfolgten Authentifizierung der Gegenstelle werden die öffentlichen Schlüssel ausgetauscht. Alle an eine Gegenstelle zu übertragenden Daten werden daraufhin mit deren öffentlichem Schlüssel verschlüsselt. Da nur die Gegenstelle selbst über den geheimen, privaten Schlüssel verfügt, kann nur sie die Daten mit dessen Hilfe auch wieder rekonstruieren.

Entwickelt wurde SSH als Ersatz solcher Protokolle, die insbesondere sensitive Daten wie Anmeldenamen und zugehörige Passwörter nur unverschlüsselt übertragen. Hierzu zählen beispielsweise Telnet, rsh und FTP.

Neben dem SSH-Protokoll existiert das gleichnamige Programm `ssh`, das als SSH-Client die Verbindung zu einem SSH-Server aufbauen kann. Dieses Programm spielt bei der Nutzung von VNUML eine große Rolle, da in der Standardkonfiguration Verbindungen vom Hostrechner zu einer virtuellen Maschine nur mittels SSH hergestellt werden können.

Als Parameter kann `ssh` die zu verwendende Version (SSHv1 bzw. SSHv2) sowie der Name bzw. die IP-Adresse des Rechners, mit dem die Verbindung hergestellt werden soll, übergeben werden. Standardmäßig wird eine Verbindung gemäß SSHv1 hergestellt.

```
ssh [-1|-2] Name|IP-Adresse
```

Eine Verbindung zum virtuellen Rechner "UML1" könnte wie folgt aufgebaut werden:

```
linux:/usr/local/share/vnuml/examples # ssh -1 10.0.0.1
uml1:~#
```

5.3 netstat

Mit Hilfe des Programmes `netstat` können Informationen zu bestehenden Verbindungen sowie ausführliche Protokollstatistiken angezeigt werden.

Die einfache Ausführung des Kommandos `netstat` auf der Kommandozeile liefert eine Statistik ähnlich der unten dargestellten. Dabei werden neben dem für eine Verbindung verwendeten Protokoll die eigene IP-Adresse sowie die der verbundenen Gegenstelle angezeigt. Zusätzlich werden die Nummern der benutzten Ports sowie der aktuelle Zustand der Verbindung ausgegeben.

Die folgende Ausgabe entstand unter Verwendung des Beispielnetzes aus Abbildung 1. Von der virtuellen Maschine "UML2" (IP 10.0.0.2) wurde dabei eine SSH-Verbindung zur virtuellen Maschine "UML1" (IP 10.0.0.1) aufgebaut und `netstat` auf "UML1" ausgeführt.

Zu erkennen sind zwei aktive (`State ESTABLISHED`) TCP-Verbindungen (`Proto tcp6`). Eine dieser Verbindungen besteht mit "UML2" an Port 1025 (`Foreign Adress 10.0.0.2:1025`), die andere mit dem Hostrechner an Port 47324 (`Foreign Adress 10.0.3.2:47324`).

Die unter `Active UNIX domain sockets` aufgeführten Angaben sind für die Arbeit mit Rechnernetzen weniger interessant; sie geben Auskunft über UNIX-interne Interprozesskommuniktionen.

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6       0      0 ::ffff:10.0.0.1:ssh     ::ffff:10.0.0.2:1025    ESTABLISHED
tcp6       0      0 ::ffff:10.0.0.1:ssh     ::ffff:10.0.3.2:47324   ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  5      [ ]         DGRAM                    1245     /dev/log
unix  2      [ ]         DGRAM                    1363
unix  2      [ ]         DGRAM                    1330
unix  2      [ ]         DGRAM                    1257
```

Wird `netstat` mit dem Parameter `-s` ausgeführt, wird eine detaillierte Statistik der einzelnen Protokolle angezeigt.

Unten abgebildet ist ein Ausschnitt der Protokollstatistik, die durch Ausführung von `netstat -s` auf "UML1" entstand. Exemplarisch dargestellt ist hier die Statistik des ICMP-Protokolls.

```
Icmp:
    4 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        echo replies: 4
    0 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
```

5.4 traceroute

`traceroute` ermittelt die Stationen, die ein IP-Paket auf seinem Weg von einem Quell- zu einem Zielrechner passiert hat.

Die Funktionsweise basiert dabei auf dem Versenden von IP-Paketen mit einer um jeweils eins erhöhten Paketlebensdauer (Time To Live, TTL). Die TTL des ersten Pakets hat den Wert eins. Diese wird vom ersten zu passierenden Router dekrementiert, so dass das Paket an dieser Stelle verworfen wird und der Router eine "time to live exceeded"-Nachricht an den Quellrechner sendet. Das nächste Paket wird dann mit einer TTL von zwei versendet, so dass es den ersten Router passiert und erst vom zweiten Router mit der entsprechenden Meldung verworfen wird. Dies wiederholt sich so lange, bis ein Paket den Zielrechner erreicht und von diesem eine "Host unreachable"-Nachricht an den Quellrechner geschickt wird. Anhand der so gesammelten Informationen ist der Quellrechner nun in der Lage, den Weg eines IP-Paketes zum Zielrechner zu rekonstruieren.

Für `traceroute` existieren zahlreiche grafische Oberflächen, die eine Visualisierung der Wegstrecken innerhalb eines Netzes erlauben.

Die unten abgebildete Ausgabe des `traceroute` Kommandos gibt den Weg eines Datenpaketes von der virtuellen Maschine "UML1" nach "UML5" an.

```
uml1:~# traceroute 10.0.2.2
traceroute to 10.0.2.2 (10.0.2.2), 30 hops max, 38 byte packets
 1  10.0.0.3 (10.0.0.3)  76.402 ms  0.831 ms  0.674 ms
 2  10.0.1.2 (10.0.1.2)  48.655 ms  1.239 ms  1.061 ms
 3  10.0.2.2 (10.0.2.2)  72.450 ms  1.602 ms  1.413 ms
uml1:~#
```

Der entsprechende Pfad wurde in der unten dargestellten Grafik nochmals veranschaulicht.

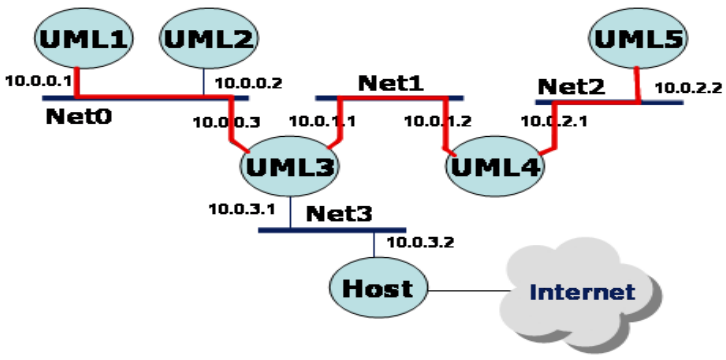

Abbildung 2: Weg eines Paketes von UML1 nach UML5

5.5 route

Zur Anzeige der Routingtabelle eines Gerätes kann das Kommando `route` verwendet werden. Alternativ ist auch der Einsatz von `netstat` mit dem Parameter `-r` möglich.

Angezeigt werden hier unter anderem folgende Informationen:

- Destination:
  Die Zieladresse des Netzes bzw. Rechners, an den ein Paket gesendet werden soll.
- Gateway:
  Die Adresse eines zu verwendenden Gateways bzw. *, falls kein Gateway benutzt werden soll.
- Genmask:
  Die Subnetzmaske des Zielnetzes.
- Flags:
  Verschiedene Flags, von denen die wichtigsten folgende sind:
  - U: Die eingetragene Route ist aktiv.
  - H: Das Ziel der Route ist ein einzelner Rechner.
  - G: Die Route zeigt auf ein ganzes Netz über ein Gateway.
- Metric:
  Gibt die Kostenmetrik einer Route an.
  Diese ist ausschlaggebend für die Auswahl einer Route, wenn mehrere mögliche Routen zur Weiterleitung eines Paketes existieren. In diesem Fall wird die Route mit dem niedrigsten Wert ausgewählt. Dieser spiegelt dabei Parameter wie beispielsweise die Anzahl zu passierender Knoten (Hops), die Zuverlässigkeit eines Pfades oder dessen Geschwindigkeit wieder.
- Iface:
  Angabe der zu verwendenden Netzwerkschnittstelle.

Die Beispielausgabe zeigt die Routingtabelle der virtuellen Maschine "UML1". Gemäß der Konfiguration des Beispielnetzes ist hier die VM "UML3" als Standardrouter ("default") eingetragen.

```
uml1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.144   *               255.255.255.252 U     0      0        0 eth0
10.0.0.0        *               255.255.255.0   U     0      0        0 eth1
default         10.0.0.3        0.0.0.0         UG    0      0        0 eth1
```

5.6 ifconfig

Mit Hilfe von `ifconfig` werden Informationen zu den in einem Rechner vorhandenen Netzwerkschnittstellen angezeigt.

Diese Informationen bestehen aus Daten wie MAC-, IPv4- und IPv6-Adresse, der Maximum Transfer Unit (MTU) sowie der Angabe, in welchem Modus die Netzwerkschnittstelle

aktuell arbeitet. Mögliche Betriebsmodi sind der so genannte "Promiscuous Mode", in dem das Interface alle im Netzwerk kursierenden Pakete empfängt sowie der "Multicast Mode", in dem die Schnittstelle alle an sie gerichteten Multicast-Pakete annimmt.

Der unten abgebildete Ausschnitt stammt aus einem Aufruf von`ifconfig` auf der VM "UML1". Insgesamt verfügt diese über drei Schnittstellen: dem Management Interface "eth0", der virtuellen Schnittstelle "eth1" sowie dem Loopback-Interface "lo", das standardmäßig über die IP-Adresse 127.0.0.1 verfügt.

Dargestellt wird hier die Konfiguration der Schnittstelle "eth1" mit der IPv4-Adresse 10.0.0.1.

```
eth1      Link encap:Ethernet  HWaddr FE:FD:00:00:01:01
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.255.255.0
          inet6 addr: fe80::fcfd:ff:fe00:101/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:239 errors:0 dropped:0 overruns:0 frame:0
          TX packets:317 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11998 (11.7 KiB)  TX bytes:23702 (23.1 KiB)
          Interrupt:5
```

5.7 arp

Das Adress Resolution Protocol (ARP) dient der Zuordnung von IP-Adressen zu Hardwareadressen.

Dabei ist das IP-Protokoll der Vermittlungsschicht des OSI-Referenzmodells (Schicht 3) zuzuordnen. Es erlaubt die logische Adressierung einzelner, in Subnetzen angeordneter Rechner über eine 32 (IPv4) bzw. 128 (IPv6) Bit lange Adresse, die aus einem Netz- und einem Hostanteil besteht.

Die Spezifikation des in der Sicherungsschicht (OSI Schicht 2) anzusiedelnden Ethernet-Protokoll sieht jedoch die Adressierung der in einem Netz vorhandenen Rechner bzw. deren Netzwerkschnittstellen über eine eindeutige Hardwareadresse, die so genannte MAC-Adresse, vor. Da keine feste Verbindung zwischen IP- und MAC-Adressen existiert, kommt das Adress Resolution Protocol (ARP) zur "Verknüpfung" dieser beiden Informationen zum Einsatz.

Möchte ein Quellrechner einen ihm nur anhand seiner IP-Adresse bekannten Rechner adressieren, nimmt er einen Broadcast vor. Hierbei wird ein Paket mit der Ziel-IP-Adresse an alle Rechner des Netzes gesendet ("ARP-Request"). Der Rechner, der seine IP-Adresse in diesem Paket "wiedererkennt", antwortet daraufhin mit einem "ARP-Reply"-Paket, in dem seine MAC-Adresse eingebettet ist. Der Quellrechner speichert diese Informationen zur späteren Wiederverwendung in einem Cache, dem so genannten ARP-Cache. Möchte er weitere Pakete an den Zielrechner senden, kann er diesen nun ohne vorherige Adressauflösung direkt über dessen MAC-Adresse adressieren.

Die Anzeige des ARP-Caches ist mit Hilfe des gleichnamigen Kommandos `arp` möglich.

Exemplarisch dargestellt ist hier der ARP-Cache von "UML1" nach dem "Anpingen" von "UML2", d.h. nachdem die MAC-Adresse der Netzwerkschnittstelle dieses Rechners dem ARP-Cache von "UML1" hinzugefügt wurde.

```
uml1:~# arp
Address                  HWtype  HWaddress           Flags Mask Iface
10.0.0.3                 ether   FE:FD:00:00:03:01   C          eth1
10.0.0.2                 ether   FE:FD:00:00:02:01   C          eth1
```

5.8 tcpdump

`tcpdump` erlaubt das Aufzeichnen in einem Netzwerk versendeter Datenpakete. Diese können entweder zur späteren Auswertung in eine Datei geschrieben oder direkt auf dem Bildschirm ausgegeben werden. Zahlreiche Filtermöglichkeiten erlauben ein Selektieren der auszuwertenden Datenpakete nach bestimmten Kriterien. Nützliche Optionen sind hierbei:

- `-c` *Anzahl*
  Dieser Parameter erlaubt die Angabe der Anzahl aufzuzeichnender Pakete. Ohne diese Option läuft das Programm so lange, bis es vom Anwender manuell abgebrochen wird. Dies geschieht in der Regel über die Tastenkombination STRG+c.

- `-i` *Interface*
  Verfügt ein Rechner über mehrere Netzwerkschnittstellen, kann es oftmals sinnvoll sein, die Überwachung des Netzwerkverkehrs auf ein über diesen Parameter anzugebendes Interface einzuschränken.

Die wichtigsten Filtermöglichkeiten sind:

- `dst host` *host*
  Mit diesem Filter werden nur die Pakete angezeigt, die an den als *host* angegebenen Rechner gesendet werden.

- `src host` *host*
  Genauso ist es möglich, nur die Pakete anzeigen zu lassen, die von dem durch *host* angegebenen Rechner an das zu überwachende Interface gesendet werden.

- `host` *host*
  Hiermit wird die Anzeige von Paketen auf solche beschränkt, die von dem durch *host* angegebenen Rechner gesendet bzw. empfangen werden.

- `'`*Pakettyp*`'`
  Möchte man nur Pakete eines bestimmten Pakettyps anzeigen lassen, kann dieser in einfachen Anführungszeichen eingeschlossen als Filterparameter übergeben werden.

Die unten dargestellte Ausgabe enstammt der Anwendung von `tcpdump` auf der virtuellen Maschine "UML1", die von "UML2" aus "angepingt" wurde. Gut zu erkennen ist hierbei die Arbeitsweise des im vorigen Abschnitt beschriebenen Adress Resolution Protocols: "UML2" stellt dabei einen "ARP-Request"(`arp who-has uml1 tell 10.0.0.2`), um die MAC-Adresse der VM "UML1" in Erfahrung zu bringen. Diese wird mit dem darauf folgenden "ARP-Reply" geliefert: `arp reply uml1 is-at fe:fd:00:00:01:01`. Anschließend folgt ein Dialog zwischen den beiden Rechnern mit ICMP-Paketen, ausgelöst durch die Verwendung von `ping`.

Die Ausgabe von `tcpdump` hängt vom jeweiligen Pakettyp ab, in der Regel liefert sie aber immer `Quelle > Ziel` sowie weitere, für den Pakettyp spezifische Details.

```
uml1:~# tcpdump -i eth1 -c 10
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
20:56:51.129912 IP uml1.ssh > 10.0.3.2.44378: P 2992043894:2992044058(164)
                ack 3212905795 win 1448 <nop,nop,timestamp 479704 2125024>
20:57:01.181252 IP 10.0.3.2.44378 > uml1.ssh: . ack 164 win 2264
                <nop,nop,timestamp 2125107 479704>
20:56:51.166271 IP uml1.1024 > 138.4.2.10.domain:
                52616+ PTR? 2.3.0.10.in-addr.arpa. (39)
20:56:53.485044 arp who-has uml1 tell 10.0.0.2
20:56:53.485182 arp reply uml1 is-at fe:fd:00:00:01:01
20:56:53.485878 IP 10.0.0.2 > uml1: icmp 64: echo request seq 1
20:56:53.497046 IP uml1 > 10.0.0.2: icmp 64: echo reply seq 1
20:56:54.488747 IP 10.0.0.2 > uml1: icmp 64: echo request seq 2
20:56:54.488910 IP uml1 > 10.0.0.2: icmp 64: echo reply seq 2
20:56:55.513709 IP 10.0.0.2 > uml1: icmp 64: echo request seq 3
10 packets captured
46 packets received by filter
0 packets dropped by kernel
```

6 Fazit

Mit VNUML wird dem Anwender ein mächtiges Werkzeug zur Simulation von Rechnernetzen an die Hand gegeben.

Dieses zeichnet sich durch eine einfach zu erlernende Syntax zur Beschreibung von Simulationsszenarien aus, so dass innerhalb kürzester Einarbeitungszeit die ersten Simulationen ablaufen können.

Durch die zahlreichen Konfigurationsmöglichkeiten der virtuellen Rechner und Netze ist der Phantasie beim Design eines Szenarios kaum Grenzen gesetzt.

Galt bisher oftmals die Installation als schwierigster Schritt bei der Arbeit mit VNUML konnte mit der Veröffentlichung des VNUML-Offline-Installers [3] hier Abhilfe geschaffen werden.

Die Möglichkeiten, die sich beim Einsatz von VNUML bieten, werden in den weiteren Arbeiten dieses Seminares ausführlich aufgezeigt.

A XML-Datei des Beispielnetzes aus dem VNUML-Tutorial

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.5.0</version>
    <simulation_name>tutorial</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
    <automac/>
    <ip_offset>100</ip_offset>
    <host_mapping/>
  </global>
  <net name="Net0" />
  <net name="Net1" />
  <net name="Net2" />
  <net name="Net3" />
  <vm name="uml1">
    <filesystem type="cow">/usr/local/share/vnuml/filesystems/
                           root_fs_tutorial
    </filesystem>
    <mem>50M</mem>
    <if id="1" net="Net0">
      <ipv4>10.0.0.1</ipv4>
    </if>
    <route type="inet" gw="10.0.0.3">default</route>
  </vm>
  <vm name="uml2">
    <filesystem type="cow">/usr/local/share/vnuml/filesystems/
                           root_fs_tutorial
    </filesystem>
    <if id="1" net="Net0">
      <ipv4>10.0.0.2</ipv4>
    </if>
    <route type="inet" gw="10.0.0.3">default</route>
  </vm>
  <vm name="uml3">
    <filesystem type="cow">/usr/local/share/vnuml/filesystems/
                           root_fs_tutorial
    </filesystem>
    <if id="1" net="Net0">
      <ipv4>10.0.0.3</ipv4>
```

```
    </if>
    <if id="2" net="Net1">
      <ipv4>10.0.1.1</ipv4>
    </if>
    <if id="3" net="Net3">
      <ipv4>10.0.3.1</ipv4>
    </if>
    <route type="inet" gw="10.0.1.2">10.0.2.0/24</route>
    <route type="inet" gw="10.0.3.2">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="uml4">
    <filesystem type="cow">/usr/local/share/vnuml/filesystems/
                           root_fs_tutorial
    </filesystem>
    <if id="1" net="Net1">
      <ipv4>10.0.1.2</ipv4>
    </if>
    <if id="2" net="Net2">
      <ipv4>10.0.2.1</ipv4>
    </if>
    <route type="inet" gw="10.0.1.1">default</route>
    <forwarding type="ip" />
  </vm>
  <vm name="uml5">
    <filesystem type="cow">/usr/local/share/vnuml/filesystems/
                           root_fs_tutorial
    </filesystem>
    <if id="1" net="Net2">
      <ipv4>10.0.2.2</ipv4>
    </if>
    <route type="inet" gw="10.0.2.1">default</route>
  </vm>
  <host>
    <hostif net="Net3">
      <ipv4>10.0.3.2</ipv4>
    </hostif>
    <route type="inet" gw="10.0.3.1">10.0.0.0/16</route>
  </host>
</vnuml>
```

Literatur

[1] VNUML Tutorial Version 1.5
`http://jungla.dit.upm.es/~vnuml/doc/1.5/tutorial/index.html`

[2] VNUML-Live-DVD
`https://www.uni-koblenz.de/~dickel/download/Knoppix-Live/`

[3] VNUML-Offline Installer
`https://www.uni-koblenz.de/~dickel/download/vnuml-offline`

[4] VNUML User Manual Version 1.5
`http://jungla.dit.upm.es/~vnuml/doc/1.5/user/index.html`

[5] VNUML Language Reference 1.5
`http://jungla.dit.upm.es/~vnuml/doc/1.5/reference/index.html`

[6] Arndt, Richard (2005): „VNUML-Virtual Network User Mode Linux“. Seminar Routing WS 04/05, Universität Koblenz-Landau.

[7] Naida Ettaous & Andrew Kiprop (2005): „Netzwerksimulation mit Virtual Network User Mode Linux-VNUML. Internet Message Control Protocol (ICMP)“. Projektpraktikum SS 05, Universität Koblenz-Landau.

[8] André Volk & Tim Keupen (2005): „Anleitung zur Installation des Netzwerk-Simulators VNUML“. Projektpraktikum SS 05, Universität Koblenz-Landau.