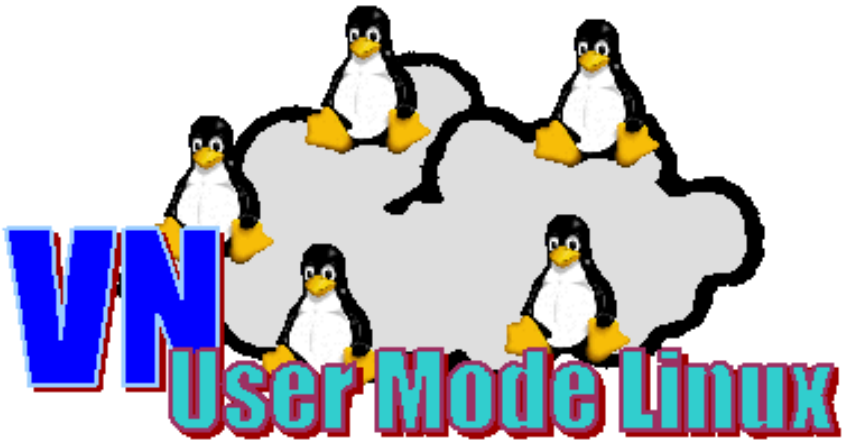


Anleitung zur Installation des Netzwerk-Simulators VNUML

SS 2005

André Volk
avolk@uni-koblenz.de

Tim Keupen
timbub@uni-koblenz.de

Universität Koblenz-Landau (Campus Koblenz)
Institut für Informatik
Prof. Dr. Christoph Steigner
31.08.2005

Inhaltsverzeichnis

1 Einleitung

Diese Ausarbeitung ist im Projektpraktikum “Routingsimulation“ im SS05 an der Universität Koblenz enstanden. Ziel war es, Hilfestellungen für den Einstieg und die Nutzung des Netzwerksimulators VNUML zu entwickeln.

Dazu wurden mehrere Installationen von VNUML auf Laborrechnern mit verschiedenen Linux-Distributionen experimentell durchgeführt. Die nötigen Arbeitsschritte und die Lösungswege bei auftretenden Problemen wurden protokolliert und kommentiert.

Im Verlauf des Projektpraktikums wurde deutlich, dass eine statische Installation von VNUML einige Nachteile aufweist: Zum einen verfügt sicherlich nicht jeder, der potenziell an der Nutzung von VNUML interessiert ist, über ein eigenes Linux-System. Zum andern lässt sich VNUML in der Version 1.5 nur mit Administrator-Rechten ausführen, die einem normalen Benutzer oft nicht zugänglich sind. Die Lösung dieser Probleme lag in der Erstellung einer bootfähigen Linux Live-DVD mit bereits installiertem VNUML.

Erfolgreich enstanden sind eine detaillierte Installationsanleitung um VNUML auf einem bestehenden Linux-System zu installieren, ein auf einem Shell-Script basierender Offline-Installer, eine Live-DVD basierend auf dem Betriebssystem Knoppix sowie umfangreiche Hilfetexte für die Benutzung von VNUML.

2 Was ist VNUML

„Virtual Network on User-Mode-Linux“ (VNUML) ist ein Tool zur Simulation von Computernetzen. Durch die Verwendung von User-Mode-Linux (UML) können virtuelle Rechner in einem Szenario erzeugt werden, die nahezu alle Eigenschaften eines realen Linux-Rechners haben. Dabei wird jeder virtuelle Rechner als ein Prozess (mit eigenem Filsystem etc) auf einem Rechner ausgeführt. Das ermöglicht je nach Performanz des Host-Rechners den Aufbau und Test fast beliebiger Netzwerktopologien, ohne die dazu erforderliche Hardware selbst zu besitzen. Die virtuellen Maschinen können beliebig konfiguriert werden und in einer Simulation verschiedene Rollen einnehmen (z.B. Router, Webserver, Client, etc).

Die Simulation besteht aus einer Folge 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 (hochgefahren) 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 Hardwareanforderungen) auf ihnen zu installieren, die dann im Laufe der Simulation ausgeführt werden können. Somit eignet sich VNUML hervorragend zum Testen von verteilter Software oder Netzwerksprotokollen. Der Simulator besteht aus zwei Komponenten

VNUML-Sprache Die XML-Sprache von VNUML dient dem Beschreiben von Szenarien. Eine XML-Datei beschreibt ein Szenario.

VNUML-Parser Der Parser besteht aus einem Perl-Skript welches das Aufbauen und Simulieren eines Szenarios ermöglicht. Die komplexen Details von UML werden vor dem Nutzer versteckt.

Bei allen Vorteilen muss an dieser Stelle jedoch auch erwähnt werden, dass die virtuellen Netze zwischen den UML-Rechnern stark abstrahiert werden. Performance-Messungen einer Netzwerktopologie durch Erzeugung von hohem Datenverkehr sind z.B. nicht möglich, da die Leitungen innerhalb der Netze überhaupt nicht modelliert werden. Da VNUML-Simulationen jedoch in Echtzeit auf einem Host laufen, könnten solche Aspekte auch nicht hinreichend simuliert werden, da der Grad des Echtzeitverhaltens stark von der Hardware des Hosts abhängt. [1]

Historie: VNUML wurde von Fermín Galán und David Fernández am Telematics Engineering Department (DIT) der technischen Universität von Madrid entwickelt. Entstanden ist VNUML dabei aus dem Euro6IX-Projekt 1, 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 Programm (vnumlparser-1.2.1 und DTD 1.2) wurde am 29. Juli 2003 veröffentlicht.

Am 12.08.2005 wurde die VNUML Version 1.6.0-1 als stable-version freigegeben. Diese Installationsanweisungen beziehen sich jedoch noch auf die Programm-Version 1.5.

Die offizielle Projekthomepage findet sich unter: *http://jungla.dit.upm.es/~vnuml/*

[1] www.uni-koblenz.de/ ∼ steigner/seminar-routingsim/arndt.pdf

3 Was ist User-Mode-Linux

„User-Mode-Linux“ (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-/Netzadministratoren.

Vieles wird 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“ (s.u.) 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. [2]

Wie ein ganz “normales“ Programm wird ein UML-Kernel gestartet sobald man ihn braucht, und kann ebenso leicht auch wieder beendet werden. Dabei läuft dieser Kernel im sogenannten „Userspace“, hat also nicht die volle Gewalt über das System. Dadurch hat der im Falle eines Fehlers oder Hacks entstehende Schaden so gut wie keine Auswirkungen auf das Hostsystem, also die GNU/Linuxumgebung in welcher der UML-Kernel gestartet wurde. Aufgrund dieser Tatsache werden z.B. sicherheitskritische Web-Services in UML Umgebungen eingebettet. Das bedeutet, dass z.b. Domain-, Mail- oder Webspaceserver als UML-Prozesse auf einem System laufen, durch die Kapselung der einzelnen Prozesse jedoch ein Hacker niemals das komplette Hostsystem blockieren kann, sondern allenfalls einen einzelnen Prozess darauf beeinflussen. Da dieser Prozess unabhängig neugestartet werden kann, spart man Wartungs- und Hardwareaufwand.

Vom Funktionsumfang lässt sich ein UML-Kernel vom echtem Betriebssystem-Kernel fast nicht unterscheiden. Einzig die Ausführung als neuer Prozess in einem bereits laufenden System bildet den Unterschied. Dabei arbeiten sowohl der Scheduler, als auch die Speicherverwaltung unabhängig vom Hostsystem. Dieses wird lediglich für die Hardwareunterstützung genutzt.

Anwendungs-Szenarien für User-Mode-Linux:

- **Virtuelles Hosting** - da es sich bei UML um einen vollwertigen Kernel handelt, ermöglicht es Hostingprovidern ein physisches System in mehrere, von einander unabhängige, virtuelle Maschinen aufzuteilen. So läuft z.B. die Communityseite usermodelinux.org unter UML (auf der Projekthomepage finden sich auch Links zu kommerziellen Betreibern).
- **Sandbox oder Jail** - als Sandbox- oder Jail-Strukturen bezeichnet man Programme die, um Schäden zu minimieren, isoliert vom Betriebssystem laufen. Prozesse die unter UML laufen haben keinen Zugriff auf das Hostsystem. So könnte man z.B. einen sicherheitskritischen Dienst unter UML

[2] http://de.wikipedia.org/wiki/User_Mode_Linux

laufen lassen. Sollte er tatsächlich gehackt werden, nimmt lediglich das UML-System Schaden.

- **Testsystem** - man kann ohne viel Aufwand und Risiko einen neuen Kernel testen, oder sich z.B. davon überzeugen, ob der Rechner mit der neuen optimierten Konfiguration auch booten würde.
- **Softwareentwicklung** - der UML-Kernel innerhalb der virtuellen Maschine kann anders als das Hostsystem konfiguriert werden. Dies versetzt Softwareentwickler in die Lage, Software unter geänderten Bedingungen zu entwickeln und zu testen. Entwickler von netzwerkbasierten Programmen können ein virtuelles Netzwerk auf einem einzigen Rechner aufziehen, um ihre Applikation kostenschonend zu testen.
- **Linuxemulation** - noch läuft UML nur unter Linux. Würde man sich eine Portierung auf andere Betriebssysteme vorstellen, würde dies bedeuten, dass Programme die für GNU/Linux geschrieben wurden, dort nativ laufen könnten.
- **Honeypot** - da es sich bei User Mode Linux um einen voll funktionalen Linuxkernel handelt, eignet es sich auch als sogenannter *Honeypot*, also ein System welches bewusst angreifbar gemacht wird, um somit Auskunft über die Strategien der Angreifer zu liefern. Durch die Besonderheiten von UML wird es möglich eine Vielzahl an virtuellen Honeypots auf einem physischen System laufen zu lassen und dabei sogar noch ein komplettes Netzwerk zu simulieren. So gibt es zum Beispiel die Möglichkeit alle Tastatureingaben innerhalb von UML auf dem Hostsystem aufzuzeichnen. Da keinerlei Netzwerktraffic oder dergleichen verursacht wird, ist dies aus dem UML-System (für potentielle Angreifer) nicht erkennbar.

Dies ist natürlich keine erschöpfende Aufzählung aller denkbaren Anwendungen und Fähigkeiten von User Mode Linux. Als weitere Anregung sei angemerkt: man kann sogar in einem UML System einen weiteren UML-Kernel erstellen und starten! [3]

[3] Einführung in User Mode Linux, Cargal 2004

4 Installation

4.1 Systemvoraussetzungen

VNUML lädt für jeden Teilnehmer im virtuellen Netzwerk (Router oder Hosts) eine virtuelle Maschine mit einem echten Linux-Kernel und einem eigenen Filesystem in den Hauptspeicher.

Daher sollte neben einem einigermaßen schnellen Prozessor (ca. 500MHz+) auf jeden Fall ausreichend Hauptspeicher (256MB+) vorhanden sein um mit VNUML arbeiten zu können.

Für eine statische Installation werden außerdem ca. 500MB freien Speicherplatzes auf der Festplatte benötigt.

4.2 Welche Installations-Möglichkeiten gibt es

Es gib verschiedene Wege, um an eine lauffähige VNUML-Installation zu gelangen. Die Wahl des richtigen Weges hängt von den gegebenen Systemvoraussetzungen und den Ansprüchen des Benutzers an VNUML ab.

4.2.1 VNUML integriert in ein KNOPPIX Live-System

Um einen ersten Eindruck von VNUML zu bekommen oder wenn ein Linux System nicht fest auf einem Rechner installiert werden kann/soll, dann kann VNUML von der bootbaren KNOPPIX Live-DVD gestartet werden.
Eine feste Installation ist in diesem Falle nicht nötig und die Daten auf der Festplatte bleiben für das Linux-System unzugänglich. Jedoch benötigt der Rechner einen ausreichend großen Arbeitsspeicher (empfohlen: 512MB+).
Weitere Informationen dazu im Kapitel 5.

4.2.2 Statische Installation

Wenn bereits ein lauffähiges Linux-System vorliegt, dann ist eine statische Installation von VNUML angeraten.
Das Arbeiten an einem fest installierten VNUML ist komfortabler und resourcenschonender (mit Ausnahme des benötigten Festplattenplatzes). Die statische Installation ist für Nutzer, die mit VNUML intensiv arbeiten wollen unerlässlich.

Welche statischen Installationen von VNUML können durchgeführt werden?

- Automatische Installation mit dem VNUML Offline Installer siehe Abschnitt 6 (**dringend empfohlen!**)
- Manuelle Step-by-Step Installation (meist sehr langwierig und fehleranfällig)

5 KNOPPIX 3.8.2 Live-DVD mit VNUML

5.1 Was ist KNOPPIX

„KNOPPIX“ ist eine komplett von CD/DVD lauffähige Zusammenstellung von GNU/Linux-Software mit automatischer Hardwareerkennung und Unterstützung für viele Grafikkarten, Soundkarten, SCSI-Geräte und sonstige Peripherie. KNOPPIX wurde von Klaus Knopper (knopper.net) entwickelt und erstellt und kann als Linux-Demo, Schulungs-CD, Rescue-System oder als Plattform für kommerzielle Software-Produktdemos angepasst und eingesetzt werden. Es ist keinerlei Installation auf Festplatte notwendig. Trotzdem bietet KNOPPIX vollen Betriebssystemumfang, von Netzwerktreibern bis hin zum OpenOffice1.1 Paket ist alles vorhanden. Da es sich um ein echtes Open Source Projekt handelt, sind die Quelltexte zu den KNOPPIX-spezifischen Paketen, die der GNU General Public License unterliegen, im Internet erhältlich unter
http://www.knoppix.net

Dieses Betriebssystem bootet komplett von DVD, außer einer Erstellung der benötigten RamDisk wird an dem laufenden System nichts verändert. Partitionstabellenänderungen oder allgemeine Schreibzugriffe in das vorhandene Dateisystem werden im normalen Betrieb nicht ausgeführt, das System kann also bedenkenlos auf jedem Rechner gestartet werden. Der Knoppix-Kernel würde sogar ohne Festplatte auskommen und somit komplett aus dem Hauptspeicher heraus laufen. Aufgrund dieser Tatsache lässt sich die Perfomanz natürlich nicht mit einem normalen Betriebssystem messen, da alle Module ständig von CD/DVD nachgeladen werden müssen. Das Hauptfeature von KNOPPIX liegt in der automatischen Hardwareerkennung, die bei jedem Systemstart ausgeführt wird, diese ermöglicht es, dass Knoppix auf 95% aller heute gängigen PCs bootet.

Mittlerweile bieten auch viele (kommerzielle) Hersteller ihre Produkte in „Live“ Versionen an. Weitere gängige Linux-Distributionen, die nach dem gleichen Schema arbeiten sind: Kanotix, Ubuntu, Mandriva-Move, Gentoo-Live und Suse-Live.

5.2 Systemvorraussetzungen

- DVD Laufwerk
- 512 MB Hauptspeicher (und ggfs eine Linux Swap-Partition oder Swap-Datei)

5.3 KNOPPIX Boot-Parameter

Im Normalfall startet Knoppix ohne spezielle Konfiguration auf fast jedem System von alleine. Gibt es Probleme oder Anpassungswünsche lassen sich sog. Cheatcodes verwenden, um KNOPPIX mit bestimmten Werten zu starten (damit man auch z.B. schwierige Hardware zum Laufen bekommt). Sie werden am Boot-Prompt (links-unten) eingegeben und mit ENTER/RETURN bestätigt. Das Format sieht folgendermaßen aus: „kernel opt opt opt...“, also zum Beispiel: *knoppix xvrefresh=60 noscsi floppyconfig* oder *knoppix nodhcp.*
Die wichtigsten Parameter sind:

lang=bg|ch|cn|cs|da|de|dk|es|fr|uk|us Einstellen des Keyboard-Layouts und der Sprache

screen=1280x1024 Benutzerdefinierte Auflösung für X einstellen

noapic noagp nodma nopcmcia noscsi noswap nousb Ausschalten einiger automatischer Hardware-Suchfunktionen

nodhcp DHCP/Network broadcast detection überspringen (im Uni-Netz oft nötig)

toram Knoppix komplett in den Hauptspeicher laden (mind. 1GB), keine Festplatte nötig

Alle verfügbaren Parameter sind hier aufgelistet: *http://www.knoppix.net/wiki/Cheat_Codes_Deutsch*

5.4 Schnellstartanleitung

Diese Anleitung bezieht sich auf das Arbeiten mit VNUML unter der KNOPPIX Live-DVD.

- Die Live-DVD ist als ISO-Image unter *https://www.uni-koblenz.de/~steigner/download/* verfügbar.
- ISO-Image (992MB) auf eine DVD-R oder DVD-RW brennen und den Rechner von dieser DVD booten.

Nach dem erfolgreichen Start von Knoppix landet man auf dem KDE- Desktop. Im Startmenü unter „KNOPPIX ⇒ Konfiguration ⇒ Swap-Datei Konfiguration“ befindet sich ein Tool zum automatischen anlegen einer Linux-Swap-Datei (empfohlen auch bei 512MB-RAM, jedoch nicht dringend erforderlich), hier am besten 128MB angeben.

1. Schritt: das Terminal Fenster (shell) öffnen (dazu auf den schwarzen Monitor in der Taskleiste klicken)
2. Schritt: als root anmelden: *su* [enter]
3. Schritt: statisches user-mode-linux Verzeichnis anlegen:
   mkdir /mnt/uml [enter]
4. Schritt: RSA-Schlüsselpaar zur Authentifizierung anlegen:
   ssh-keygen -t rsa1 [enter]
   Jetzt die nächsten 3 Abfragen mit [enter] bestätigen
5. Schritt: ins Verzeichnis der Beispieldateien wechseln:
   cd /usr/local/share/vnuml/examples [enter]
6. Schritt: VNUML testen: *vnumlparser.pl* [enter] zeigt die Parameter-Liste des VNUML-Parsers
7. Schritt: Szenario starten: *vnumlparser.pl -t tutorial.xml -vB* [enter]
8. Schritt: Simulationen starten oder beliebige Befehle ausführen (nachzulesen in den entsprechenden Manuals), z.B.:
   - einloggen auf der virtuellen Maschine uml1: *ssh -1 uml1* [enter]
   - ping an 10.0.0.2: *ping 10.0.0.2* [enter]
   - ping beenden mit [strg+c]
   - ausloggen aus uml1: *exit* [enter]
   - etc...
9. Schritt: Szenario herunterfahren *vnumlparser.pl -d tutorial.xml -v* [enter]

Beim Starten eines Szenarios werden eine Menge Informationen auf der Kommandazeile ausgegeben. Bis alle virtuellen Maschinen per sshd erreichbar sind, können mehrere Minuten vergehen. Anstelle von „tutorial.xml“ kann hier natürlich jedes beliebige Beispielszenario gestartet werden. Grafiken der gestarteten Netzwerktopologien finden sich in den entsprechenden Ausarbeitungen. (z.B. für „tutorial.xml“ auf der VNUML-Homepage: *http://jungla.dit.upm.es/~vnuml/doc/1.3/examples/tutorial/tutorial.html*)

5.5 KNOPPIX permanent auf Festplatte installieren

Möchte man den Inhalt der KNOPPIX Live-DVD dauerhaft auf der Festplatte ablegen, bringt das System dafür schon ein geeignetes Skript mit. Nach der erfolgreichen Installation lassen sich die Performanz-Vorteile eines Betriebssystems von Platte stark erkennen. Der Hauptspeicher ist nichtmehr belegt durch das System selbst, sondern bleibt den Benutzeranwendungen vorbehalten.

1. Desktop-Icon „HD-Install“ anklicken, bzw in der shell: *knoppix-installer* (als root)
2. Menüpunkt 1 „Erstellen einer neuen Konfiguration“ auswählen
3. Installationsart „Knoppix System wie von CD“ und das passende Laufwerk auswählen (das Laufwerk muss hierbei ein Dateisystem enthalten das Knoppix bekannt ist)
4. Jetzt im Installations-Hauptmenü den Punkt „Installation starten“ anwählen
5. Nach erfolgreicher Installation System ohne DVD vom entsprechenden Laufwerk booten

Das Betriebssystem entspricht nun dem der Live-DVD. Änderungen aller Art - vom Erscheinungsbild bis hin zu Systempaketen - bleiben jedoch dauerhaft erhalten.

Möchte man das System nicht dauerhaft auf Festplatte haben, aber trotzdem länger damit arbeiten und Änderungen und Einstellungen nicht jedesmal beim Neustart verlieren, stellt Knoppix zum Abspeichern von persönlichen Daten und Systemdaten zwei verschiedene Ansätze zur Verfügung:

- Das **persistente** (sprich: feste) **Heimverzeichnis**. Hierbei wird das Heimverzeichnis des Standardbenutzers „knoppix“ auf einem externen Datenträger wie z. B. einem USB-Memorystick abgelegt.
- Die **Sicherung** der **Konfigurationsdateien**. Dabei werden diejenigen Änderungen gesichert, die der Benutzer mit Hilfe der Knoppix Konfigurationswerkzeuge vorgenommen hat.

5.5.1 Ein festes Heimverzeichnis erstellen

Dazu öffnet man im Startmenü unter „KNOPPIX ⇒ Konfiguration ⇒ Permanentes KNOPPIX-Image einrichten“ den Link zu einem Installationsskript. Mit dem Boot-Parameter *home=scan* sucht die Live-DVD automatisch beim Start nach einem solchen Image und verwendet dieses unter dem Standard-User „knoppix“. Beim Booten erscheint nun die Meldung, dass ein permanentes Heimverzeichnis gefunden wurde und eingebunden wird. Die entsprechende Meldung muss dazu mit OK bestätigt werden. Eine Schritt-für-Schritt Anleitung, insbesondere für die Speicherung auf einem externen Medium wie USB-Sticks, findet sich in der Quelle. [4]

[4] http://www.galileocomputing.de/openbook/knoppix/knoppix20.htm

5.5.2 Konfiguration des Systems speichern

Die zweite wichtige Anwendung des Speicherns von Daten besteht darin, bereits vorgenommene Konfigurationen wie z. B. die Netzwerkeinstellungen auf einem Datenträger permanent zu sichern. Die Vorgehensweise gestaltet sich ähnlich einfach wie bei der Erstellung des Heimverzeichnisses im vorangegangenen Abschnitt. Nachdem das System fertig eingerichtet ist, öffnet man im Startmenü unter „KNOPPIX ⇒ Konfiguration ⇒ KNOPPIX-Konfiguration speichern“ das dafür mitgelieferte Skript. Im sich darauf öffnenden Dialog werden einige Komponenten angegeben, die sich zum Abspeichern anbieten. Im nächsten Schritt ist das Speichermedium auszuwählen. Das kann z.B. eine Diskette oder ein Memorystick sein. Schließlich speichert das Skript die Einstellungen ab. Diese können beim nächsten Booten mit Hilfe des Bootparameters *knoppix myconfig=scan* wieder hergestellt werden. Soll das persistente Heimverzeichnis ebenfalls rückgesichert werden, so sind beide Bootparameter zu kombinieren.

6 VNUML Offline Installer

VNUML setzt auf einer großen Zahl von Programmpaketen und Modulen auf. Diese Pakete und Module gilt es vor der Installation von VNUML zu ermitteln und nacheinander in der richtigen Reihenfolge zu installieren.
Das von den VNUML Entwicklern gelieferte Installationsscript soll diese Arbeit zwar übernehmen, jedoch zeigte sich das Script während unserer Tests als sehr unvollständig und fehlerhaft.
Wir haben alle für die Installation von VNUML nötigen Pakete und Module gesammelt und in einem Installationspaket mit der Bezeichnung **VNUML Offline Installer** gebündelt. Die zahlreichen Arbeitsschritte zur Installation und Konfiguration übernimmt ein Shell-Script (bash), das mit möglichst wenig Benutzereingaben auskommen sollte.
Der VNUML Offline Installer benötigt, anders als das offizielle VNUML Installationsscript, keine Internetverbindung. Dabei muss jedoch beachtet werden, dass die Pakete und Module im VNUML Offline Installer evtl. nicht in den aktuellsten Versionen vorliegen. Eine Aktualisierung des Offline Installers ist jedoch ohne großen Aufwand möglich.
Das komplette VNUML Offline Installer Paket ist 240 MB groß und kann unter *https://www.uni-koblenz.de/~steigner/download* heruntergeladen werden.

6.1 Vorraussetzungen für jede statische Installation von VNUML

Bei Standard-Installationen verschiedener Linux-Distributionen waren essentielle Linux-Programmpakete häufig nicht vorinstalliert. Zu Beginn der Installation prüft der Offline-Installer ob folgende Linux-Pakete auf dem Linux-System installiert sind:

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

Sollte eines dieser Pakete nicht installiert sein, so bricht die Installation mit einer erklärenden Meldung ab. Das fehlende Programmpaket muss in diesem Falle erst manuell vom Benutzer installiert werden. Unter SuSE Linux kann dies z.B. im Paketmanager *YAST* passieren, unter Debian verwendet man üblicherweise den Befehl `apt-get install name-des-pakets`. Der Paketmanager fordert dann zur Eingabe des Installationsmediums (z.B. SuSE Installations-CD) auf oder lädt die nötigen Dateien aus dem Internet.

6.2 Offline-Installation

Folgende Arbeitsschritte sind nötig, um VNUML auf einem bestehenden Linux System zu installieren:

1. Schritt: Offline Installer Paket (*VNUML-Offline.zip*) aus dem Internet herunterladen
2. Schritt: Entpacken der Datei *VNUML-Offline.zip*
3. *Optionaler Schritt:* Die entpackten Dateien können auf eine CD gebrannt werden. Die Installation kann dann direkt von CD gestartet werden.
4. Schritt: Öffnen einer Linux Shell (Konsole, Eingabeaufforderung)
5. Schritt: Mit dem Befehl `su` als Benutzer *root* anmelden
   (Für die Installation werden Administrator-Rechte benötigt)
6. Schritt: Mit dem Befehl `cd verzeichnisname` in das Verzeichnis wechseln, in dem das Offline Installer Paket entpackt wurde (bzw. auf die Offline Installations CD).
7. Schritt: `./install` eingeben (die genaue Schreibweise beachten!!)
8. Schritt: Die Hinweise, die der Installer am Bildschirm ausgibt, geben Auskunft über evtl. auftretende Probleme oder beschreiben Handlungsanweisungen für den Benutzer.

6.3 Die uml-utilities

Die *uml-utilities* sind ein problematisches Programmpaket, das für die Nutzung von VNUML unbedingt erforderlich ist.
Wenn VNUML auf einem Rechner mit SuSE Linux in der Version 9.X installiert werden soll, so wird der Offline Installer die *uml-utilities* vermutlich ohne Probleme automatisch installieren können.
Ebenso wird eine VNUML Installation auf einem Linux System mit funktionierendem *Advanced Packaging Tool* (apt) unproblematisch verlaufen.
Sollte die VNUML Offline Installation jedoch mit der Meldung abbrechen, man solle das Paket *uml-utilities* manuell installieren, so gilt folgendes:

- Suchen des Paketes *uml-utilities* für die eingesetzte Linux-Distribution. Dies kann entweder über einen Paketmanager erfolgen (unter SuSE Linux z.B. mit YAST) oder im Internet.
- Leider kann das Paket *uml-utilities* laut Aussagen der Entwickler[5] nur sehr umständlich aus dem Quellcodepaket installiert werden, daher ist ein speziell für die jeweilige Linux-Distribution vorkompiliertes Paket zu bevorzugen.
- Für SuSE, Mandrake, Fedora oder RedHat Linux gibt es sogenannte RPM-Pakete. Ein guter Anlaufpunkt um RPM Pakete zu finden ist z.B.:
  http://rpm.pbone.net/
  oder Google *(http://www.google.de/)*

[5] http://user-mode-linux.sourceforge.net/

- Wenn das entsprechende RPM Paket (*uml-utilitiesXXX.rpm*) vorliegt, kann es mit dem Befehl
  `rpm -i uml-utilitiesXXX.rpm` installiert werden.

Wenn die *uml-utilities* manuell installiert wurden, muss der Offline Installer mit `./install` (siehe Abschnitt 6.2) erneut gestartet werden.

6.4 Fehlersuche

Um den genauen Verlauf der Installation (und auch evtl. übergangene Fehlermeldungen) ansehen zu können, kann man die VNUML Offline Installation mit dem Befehl `./install --debug` starten. Das Programm stoppt in diesem Fall nach jedem Schritt, damit man die Bildschirmausgabe besser verfolgen kann.

Generell gilt: Wenn ein Fehler auftritt und die Installation stoppt, dann sollte zuerst versucht werden den Fehler zu beheben (z.B. mit Hilfe der Lösungsvorschläge, die die Installation am Bildschirm ausgibt, oder mit Hilfe dieser Installationsanleitung). Eine Auflistung von häufig aufgetretenen Fehlern ist im **Anhang D** zu finden.
Wenn der Fehler beseitigt wurde, muss die Installation mit `./install` neu gestartet werden. Die Installation kann erst dann vollständig sein, wenn der Glückwunsch-Bildschirm des Installations-Programms erscheint.

7 Manuelle Step-by-Step Installation

7.1 Allgemeines

Ab VNUML Version 1.4 wird von den VNUML-Entwicklern ein Installationsscript mitgeliefert, das die Installation von VNUML weitgehend automatisieren soll. In der Praxis hat sich jedoch gezeigt, dass diese Installation nur selten unproblematisch verläuft.
Will man also VNUML statisch auf einem Linux-System installieren, so muss man eine Reihe von Arbeitsschritten manuell ausführen, was mitunter sehr viel Zeit in Anspruch nehmen kann.
Das unter Abschnitt 6 beschriebene Offline Installer Paket arbeitet systematisch die nötigen Installationsschritte ab und soll damit den Anwender entlasten.
Die folgende Beschreibung einer manuellen Step-by-Step Installation ist somit als tieferer Einblick in die Arbeitsweise des Offline Installers zu verstehen und kann zu evtl. auftretenden Problemen Aufschluss und Hilfestellung geben.
Die Nutzung des Offline Installers ist jedoch in jedem Fall zu bevorzugen.

7.2 Welche Programmpakete werden benötigt

Die folgenden Programmpakete werden für die Installation oder die Benutzung von VNUML benötigt. Der Offline Installer liefert all diese Pakete direkt mit.

7.2.1 Linux-Pakete

Unter *Linux-Paketen* oder einfach *Paketen* verstehen wir eigenständige Linux-Anwendungen von Drittanbietern, die von VNUML benötigt werden.
Für eine SuSE-Linux Distribution in der Version 9.X werden diese Pakete aus sogenannten RPM-Dateien installiert. Es handelt sich dabei um vorkompilierte Programmpakete, die der SuSE Paketmanager YAST verwalten kann.
Das RPM Format wird auch von anderen Linux-Distributionen verwendet (z.B. Mandrake, Fedora oder RedHat), jedoch muss eine speziell für die jeweilige Distribution vorkompilierte Variante des RPM-Pakets vorliegen, damit das Linux-Paket installiert werden kann.
Es gibt auch die Möglichkeit, Linux-Pakete aus dem Quellcode heraus zu installieren. Diese sind weitgehend unabhängig von der eingesetzten Linux-Distribution.
Darum bringt der VNUML Offline Installer neben den SuSE-Linux 9.X RPM Paketen auch noch die Quellcode Pakete der jeweiligen Linux-Pakete mit (mit Ausnahme der *uml-utilities* - siehe Abschnitt 6.3).

Linux-Pakete in RPM Format	Linux-Pakete im Quellcode
readline-5.0-1.2.i586.rpm	readline-5.0.tar.gz
libxml2-2.6.12-3.4.i586.rpm	libxml2-2.6.19.tar.gz
expat-1.95.8-2.i586.rpm	expat-1.95.8.tar.gz
bridge-utils-1.0.4-2.i586.rpm	bridge-utils-1.0.6.tar.gz
uml-utilities-20040114-21.1.i586.rpm	

7.2.2 Perl Module

Perl Module sind Sammlungen von Perl-Scripten, die verschiedene Routineaufgaben erledigen sollen und dadurch die Programmierung in Perl erleichtern und beschleunigen.
VNUML benötigt eine ganze Reihe solcher Module, die allesamt im VNUML Offline Installer enthalten sind:

- Compress-Zlib-1.34
- libxml-perl-0.08
- Math-Base85-0.2
- Net-IPv4Addr-0.10
- Net-IPv6Addr-0.2
- TermReadKey-2.30
- XML-RegExp-0.03
- URI-1.35
- HTML-Tagset-3.04
- libwww-perl-5.803
- XML-DOM-1.43
- XML-Checker-0.13
- HTML-Parser-3.45
- XML-Parser-2.34
- Class-Data-Inheritable-0.02
- Devel-StackTrace-1.11
- Exception-Class-1.21
- Module-Build-0.2611
- Error-0.15
- NetAddr-IP-3.24

7.2.3 Linux-Kernel und Filesystem

Die virtuellen Maschinen, die VNUML startet, benötigen einen echten Linux-Kernel (Basis Linux System) und ein Filesystem. Bei dem Linux-Kernel handelt es sich um die Datei „linux“, bei dem Filesystem handelt es sich um die Datei „root-fs-tutorial“. Beide Dateien sind im Offline Installer enthalten oder auf der VNUML Project Homepage [6] zu finden.

7.2.4 VNUML Installationsdateien

Die Basis von VNUML bildet ein Parser, der die VNUML XML-Dateien (Netzwerkszenarien) parst, mit Hilfe von UML (User Mode Linux) virtuelle Maschinen startet und zwischen diesen alle gewünschten (virtuellen) Netzwerkverbindungen herstellt.
Das Basispaket der VNUML Version 1.5 ist in leicht modifizierter Form im VNUML Offline Installer integriert.
Man findet es jedoch auch auf der VNUML Project Homepage. Das VNUML Installationspaket trägt den Dateinamen *vnuml-1.5.0-1.tar.gz*.

7.3 Verlauf

Die Diagramme in **Anhang B** zeigen die VNUML Installation in einem Flussdiagramm.

7.3.1 Prüfung

In der einleitenden Prüfungsphase wird sichergestellt, dass folgende essentielle Linux-Pakete installiert sind:

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

Sollte das von den VNUML Entwicklern gelieferte Installations-Script verwendet werden und nicht der Offline Installer, dann sollten auch noch folgende Programmpakete installiert werden:

- Paket *lynx*
- Paket *ncftpget*

Diese Pakete sind für Voraussetzung für den Einsatz von CPAN (siehe Abschnitt 7.4)

Außerdem muss in der Prüfungsphase sichergestellt werden, dass der Benutzer über Administrator-Rechte verfügt.

[6] http://jungla.dit.upm.es/~vnuml/

7.3.2 Kopieren

In der Kopierphase kopiert der VNUML Offline Installer solche Installationsdateien in ein temporäres Verzeichnis auf der Festplatte, die z.B. während des Kompiliervorgangs Schreibzugriff auf die Festplatte benötigen.
Konkret sind das die **PerlModule** (siehe Abschnitt 7.2.2), die **Linux-Pakete** im Quellcode (siehe Abschnitt 7.2.1) und die eigentlichen **VNUML Installationsdateien** (siehe Abschnitt 7.2.4)

Im Anschluss daran werden die Dateien für den Linux Kernel und das Filesystem an die richtige Stelle im System kopiert:

Bezeichnung	**Dateiname**	**Verzeichnis**
Linux Kernel	linux	*usr/local/share/vnuml/kernels*
Filesystem	root_fs_tutorial	*usr/local/share/vnuml/filesystems*

7.3.3 Linux Pakete installieren

Bevor der VNUML Offline Installer die Linux Pakete aus Abschnitt 7.2.1 installiert, wird überprüft, ob es sich bei dem verwendeten Linux-System um eine SuSE Distribution in der Version 9.X handelt oder nicht. Sollte es sich um eine SuSE Linux 9.X Distribution handeln, werden die Linux-Pakete in der oben angegebenen Reihenfolge aus den vorkompilierten **RPM-Paketen** installiert.

Um RPM Pakete zu installieren benutzt man üblicherweise den Befehl:
`rpm -i name-des-pakets.rpm`

Sollte keine SuSE Linux 9.X Distribution vorliegen, installiert der VNUML Offline Installer die Linux Pakete aus den Quellcode Paketen.

Um Linux-Pakete aus dem Quellcode zu installieren, geht man üblicherweise wie folgt vor:

1. Schritt: Entpacken des Quellcodepakets mit dem Befehl
   `tar -xzvf name-des-pakets.tar.gz`
2. Schritt: In das beim Entpacken neu angelegte Verzeichnis wechseln
   `cd name-des-verzeichnisses`
3. Schritt: Als Administrator anmelden mit dem Befehl `su`
   und dem Administrator-Passwort
4. Schritt: Aufruf des Befehls `./configure`
5. Schritt: Aufruf des Befehls `make`
6. Schritt: Aufruf des Befehls `make install`

Das Paket „uml-utilities“ kann nur sehr kompliziert aus dem Quellcode installiert werden. Besser ist hier die Verwendung eines vorkompilierten Installationspaketes (siehe Abschnitt 6.3).

7.3.4 Perl Module installieren

Bei der Installation von Perl Modulen kommt es oft vor, dass ein Perl Modul von verschiedenen anderen Modulen abhängig ist. Bei der Installation der Perl Module muss also auf die genaue, in Abschnitt 7.2.2 angegebene Reihenfolge geachtet werden.

Ein Perl Modul lässt sich mit folgender Befehlsfolge installieren:

1. Schritt: Entpacken des Perl Modul Archivs mit dem Befehl
   `tar -xzvf name-des-pakets.tar.gz`
2. Schritt: In das beim Entpacken neu angelegte Verzeichnis wechseln
   `cd name-des-verzeichnisses`
3. Schritt: Als Administrator anmelden mit dem Befehl `su`
   und dem Administrator-Passwort
4. Schritt: Aufruf des Befehls `perl Makefile.PL`
5. Schritt: Aufruf des Befehls `make`
6. Schritt: Aufruf des Befehls `make install`
7. Schritt: Aufruf des Befehls `make test`

Nach dem Aufruf des Befehls `make test` wird das gerade installierte Perl Modul automatisch auf Korrektheit und Funktionstüchtigkeit geprüft. Dabei muss die angegebene Erfolgsquote nicht unbedingt 100% erreichen.

7.3.5 VNUML installieren

Das eigentliche Installationspaket von VNUML (siehe Abschnitt 7.2.4 wurde im VNUML Offline Installer leicht modifiziert. Hier wurde der Inhalt der Datei „sbin/required.pl“ gelöscht, damit das Programm CPAN nicht ausgeführt werden kann. Nähere Informationen zu CPAN sind unter Abschnitt 7.4 zu finden.

Der VNUML Offline Installer geht in dieser Phase wie folgt vor:

1. Schritt: Anlegen eines Verzeichnisses mit dem Befehl
   `mkdir -p /usr/local/share/doc/vnuml/html`.
   Dieses Verzeichnis wird in dem ursprünglichen VNUML Installationspaket nicht korrekt angelegt, wodurch die Installation mit einer Fehlermeldung abbrechen würde.

2. Schritt: Wechseln in das Verzeichnis mit den modifizierten VNUML Installationsdateien `cd VNUML`
3. Schritt: Als Administrator anmelden mit dem Befehl `su` und dem Administrator-Passwort
4. Schritt: Aufruf von `./configure --without-expat --without-libxml`
5. Schritt: Aufruf von `make`
6. Schritt: Aufruf von `make install`
7. Schritt: Anlegen eines SSH-Keypair mit dem Befehl: `ssh-keygen -t rsa1`
8. Schritt: Eintrag des Verzeichnisses `/usr/local/bin` in den Systempfad
9. Schritt: Test des VNUML Parsers durch den Aufruf `/usr/local/bin/vnumlparser.pl -H`

Wenn die Installation bis hierhin fortgeschritten ist, dann ist VNUML einsatzbereit. Eine ausführliche Beschreibung zur Anwendung von VNUML ist ab Kapitel 8 zu finden.

7.4 CPAN

CPAN (Comprehensive Perl Archive Network) ist ein Programm, mit dem Perl Module von speziellen CPAN Servern im Internet heruntergeladen und automatisch installiert werden können. Dabei sollen eventuelle Abhängigkeiten zwischen verschiedenen Perl Modulen automatisch aufgelöst werden.
Leider ist die Benutzung von CPAN ohne eine Internetverbindung nicht möglich und es traten durchaus auch auf manchen Systemen mit Internetverbindung bei der Einrichtung und Benutzung von CPAN verschiedene Fehler auf.

CPAN wird normalerweise bei dem Aufruf `make install` bei der Installation des VNUML Basispaketes (siehe Abschnitt 7.3.5) aufgerufen. Mit dem VNUML Offline Installer wird dieser Aufruf jedoch unterdrückt und die benötigten Perl Module werden manuell installiert.

Für den Fall, dass dennoch eine VNUML Installation mit CPAN durchgeführt werden soll, findet sich im **Anhang C** ein kommentierter CPAN Konfigurationsdialog.

8 Anwendung von VNUML

Beim VNUML-Parser handelt es sich um ein Perl-Skript (vnumlparser.pl), welches von der Konsole aus aufgerufen werden muss. Das Verhalten des Parsers kann durch eine Vielzahl von Parametern gesteuert werden. Alle verfügbaren Parameter könnnen über den Aufruf *vnumlparser.pl -H* in der Konsole ausgegeben werden:

```
 Usage: vnumlparser.pl -t VNUML_file [-o prefix] [-m mount_point] [-c vnuml_dir]
[-T tmp_dir] [-k] [-i] [-S scan_mode ] [-w delay] [-B]
[-e screen_file] [-v] [-g]
vnumlparser.pl -s VNUML_file [-T tmp_dir] [-M vm_list] [-k] [-i] [-S scan_mode ] [-B] [-v] [-g]
vnumlparser.pl -p VNUML_file [-T tmp_dir] [-M vm_list] [-k] [-i] [-S scan_mode ] [-B] [-v] [-g]
vnumlparser.pl -r VNUML_file [-T tmp_dir] [-M vm_list] [-k] [-i] [-S scan_mode ] [-B] [-v] [-g]
vnumlparser.pl -d VNUML_file [-c vnuml_dir] [-F] [-T tmp_dir] [-k] [-i] [-S scan_mode ]
[-B] [-v] [-g]
vnumlparser.pl -P VNUML_file [-T tmp_file] [-k] [-i] [-v] [-g]
vnumlparser.pl -H
vnumlparser.pl -V
Mode:
-t VNUML_file, build topology (using VNUML_file) as source
-s VNUML_file, start simulation (using VNUML_file) as source
-p VNUML_file, stop simulation (using VNUML_file as source)
-r VNUML_file, restart simulation (using VNUML_file as source)
-d VNUML_file, destroy current simulation (using VNUML_file as source)
-P VNUML_file, purge simulation (WARNING: it will remove UML cowed filesystems!)
Pseudomode:
-V, show program version and exits.
-H, show this help message and exits.
Options:
-o prefix, dump UML boot messages output to files (using given prefix in pathname)
-m mount_point, use mount_point as mount point to configure UML filesystem (default is /mnt/
-c vnuml_dir, vnumlparser.pl working directory (default is /var/vnuml)
-F force stopping of UMLs (warning: UML filesystems may be corrupted)
-S scan_mode, the way UML readiness is scanned (default is 0, socket style)
-w delay, waits delay seconds between UMLs booting (defult is wait no time)
-B blocking mode
-e screen_file, make screen configuration file for pts devices
-i interactive execution (in combination with -v mode)
-v verbose mode on
-g debug mode on (overrides verbose)
-T tmp_dir, temporal files directory (default is /tmp)
-M vm_list, start/stop/restart simulation in vm_list UMLs (a list of names separated by ,)
```

Vor dem ersten Start muss ein RSA-Schlüsselpaar zur Authentifizierung in die Datei */root/.ssh/identity.pub* angelegt werden: *ssh-keygen -t rsa1* [enter] Jetzt die nächsten 3 Abfragen mit [enter] bestätigen.

8.1 VNUML starten und beenden

8.1.1 Szenarien starten

Ein in einer VNUML-Datei definiertes Szenario wird durch die Angabe des Parameters *-t* beim Aufruf des Parsers hochgefahren. Wenn mehrere Szenarien gleichzeitig laufen, müssen sich diese im Namen unterscheiden.

Mit dem zusätzlichen Parameter *-w* ist es möglich eine Bootverzögerung der virtuellen Maschinen in Sekunden anzugeben. Auf diese Weise können UML-Stabilitätsprobleme umgangen werden, die bei gleichzeitigen oder dicht aufeinanderfolgenden Boot-Vorgängen auftauchen können.

Das Hochfahren des Szenarios nimmt i.d.R. eine längere Zeit in Anspruch. Es sollte unbedingt gewartet werden, bis der Parser sich von selbst beendet. Ist das Szenario hochgefahren, folgt eine etwas längere Wartezeit, in der ständig angegeben wird, dass der Parser darauf wartet, dass der erste virtueller Rechner per sshd erreichbar wird, z.B.

```
211 seconds eslapsed...
R1 sshd is not ready (socket style)
```

Vorgehen beim Hochfahren eines Szenarios:

1. Schritt: ins Verzeichnis der XML-Szenariodatei wechseln, z.B.: *cd /usr/local/share/vnuml/examples* [enter]
2. Schritt: Szenario starten: *vnumlparser.pl -t tutorial.xml -vB* [enter]
3. Schritt: Simlationen starten oder beliebige Befehle ausführen (nachzulesen in den entsprechenden Manuals) z.B.:
   - einloggen auf der virtuellen Maschine uml1: *ssh -1 uml1* [enter]
   - ping an 10.0.0.2: *ping 10.0.0.2* [enter]
   - ping beenden mit [strg+c]
   - ausloggen aus uml1: *exit* [enter]
   - etc...

Nachdem nun alle Rechner der Topologie (virtuell) zur Verfügung stehen und z.B. per ssh erreichbar sind, können in dieser Umgebung die Simulationen gestartet werden. In der aktuellen VNUML-Version 1.5 startet man diese unter Verwendung des Parameters *-x*, z.B.:*vnumlparser.pl -x start@tutorial_plus.xml -v*. Mit diesem Aufruf werden alle Befehle ausgeführt, die in der XML-Datei innerhalb eines <exec>-Tag mit dem Wert „start“ im seq-Attribut definiert sind, z.B.:

```
<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>
```

Beendet wird die Simulation entsprechend mit dem Aufruf:*vnumlparser.pl -x stop@tutorial_plus.xml -v*, wobei alle Befehle innerhalb des <exec>-Tag mit dem Wert „stop“ im seq-Attribut ausgeführt werden (siehe Kapitel XML Syntax). Mit diesem Verfahren lassen sich in einem laufenden Szenario beliebige Simulationen, wie z.B. das Betreiben von Routern mit den Linux Routing-Deamons, durchführen. Weitere Informationen zum Arbeiten in Simulationen sind in den entsprechenden Ausarbeitungen zu finden.

8.1.2 Szenarien beenden

Ein Szenario wird mit dem Aufruf *vnumlparser.pl -d tutorial.xml -v* beendet. Hierbei werden zunächst alle virtuellen Rechner ordnungsgemäß heruntergefahren. Im Anschluss wird die Netzwerktopologie, also die virtuellen Bridges oder Switches sowie alle virtuellen Schnittstellen, vom Host entfernt. Das Dateisystem der VNUML-Rechner bleibt nach dem Herunterfahren des Szenarios weiterhin bestehen und kann für zukünftige Simulationen erneut verwendet werden.

Nach einem Absturz oder Fehler kann mit dem Parameter *-F* ein Szenario auch gewaltsam beendet werden.

8.2 XML Syntax

Die XML Sprache von VNUML wird durch die DTD Version 1.5 (wird automatisch mitinstalliert) definiert. Wie jede andere wohl-geformte XML-Datei beginnt eine VNUML-Datei mit den folgenden zwei Zeilen:

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

Kommentare innerhalb der Dateien können wie gewohnt benutzt werden:

```
<!-- this is a comment -->
```

Die vier Haupt-Tags 1.Stufe sind

<**global**> beschreibt allen globalen Elemtente der Simulation

<**net**> beschreibt virtuelle Netzwerke

<**vm**> spezifiziert die virtuellen Maschinen

<**host**> Konfiguration des Host (optional)

und werden im Folgenden genauer spezifiziert.

Ein Szenario wird also folgendermaßen umrahmt:

```
<!-- Rahmen einer VNUML-Spezifikation -->
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE vnuml SYSTEM "usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
 <global>
  ...
 </global>
   <vm>
    ...
   </vm>
...
</vnuml>
```

8.2.1 Globale Definitionen <global>

Globale Definitionen betreffen das gesamte Szenario und gelten somit für alle Rechner des virtuellen Netzwerks. Anhand der Einstellungen, die zwischen den beiden Tags <global> und </global> vorgenommen werden können, werden

die Rahmenbedingungen der Simulation abgesteckt. Die globalen Definitionen folgen unmittelbar nach dem <vnuml>- Tag. Die folgende Auflistung erläutert die Bedeutungen der 12 verfügbaren Tags.

<**version**> Dieses Tag muss in der Datei genau ein Mal gesetzt werden und gibt die Version der VNUML-DTD-Datei (also der verwendeten Sprache) an, z.B. <version>1.5</version>

<**simulation_name**> Über dieses Tag wird einem Szenario ein (eindeutiger) Name zugewiesen. Die Angabe eines Namens ist zwingend. Der Dateiname des Szenarios sollte dem Dateinamen entsprechen. Das Szenario mit Namen *tutorial* sollte in der Datei *tutorial.xml* gespeichert werden.

<**ssh_key**> Dieses optionale Tag ermöglicht die Angabe des absoluten Pfades der Datei, die den öffentlichen SSH-Schlüssel des Hosts enthält. Dieser Schlüssel wird dann auf allen virtuellen Rechnern installiert, wodurch die ständige Eingabe des SSH-Passwortes entfällt. Das Attribut „version“ kann angegeben werden, um zwischen SSH Version 1 und 2 zu unterscheiden (version=“1“ für SSHv1 oder version=“2“ für SSHv2). Einen RSA1 public key (SSHv1) erzeugt man vor dem ersten Start mit *ssh-keygen -t rsa1.*

<**automac**> Dieses Tag bewirkt die automatische Erzeugung von MAC- Adressen. Die Verwendung ist optional. Wird <automac> verwendet, müssen die MAC-Adressen der virtuellen Rechner nicht mehr von Hand über das <mac>-Tag gesetzt werden. Die Angabe eines Offsets ist über das Attribut „offset“ möglich. Dies ist sinnvoll, wenn mehrere Simulationen mit automatischen MAC-Adressen gleichzeitig laufen und interagieren sollen. Die Verwendung dieses Tags wird empfohlen um doppelten Mac-Adressen vorzubeugen. Wird innerhalb des <if>-Tags durch <mac> eine spezifische Adresse zugewiesen und global ist trotzdem <automac> aktiv, wird hierbei vom Parser die selbst definierte Adresse gewählt. In diesem Fall sollte besonders auf die Eindeutigkeit geachtet werden, damit durch <automac> nicht eine schon existierende Adresse generiert wird. Adressen werden in folgendem Format generiert: *fe:fd:0:Z:X:Y*, wobei X die Nummer der virtuellen Maschine ist (aufsteigend nach Definition in der XML-Datei, beginnend bei 1) und Y die Nummer der Schnittstelle (definiert über das id-Atrribut im <if>-Tag). Z ist der Wert des offset (0 falls nicht angegeben).

<**ip_offset**> Über dieses optionale Tag kann ein Offset definiert werden, der auf die Management-Interfaces (siehe Tag <mng_if>) der virtuellen Rechner addiert wird. Der Defaultoffset ist 0. Anhand des Attributs prefix kann ein Präfix für IPv4-Adressen angegeben werden, die auf dem Host für die virtuellen Rechner generiert werden müssen. Es ist optional und wird standardmäßig auf 192.168 gesetzt. Das Präfix umfasst stets die ersten zwei Byte der IP-Adresse. Somit wird mit prefix immer ein Klasse B

Netz angegeben. Laufen mehrere Szenarien gleichzeitig muss ein ip_offset angegeben werden, um Überschneidungen zu vermeiden.

<**netconfig**> Dieses optionale Argument ermöglicht das An- und Ausschalten des Spanning Tree Protokolls für die virtuellen Bridges, die von VNUML erzeugt werden. Desweiteren kann der promiscuous Mode (Schnittstelle leitet alle Frames an Schnittstellen-Host weiter, auch wenn sie nicht für ihn bestimmt sind) aktiviert oder deaktiviert werden (promisc-Attribut). Standardmäßig gilt: stp="off" und promisc="on".

<**host_mapping**> Dieses Tag kann optional gesetzt werden und ermöglicht das Ansprechen der virtuellen Rechner mit ihrem Namen, den sie über das Attribut „name" des <vm>-Tags bekommen haben (sh. unten), anstatt über die IP-Adresse. Das Einloggen per SSH z.B. ist so einfacher möglich durch: *ssh -1 uml1* anstatt *ssh -1 192.168.0.2*.

<**shell**> Dieses Tag ist optional und ermöglicht die Angabe einer Shell, die zum Ausführen der Kommandos des VNUML-Parsers verwendet werden soll. Standardmäßig wird */bin/bash* verwendet. Während der Installation kann auch schon eine andere Standard-Shell angegeben werden.

<**tun_device**> Über dieses optionale Tag kann ein alternatives Device zum Erzeugen von virtuellen Netzwerk- Schnittstellen angegeben werden. Das default-Device ist */dev/net/tun*.

<**default_filesystem**> Mit diesem optionalen Tag kann ein neues Standard-Dateisystem angegeben werden, welches verwendet werden soll, wenn bei der Definition eines virtuellen Rechners das Tag <filesystem> ausgelassen wird. Standardmäßig wird das Dateisystem verwendet, welches sich in der Datei *filesystems/root_fs_tutorial* des VNUML-Installationsverzeichnisses befindet. Mit dem Attribut „type" kann man zwischen den Werten "cow" und "copy" wählen. Empfohlen wird hier aus Performanzgründen der Wert "cow", da so nur Änderungen am Dateisystem gespeichert werden.

<**default_kernel**> Dieses Tag ermöglicht die Angabe eines neuen default-Linux-Kernels, der verwendet werden soll, wenn bei der Definition eines virtuellen Rechners kein expliziter Kernel durch das <kernel>-Tag angegeben wird. Wird nichts angegeben, wird der Defaultkernel in */usr/local/share/vnuml/kernels/linux* verwendet.

<**basedir**> Zum Kopieren von Verzeichnissen (samt Inhalt) des Hosts auf einen virtuellen Rechner kann während der Definition eines virtuellen Rechners mit dem Tag <filetree> ein Host-Verzeichnis angegeben werden. Die Pfadangabe, die im <filetree>-Tag gemacht wird, ist relativ zum Pfad, welcher mit dem <basedir>-Tag angegeben wird. Wenn also z.B. Config-Dateien von Routern in verschiedenen Unterverzeichnissen eines einzigen Verzeichnisses liegen, und sich dieses ändert, so reicht eine Änderung des <basedir>-Tag um die komplette Pfadangabe aller Dateien zu

berichtigen. Dieses Tag ist optional und standardmäßig der Pfad /.

8.2.2 Virtuelle Netzwerke <net>

Dieses Tag bietet keine weiteren Tags zum Einstellen weiterer Optionen (ausser <bw> falls "ppp" genutzt wird, siehe Online-Referenz) und dient lediglich dem Einführen eines virtuellen Netzes mit einem Namen. Die IP-Adressen, die in einem Netz vorhanden sind, ergeben sich aus den Schnittstellen der virtuellen Rechner, die mit einem Netz verbunden sind. Um ein Netzwerk identifizieren zu können, muss ein eindeutiger Name über das Attribut „name" vergeben werden. Version 1.5 bietet zwei mögliche Netzwerke an, die über das Attribut „type" gesetzt werden können.

virtual_bridge Dieser Wert bewirkt den Einsatz einer virtuellen Bridge zum Verbinden der virtuellen Rechner, die mit dem Netz verbunden sind. Zum Erzeugen eines solchen Netzwerks werden root-Rechte benötigt. Nur für diese Art Netz kann das Attribut „external" angegeben werden, welches das Verbinden der virtuellen Bridge mit einer realen Schnittstelle des Hosts ermöglicht. Auf diese Weise kann ein virtuelles Netzwerk mit dem Internet verbunden werden. Falls die angegebene Host-Schnittstelle einem VLAN angehört, kann über das Attribut „vlan" die Nummer des VLANS angegeben werden, in dem sich die Schnittstelle befindet.

uml_switch Durch die Angabe dieses Werts wird ein UML-Switch zum Verbinden der virtuellen Rechner des Netzes erzeugt. Für solche Netzwerke sind keine root-Rechte notwendig. Möchte man ein Hub anstatt einem Switch verwenden, kann über das Attribut „hub" der Switch in ein Hub umgewandelt werden (hub = "yes"). Der Defaultwert für dieses Attribut ist "no".

Der Standardwert von „type" ist "virtual_bridge". Ein Netz kann in mehreren Simulationen gleichzeitig verwendet werden, wenn in allen VNUML-Dateien das betreffende Netz mit exakt den gleichen Attributen definiert ist. Die Namen der Netzwerke müssen über alle gleichzeitig laufenden Simulationen einzigartig sein. Es können nahezu beliebig viele Netze erzeugt werden. Die einzige Begrenzung besteht in der Tatsache, dass die Länge der Netznamen auf sieben Buchstaben begrenzt ist.

8.2.3 Virtuelle UML-Rechner <vm>

Virtuelle Rechner werden mit <vm>-Tags definiert, die wiederrum aus einer Reihe von weiteren Tags bestehen, die u.A. die Vernetzung des virtuellen Rechners mit virtuellen Netzen beschreiben. Jeder virtuelle Rechner bekommt über

das Attribut „name“ einen eindeutigen Namen zugewiesen. Die Länge dieses Namens ist auf sieben Zeichen beschränkt. In einer Simulation (einem Szenario) können maximal 214 = 16384 virtuelle Rechner definiert werden. In der Praxis ist diese hohe Anzahl an virtuellen Rechnern jedoch nicht zu erreichen, da hierfür i.d.R. nicht genügend Ressourcen auf dem Host vorhanden sind. Wird das globale Tag <automac> gesetzt, ist die Maximalanzahl an virtuellen Rechnern auf 255 pro Simulation beschränkt. Mit dem optionalen Attribut „order“ kann durch eine positive ganze Zahl angegeben werden, welcher virtuelle Rechner auf dem Host bevorzugt simuliert werden sollen. Standardmäßig bekommen alle virtuellen Rechner die gleiche Priorität zugewiesen und werden nach der Reihe ihrer Definition gebootet bzw. heruntergefahren. Im Folgenden werden die Tags beschrieben, die innerhalb des <vm>-Tags gesetzt werden können und jede VM genauer spezifizieren.

<**filesystem**> Dieses Tag ist optional und wird standardmäßig auf den Wert des globalen Tags <default_filesystem> gesetzt. Es gibt den Ort der Datei an, in der das root-Dateisystem für den virtuellen Rechner angelegt werden soll. Über das Attribut „type“ kann angegeben werden, wie die Datei mit dem Dateisystem des virtuellen Rechners benutzt werden soll. Wegen Performanzgründen wird auch hier nur der Wert “cow“ (copy-on-write) empfohlen (für die speziellen Werte “copy“, “direct“, “hostfs“ siehe Online-Referenz).

<**mem**> Über dieses optionale Tag kann die Größe des Arbeitsspeichers des virtuellen Rechners angegeben werden. Die Suffixe K und M können für Kilobyte bzw. Megabyte verwendet werden. Der Standardwert ist 32M, was 32 MByte bedeutet.

<**kernel**> Mit diesem Tag kann der Ort eines speziellen Linux-Kernels angegeben werden, den der virtuelle Rechner booten soll. Der Defaultkernel wird über das globale Tag <default_kernel> gesetzt und automatisch gebootet, wenn nichts anderes angegeben wird. Der Dateiname eines alternativen Kernels muss am Anfang den String „kernel“ enthalten.

<**boot**> Dieses optionale Tag erlaubt die Angabe von Boot-Parametern für den Kernel. Die Parameter müssen sinnvoll angegeben werden, da sie vom VNUML-Parser nur an den Kernel weitergegeben, jedoch nicht kontrolliert werden. Innerhalb dieses Tags kann mit dem Tag <con0> die Konsole auf dem Host angegeben werden, auf die der virtuelle Rechner seine Ausgaben schreiben soll. Die Ausgabe auf eine X-Konsole ist auch möglich (<con0>xterm</con0>). Nur in diesem Fall dürfen durch ein <xterm>-Tag dem Aufruf dieser X-Konsole weitere Parameter mitgegeben werden.

<**mng_if**> Dieses optionale Tag ermöglicht das Erzeugen von Management-Interfaces. Management-Interfaces sind Punkt-zu-Punkt-Verbindungen von einem virtuellen Rechner zum Host. Sie existieren im virtuellen Rechner immer als eth0. Im Host wird für jeden virtuellen Rechner, der ein

Management-Interface zur Verfügung stellt eine Schnittstelle mit dem Namen name-ethX erzeugt, wobei name den Namen des virtuellen Rechners meint. Der Wert “no“ bewirkt, dass der virtuelle Rechner kein Management-Interface zur Verfügung stellt und somit allein durch die Simulation selbst erreichbar und konfiguriert sein muss. Standardmäßig wird jedoch ein Management-Interface zur Verfügung gestellt. Über Dieses Interface ist das Einloggen vom Host aus mit SSH auf den virtuellen Rechner möglich.

<**if**> Mit diesem optionalen Tag werden Netzwerk-Schnittstellen auf dem virtuellen Rechner angelegt. Das Attribut „id“ gibt das Interface an, welches verbunden werden soll. Die Nummer des Interfaces muss größer als 0 sein (0 ist für das Management-Interface reserviert). Im virtuellen Rechner wird dann das Interface ethn angelegt (wenn n die Nummer des Interfaces ist). Im Host wird ein Interface mit dem Namen name-ethn angelegt (sog. UML-Interfaces). Mit dem Attribut „net“ wird das Netzwerk angegeben, mit dem die neue Schnittstelle verbunden werden soll. Der Name muss einem zuvor angelegten Netz (im Attribut „name“) entsprechen. Innerhalb von <if> können drei weitere Tags benutzt werden:

<**mac**> Hier kann eine MAC-Adresse angegeben werden. Falls dieses Tag nicht benutzt wird, erfolgt eine automatische Zuweisung einer Adresse, wenn das globale Tag <automac> gesetzt ist. Falls nicht, kann eine MAC-Adresse nur dann automatisch vergeben werden, wenn IPv4 verwendet wird. Die ersten beiden Bytes einer hier angegebenen MAC-Adresse sollten fe:fd: sein. MAC-Adressen müssen über alle gleichzeitig laufenden Simulationen einzigartig sein.

<**ipv4**> Mit diesem Tag wird eine IPv4-Adresse an die Schnittstelle gebunden. Mit dem Attribut „mask“ kann eine Subnetzmaske angegeben werden. Der Standardwert ist die Klasse C Subnetzmaske. Eine Schnittstelle kann mehrere IPv4-Adressen haben.

<**ipv6**> Mit diesem Tag wird eine IPv6-Adresse an die Schnittstelle gebunden. Die Maske wird nicht mit einem Attribut angegeben, sondern durch das Anhängen der Anzahl der Maskenbits mit /. Ein Beispiel hierfür ist die Adresse 3ffe:ffff::3/64 mit einer 64-Bit-Maske. Eine Schnittstelle kann mehrere IPv6-Adressen haben.

<**route**> Dieses optionale Tag ermöglicht das Konfigurieren von statischen Routen, die dann der Routing-Tabelle des virtuellen Rechners hinzugefügt werden. Dieses Tag ist nicht die einzige Möglichkeit statische Routen in die Tabelle einzutragen. Optional kann die Route auch bei laufender Simulation in den virtuellen Rechner per SSH eingetragen werden (mit dem route-Befehl). Die Gateway-Adresse für einen Routing-Eintrag wird über das Attribut „gw“ angegeben. Der Routertyp kann über das Attribut „type“ bestimmt werden. Mit type = “inet“ wird ein IPv4- und mit type = “inet6“ ein IPv6-Routing-Eintrag erzeugt. Der Wert dieses Tags ist das Ziel des Routing-Eintrags. Es können mehrere statische Routen erzeugt werden.

<**forwarding**> Mit diesem Flag kann das Weiterreichen von IP-Paketen anhand der Routing-Tabelle des virtuellen Rechners aktiviert werden. Das Attribut „type“ gibt dabei an, welche Art von IP-Paketen weitergeleitet werden soll. Durch die beiden zulässigen Werte “ipv4“ und “ipv6“ lassen sich hier v4 und v6 Pakete unterscheiden. Standardmäßig werden IPv4- und IPv6-Pakete weitergeleitet.

<**filetree**> Dieses Tag ermöglicht die Angabe eines Host-Verzeichnisses (relativ zum global definierten Tag <basedir>), welches vom Host komplett in das Dateisystem des virtuellen Rechners kopiert wird. Das angegebene Verzeichnis überschreibt bereits vorhandene Verzeichnisse und Dateien auf dem virtuellen Rechner. Auf diese Weise kann ein Rechner einfach durch das Kopieren eines vorbereiteten /etc-Verzeichnisses konfiguriert werden. Dem Tag wird über das Attribut „root“ das Verzeichnis auf dem virtuellen Rechner angegeben, in welches das Host-Verzeichnis kopiert werden soll. Das Attribut „when“ gibt an, ob das Verzeichnis bei jedem Start (“start“) oder am Ende (“stop“) einer Simulation kopiert werden soll. Durch Angabe von “always“ wird das Verzeichnis bei beiden Ereignissen kopiert. Dieses Tag ist optional und kann mehrfach innerhalb einer Definition eines Rechners verwendet werden.

<**exec**> Dieses optionale Tag ersetzt die <start>- und <stop>-Tags der Versionen bis 1.4. Es spezifiziert ein Kommando, welches beim Start der Simulation auf dem Rechner ausgeführt werden soll. Durch die mehrfache Verwendung von <exec>-Tags können mehrere Kommandos in der angegebenen und somit vorbestimmten Reihenfolge ausgeführt werden. Das Attribut „seq“ definiert einen String (z.B. seq=“test“), der die Kommandosequenz bezeichnet, die bei Aufruf mit dem -x Parameter ausgeführt wird (z.B. Sequenz test in file.xml wird ausgeführt bei *-x test@file.xml*).

Beispiel: Ausschnitt aus einer Datei file.xml

```
...
<exec seq="sample" type="verbatim">/usr/local/bin/sample1.sh</exec>
<exec seq="test" type="verbatim">/usr/local/bin/test1.sh</exec>
<exec seq="sample" type="verbatim">/usr/local/bin/sample2.sh</exec>
<exec seq="test" type="verbatim">/usr/local/bin/test2.sh</exec>
...
```

Wird nun auf der Kommandozeile folgender Befehl ausgeführt:

```
vnumlparser.pl -x sample@file.xml
```

so wird in der entsprechenden virtuellen Maschine oder dem Host folgendes ausgeführt:

```
/usr/local/bin/sample1.sh
/usr/local/bin/sample2.sh
```

Es werden also die Shell-Skripte sample1.sh und sample2.sh in dieser Reihenfolge auf der virtuellen Maschine gestartet.
Das Attribut „type“ ermöglicht die Interpretaion des Tag-Wertes als einen (auf dem Host absoluten) Pfad, der auf eine Datei zeigt, die alle auf dem virtuellen Rechner auszuführenden Kommandos enthält (type = “file“). Soll der angegebene Tag-Wert als einzelnes shell-Kommando interpretiert werden, muss type auf “verbatim“ gesetzt werden. Die Angabe einer Datei ist insbesondere dann empfehlenswert, wenn das Kommando Zeichenketten enthält, die als XML-Sonderzeichen missverstanden werden können.

8.2.4 Host-Konfiguration <host>

Die Host-Konfiguration sollte nur durchgeführt werden, wenn der Host auch Teil der Simulation ist. Das <host>-Tag funktioniert analog zum <vm>-Tag. Innerhalb des Tags können die Tags <route>, <forwarding> und <exec> mit der oben beschriebenen Semantik verwendet werden. Die beiden folgenden Tags sind zusätzlich im <host>-Tag vorhanden:

<**hostif**> Dieses Tag ist das Pendant zum <if>-Tag. Es dient zum Definieren einer Verbindung zwischen einer Schnittstelle und einem Netz. Die Schnittstelle für die neue virtuelle Verbindung wird auf dem Host automatisch erzeugt und hat stets den Namen des virtuellen Netzwerks, mit dem sie verbunden ist. Dieses Tag hat kein id-Attribut. Das Tag <mac> ist ebenfalls nicht verfügbar. Die beiden Tags <ipv4> und <ipv6> haben die gleiche Bedeutung. Virtuelle Netze, deren external-Attribut auf eine Schnittstelle des Hosts gesetzt wurden, müssen mit diesem Tag konfiguriert werden, da jegliche Konfiguration der physikalischen Schnittstelle durch die Verbindung mit der virtuellen Bridge verloren geht. Um die Schnittstelle wie bisher (ohne laufende Simulation) weiternutzen zu können, müssen die vorherigen Einstellungen an dieser Stelle mit den <ipv4>- bzw. <ipv6>-Tags wieder eingetragen werden. Eventuelle Routingeinträge und Standardgateways müssen innerhalb des <host>-Tags mit <route> wiederhergestellt werden.

<**physicalif**> In diesem Tag kann die ursprüngliche Konfiguration einer physikalischen Schnittstelle abgespeichert werden, damit sie nach der Simulation vom VNUML-Parser wiederhergestellt werden kann. Die Konfiguration wird anhand der Attribute „type“ (ipv4 oder ipv6), „name“ (z.B. eth0), „ip“ (IPv6-Adressen mit Angabe der Maske), „mask“ (nur für IPv4-Adressen) und „gw“ (Standardgateway) gesetzt. Das Tag ist optional und standardmäßig leer. Für jede Schnittstelle können maximal zwei dieser Tags definiert werden (jeweils eins für IPv4 und IPv6).

Diese Auflistung ist aus der Seminararbeit *www.uni-koblenz.de/˜steigner/seminar-routingsim/arndt.pdf* entnommen. Die Beschreibungen der Tags wurden erweitert und auf Version 1.5 der VNUML-Syntax angepasst.

Die vollständige Sprach-Definition findet sich unter
http://jungla.dit.upm.es/∼vnuml/doc/1.5/reference/index.html.

Eine XML-Beispieldatei die alle XML Tags und Attribute der VNUML-Sprache enthält ist in **Anhang A** aufgeführt.

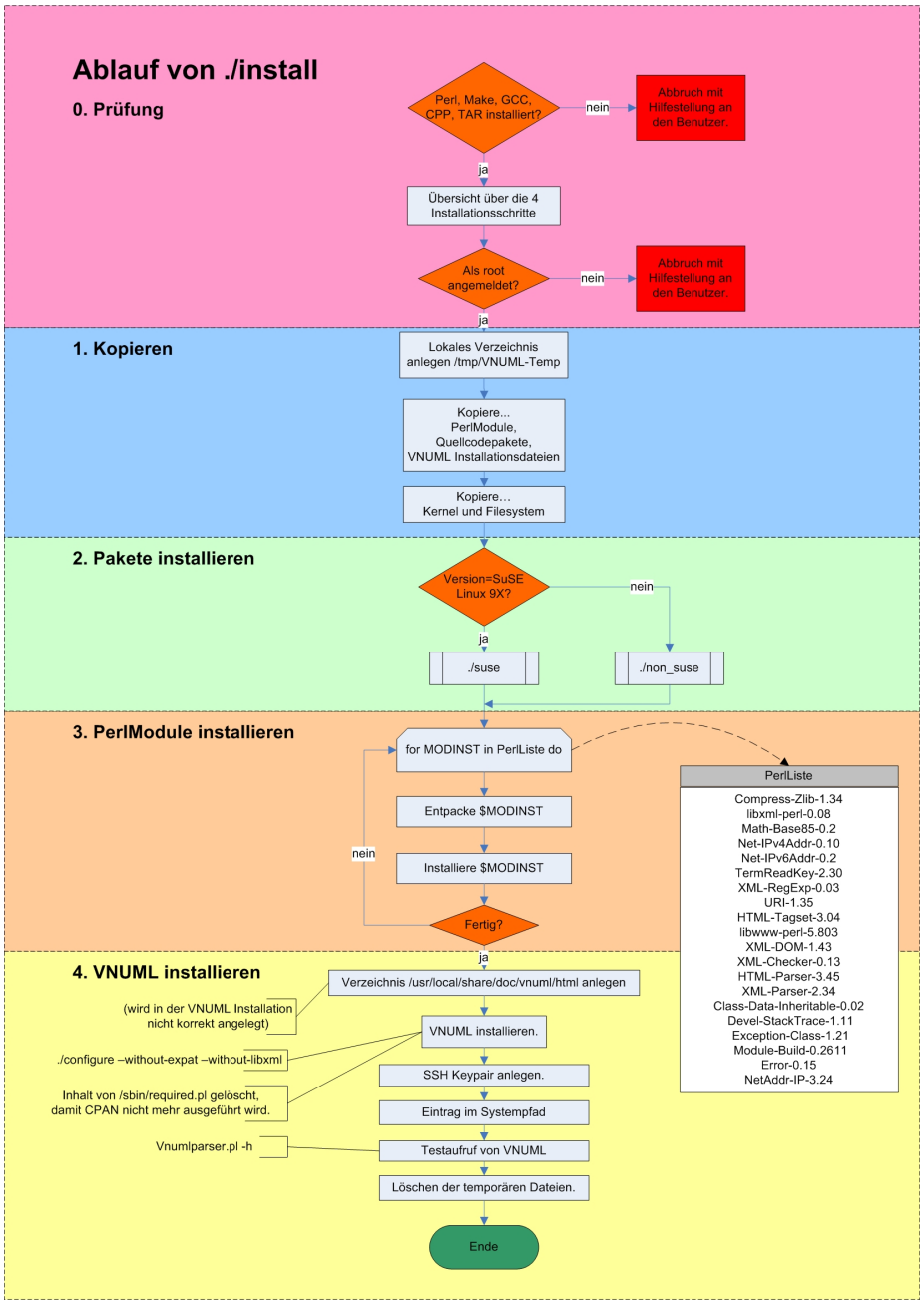
9 ANHANG A - XML-Beispieldatei

XML-Beispieldatei die alle VNUML 1.5 Tags und Attibute aufspannt:

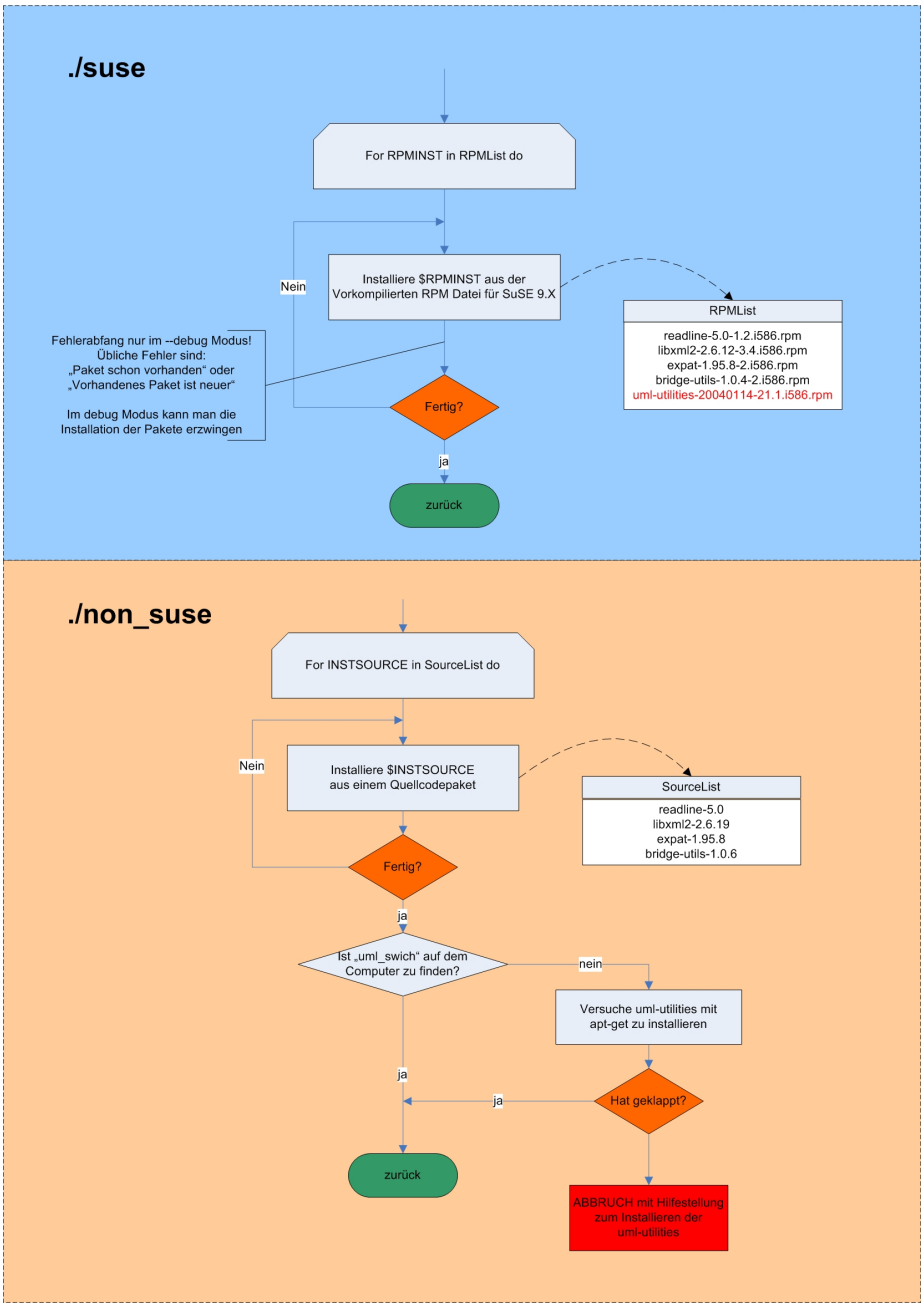
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.5</version>
    <simulation_name>parser_test</simulation_name>
    <ssh_key version="2">/tmp/SomeSshKey</ssh_key>
    <automac offset="10"/>
    <ip_offset prefix="10.60">100</ip_offset>
    <netconfig stp="on" promisc="off"/>
    <host_mapping/>
    <shell>/bin/otherbash</shell>
    <tun_device>/dev/net/othertun</tun_device>
    <default_filesystem type="copy">/tmp/default_filesystem</default_filesystem>
    <default_kernel>/tmp/default_kernel</default_kernel>
    <basedir>/tmp/basedir</basedir>
  </global>
  <net name="Net0" mode="virtual_bridge" type="lan" external="eth11" vlan="11"/>
  <net name="Net1" mode="uml_switch" type="lan" external="eth12" vlan="12" hub="yes"/>
  <net name="Net2" type="ppp">
    <bw>10M</bw>
  </net>
  <vm name="uml1" order="5">
    <filesystem type="hostfs">/tmp/uml1-hostfs</filesystem>
    <mem>50M</mem>
    <kernel>/tmp/uml1-kernel</kernel>
    <boot>
      <con0>xterm</con0>
      <xterm>xterm,and,it's,arguments</xterm>
    </boot>
    <mng_if>no</mng_if>
    <if id="1" net="Net0">
      <mac>ff:ff:ff:ff:ff:ff</mac>
      <ipv4 mask="255.255.255.128">1.2.3.4</ipv4>
      <ipv6>fe80::2c0:cccc:cccc:cccc/64</ipv6>
    </if>
    <route type="inet" gw="1.2.3.254">default</route>
    <route type="inet" gw="1.2.3.253">0.0.0.0/0</route>
    <route type="inet" gw="1.2.3.252">192.168.0.0/24</route>
    <route type="inet6" gw="fe80::2c0:cccc:cccc:ffff">default</route>
    <route type="inet6" gw="fe80::2c0:cccc:cccc:fffe">2000::/3</route>
    <route type="inet6" gw="fe80::2c0:cccc:cccc:fffd">3ffe:80ee:2fe3:1:/64</route>
    <forwarding type="ipv6"/>
    <filetree root="/usr/local/root_always" when="always">/tmp/root_always</filetree>
    <filetree root="/usr/local/root_start" when="start">/tmp/root_start</filetree>
```

```
    <filetree root="/usr/local/root_stop" when="stop">/tmp/root_stop</filetree>
    <exec seq="sample" type="verbatim">/usr/local/bin/sample_verbatim.sh</exec>
    <exec seq="test" type="file">/usr/local/bin/test_file.sh</exec>
    <exec seq="sample" type="file">/usr/local/bin/sample_file.sh</exec>
    <exec seq="test" type="verbatim">/usr/local/bin/test_verbatim.sh</exec>
  </vm>
  <host>
    <hostif net="Net2">
      <ipv4 mask="255.255.255.128">4.3.2.1</ipv4>
      <ipv6>fe80::2c0:aaaa:aaaa:aaaa/64</ipv6>
    </hostif>
    <physicalif type="ipv4" name="eth8" ip="9.8.7.6" mask="255.255.0.0" gw="9.8.9.9"/>
    <physicalif type="ipv6" name="eth8" ip="fe80::2c0:aaaa:aaaa:bbbb"
     gw="fe80::2c0:cccc:cccc:fffa"/>
    <route type="inet" gw="10.0.3.1">10.0.0.0/16</route>
    <forwarding type="ip"/>
    <exec seq="sample" type="verbatim">/usr/local/bin/sample_verbatim.sh</exec>
    <exec seq="test" type="file">/usr/local/bin/test_file.sh</exec>
    <exec seq="sample" type="file">/usr/local/bin/sample_file.sh</exec>
    <exec seq="test" type="verbatim">/usr/local/bin/test_verbatim.sh</exec>
  </host>
</vnuml>
```

10 ANHANG B - Flussdiagramme



Verlauf der VNUML Offline Installation



Installation der Linux Pakete

11 ANHANG C - CPAN Dialog

Nachfolgend sind die Bildschirmausgaben des CPAN Konfigurationsdialogs incl. beispielhaften Benutzereingaben gelistet. Die Eingaben beziehen sich auf einen Rechner der über das Netzwerk der Universität Koblenz mit dem Internet verbunden ist. Jedoch sind die meisten Eingaben sicherlich auch für viele Rechner außerhalb der Universität passend.

Der CPAN Konfigurationsdialog lässt sich manuell durch Eingabe folgender Befehle aufrufen:

- `cpan [enter]`
- `o conf init [enter]`

Der CPAN Konfigurations-Dialog:

```
CPAN is the world-wide archive of perl resources. It consists of about
100 sites that all replicate the same contents all around the globe.
Many countries have at least one CPAN site already. The resources
found on CPAN are easily accessible with the CPAN.pm module. If you
want to use CPAN.pm, you have to configure it properly.

If you do not want to enter a dialog now, you can answer 'no' to this
question and I'll try to autoconfigure. (Note: you can revisit this
dialog anytime later by typing 'o conf init' at the cpan prompt.)

Are you ready for manual configuration? [yes] yes


The following questions are intended to help you with the
configuration. The CPAN module needs a directory of its own to cache
important index files and maybe keep a temporary mirror of CPAN files.
This may be a site-wide directory or a personal directory.



First of all, I'd like to create this directory. Where?

CPAN build and cache directory? [/root/.cpan]


If you want, I can keep the source files after a build in the cpan
home directory. If you choose so then future builds will take the
files from there. If you don't want to keep them, answer 0 to the
next question.
```

```
How big should the disk cache be for keeping the build directories
with all the intermediate files?

Cache size for build directory (in MB)? [10]


By default, each time the CPAN module is started, cache scanning
is performed to keep the cache size in sync. To prevent from this,
disable the cache scanning with 'never'.

Perform cache scanning (atstart or never)? [atstart]


To considerably speed up the initial CPAN shell startup, it is
possible to use Storable to create a cache of metadata. If Storable
is not available, the normal index mechanism will be used.

Cache metadata (yes/no)? [yes]


The next option deals with the charset your terminal supports. In
general CPAN is English speaking territory, thus the charset does not
matter much, but some of the aliens out there who upload their
software to CPAN bear names that are outside the ASCII range. If your
terminal supports UTF-8, you say no to the next question, if it
supports ISO-8859-1 (also known as LATIN1) then you say yes, and if it
supports neither nor, your answer does not matter, you will not be
able to read the names of some authors anyway. If you answer no, names
will be output in UTF-8.

Your terminal expects ISO-8859-1 (yes/no)? [yes]


If you have one of the readline packages (Term::ReadLine::Perl,
Term::ReadLine::Gnu, possibly others) installed, the interactive CPAN
shell will have history support. The next two questions deal with the
filename of the history file and with its size. If you do not want to
set this variable, please hit SPACE RETURN to the following question.

File to save your history? [/root/.cpan/histfile]
Number of lines to save? [100]


The CPAN module can detect when a module that which you are trying to
build depends on prerequisites. If this happens, it can build the
prerequisites for you automatically ('follow'), ask you for
confirmation ('ask'), or just ignore them ('ignore'). Please set your
policy to one of the three values.
```

```
Policy on building prerequisites (follow, ask or ignore)? [ask] follow


The CPAN module will need a few external programs to work properly.
Please correct me, if I guess the wrong path for a program. Don't
panic if you do not have some of them, just press ENTER for those. To
disable the use of a download program, you can type a space followed
by ENTER.

Where is your gzip program? [/bin/gzip]
Where is your tar program? [/bin/tar]
Where is your unzip program? [/usr/bin/unzip]
Where is your make program? [/usr/bin/make]
Warning: lynx not found in PATH
Where is your lynx program? [] /usr/bin/lynx
Where is your wget program? [/usr/bin/wget]
Warning: ncftpget not found in PATH
Where is your ncftpget program? [] /usr/local/bin/ncftpget
Where is your ftp program? [/usr/bin/ftp]
Where is your gpg program? [/usr/bin/gpg]
What is your favorite pager program? [less]
What is your favorite shell? [/bin/sh]


Every Makefile.PL is run by perl in a separate process. Likewise we
run 'make' and 'make install' in processes. If you have any
parameters (e.g. PREFIX, LIB, UNINST or the like) you want to pass
to the calls, please specify them here.

If you don't understand this question, just press ENTER.

Parameters for the 'perl Makefile.PL' command?
Typical frequently used settings:

    PREFIX=~/perl       non-root users (please see manual for more hints)

Your choice:  []
Parameters for the 'make' command?
Typical frequently used setting:

    -j3             dual processor system

Your choice:  []
Parameters for the 'make install' command?
Typical frequently used setting:

    UNINST=1        to always uninstall potentially conflicting files

Your choice:  []
```

```
Sometimes you may wish to leave the processes run by CPAN alone
without caring about them. As sometimes the Makefile.PL contains
question you're expected to answer, you can set a timer that will
kill a 'perl Makefile.PL' process after the specified time in seconds.

If you set this value to 0, these processes will wait forever. This is
the default and recommended setting.

Timeout for inactivity during Makefile.PL? [0]


If you're accessing the net via proxies, you can specify them in the
CPAN configuration or via environment variables. The variable in
the $CPAN::Config takes precedence.

Your ftp_proxy?  http://cache64.uni-koblenz.de:3128
Your http_proxy?  http://cache64.uni-koblenz.de:3128
Your no_proxy?  localhost


If your proxy is an authenticating proxy, you can store your username
permanently. If you do not want that, just press RETURN. You will then
be asked for your username in every future session.

Your proxy user id?
You have no /root/.cpan/sources/MIRRORED.BY
  I'm trying to fetch one
LWP not available
CPAN: Net::FTP loaded ok
Fetching with Net::FTP:
  ftp://ftp.perl.org/pub/CPAN/MIRRORED.BY
Fetching with Net::FTP
  ftp://ftp.perl.org/pub/CPAN/MIRRORED.BY.gz

Trying with "/usr/bin/lynx -source" to get
    ftp://ftp.perl.org/pub/CPAN/MIRRORED.BY


Now we need to know where your favorite CPAN sites are located. Push
a few sites onto the array (just in case the first on the array won't
work). If you are mirroring CPAN to your local workstation, specify a
file: URL.

First, pick a nearby continent and country (you can pick several of
each, separated by spaces, or none if you just want to keep your
existing selections). Then, you will be presented with a list of URLs
of CPAN mirrors in the countries you selected, along with previously
selected URLs. Select some of those URLs, or just keep the old list.
Finally, you will be prompted for any extra URLs -- file:, ftp:, or
```

```
http: -- that host a CPAN mirror.

(1) Africa
(2) Asia
(3) Central America
(4) Europe
(5) North America
(6) Oceania
(7) South America
Select your continent (or several nearby continents) [] 4
Sorry! since you don't have any existing picks, you must make a
geographic selection.

(1) Austria
(2) Belgium
(3) Bosnia and Herzegovina
(4) Bulgaria
(5) Croatia
(6) Czech Republic
(7) Denmark
(8) Estonia
(9) Finland
(10) France
(11) Germany
(12) Greece
(13) Hungary
(14) Iceland
(15) Ireland
(16) Italy
16 more items, hit SPACE RETURN to show them
Select your country (or several nearby countries) [] 11
Sorry! since you don't have any existing picks, you must make a
geographic selection.

(1) ftp://cpan.noris.de/pub/CPAN/
(2) ftp://cpan.provocation.net/
(3) ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/CPAN
(4) ftp://ftp.cs.tu-berlin.de/pub/lang/perl/CPAN/
(5) ftp://ftp.freenet.de/pub/ftp.cpan.org/pub/CPAN/
(6) ftp://ftp.gmd.de/mirrors/CPAN/
(7) ftp://ftp.gwdg.de/pub/languages/perl/CPAN/
(8) ftp://ftp.kgt.org/pub/mirrors/CPAN/
(9) ftp://ftp.leo.org/pub/CPAN/
(10) ftp://ftp.mpi-sb.mpg.de/pub/perl/CPAN/
(11) ftp://ftp.rub.de/pub/CPAN/
(12) ftp://ftp.uni-erlangen.de/pub/source/CPAN/
(13) ftp://ftp.uni-hamburg.de/pub/soft/lang/perl/CPAN/
(14) ftp://netmirror.org/CPAN/
(15) ftp://pandemonium.tiscali.de/pub/CPAN/
(16) http://www.mirrorspace.org/cpan/
```

```
1 more items, hit SPACE RETURN to show them
Select as many URLs as you like (by number),
put them on one line, separated by blanks, e.g. '1 4 5' [] 12 13 6 4

Enter another URL or RETURN to quit: []
New set of picks:
  ftp://ftp.uni-erlangen.de/pub/source/CPAN/
  ftp://ftp.uni-hamburg.de/pub/soft/lang/perl/CPAN/
  ftp://ftp.gmd.de/mirrors/CPAN/
  ftp://ftp.cs.tu-berlin.de/pub/lang/perl/CPAN/
```

12 ANHANG D - Fehlermeldungen

Es folgt eine Sammlung von Fehlermeldungen, die uns während der Installation und der Benutzung von VNUML begegnet sind mit Lösungsvorschlägen.

12.1 Während der Installation

Meldung: *no acceptable C compiler found in PATH*
Auftreten: VNUML Installationspaket: Ausführung von `./configure`
Lösung: *gcc* unter Yast installieren
(*glibc-devel* wird automatisch mitinstalliert)

Meldung: *[make1]: *** [uml_console.o] Fehler 1*
Auftreten: Installation von uml_utilities aus einem Quellcode-Paket
Lösung: uml-utilities nicht aus dem Quellcode-Paket installieren, sondern eine vorkompilierte Installationsdatei benutzen. (siehe Abschnitt 6.3)

Meldung: *Conflict...*
Auftreten: Installation von RPM Paketen
Lösung: `rpm -i --force name-des-pakets.rpm`

Meldung: *Verzeichnis /usr/local/share/doc/vnuml/html nicht gefunden*
Auftreten: *make install* bei Installation von VNUML
Lösung: Verzeichnis manuell anlegen mit
`mkdir -p /usr/local/share/doc/vnuml/html`
Anschließend erneut `make install`

Meldung: *Cant't locate XML/DOM.pm in @INC...*
Auftreten: Start von *vnumlparser.pl*
Lösung: Perl-Modul *XML/DOM* manuell installieren.

Meldung: *Can't locate XML/DOM/ValParser.pm in @INC...*
Auftreten: Start von *vnumlparser.pl*
Lösung: Perl-Modul *XML-Checker-0.13* manuell installieren.

Meldung:	*Some required binary files are missing.*
Auftreten:	Start von *vnumlparser.pl* mit einer XML Datei
Lösung:	Erneute Installation der Linux Programm-Pakete. (siehe Abschnitt 7.3.3)

Meldung (ähnlich):	*Cannot parse DTD file://...* oder *Cannot locate file file://...*
Auftreten:	Start von *vnumlparser.pl* mit einer XML Datei
Lösung:	Installieren des Perl-Moduls *URI*.

Meldung:	*524 (/root/.ssh/identity.pub is not a valid file (perhaps does not exist)*
Auftreten:	Start von vnumlparser.pl
Lösung:	`ssh-keygen -t rsa1` *Enter file in which to save the key (/root/.ssh/identity):* `[Enter]` *Enter passphrase...* `[Enter]`

12.2 Bei der Anwendung von VNUML

Problem:	Passwortabfrage bei Einwahl per SSH in eine virtuelle Maschine
Auftreten:	Das VNUML Szenario ist gestartet. Beim Einwählen per SSH in eine Virtual Machine wird ein Passwort abgefragt.
Ursache:	SSH benutzt verschiedene Authentifikations-Modi.
Lösung1:	Passwort für das Filesystem eingeben: `xxxx`
Lösung2:	SSH explizit im Modus 1 starten: `ssh -1 name-der-virtuellen-maschine`

Problem:	Das VNUML Szenario ist ordnungsgemäß gestartet, man kann sich auch mit SSH in die einzelnen virtuellen Maschinen einwählen. Jedoch lässt sich keine andere virtuelle Maschine anpingen.
Ursache:	Die Linux Firewall (besonders bei SuSE) blockiert die Verbindungen zwischen den virtuellen Maschinen.
Lösung1:	Deaktivieren der Linux Firewall (bei SuSE unter YAST!)
Lösung2:	In der XML Datei des Szenarios muss in den <net> -Tags der Parameter *mode='uml_switch'* eingefügt werden. (bzw. ein evtl. vorhandener Parameter *mode='virtual_bridge'* muss in *mode='uml_switch'*) geändert werden.