VNUML-Installation im Windows-Pool an der Universität Koblenz-Landau

Studienarbeit
im Studiengang

Informatik

an der
Universität Koblenz-Landau

vorgelegt von

Mario Wendling
- 200210116-

Arbeitsgruppe Prof. Dr. Steigner

Betreuer: Prof. Dr. Christoph Steigner, Harald Dickel, Frank Bohdanowicz

Koblenz, im August 2009

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

……...
(Ort, Datum) (Unterschrift)

VNUML-Installation im Windows-Pool an der Universität Koblenz-Landau

Inhaltsverzeichnis:

1. Zielsetzung

Die Motivation für diese Studienarbeit bestand darin, den Studierenden in den Rechnerpools der Uni-Koblenz die Möglichkeit zu geben, mit der Simulationssoftware VNUML[1] zu arbeiten.
Gearbeitet wird mit dieser Simulationssoftware in den Vorlesungen und Übungen zu Rechnernetzen I + II, was eine Anwendung der Software für die Studierenden unumgänglich macht. In der Vergangenheit gab es jedoch immer wieder Probleme bei der Installation und Einrichtung auf den privaten Rechnern, obwohl schon mehrfach vereinfachte Installationsroutinen in früheren Studienarbeiten entwickelt wurden. Ein weiteres Problem für die Verwendung von VNUML stellt auch die Tatsache dar, dass die Software nur in einer Linux-Umgebung lauffähig ist. Da aber nicht alle Studentinnen und Studenten das Betriebssystem Linux benutzen und vor einer Installation nur zur Verwendung von VNUML zurückschrecken, war es schon länger angedacht diese Software an den Rechner der Universität zur Verfügung zu stellen.
In dieser Arbeit wird nun der Prozess beschrieben, wie eine Installation der VNUML-Software in den Rechnerpools möglich war, welche Probleme dabei aufgetreten sind und welche Alternativen zur gewählten Vorgehensweise möglich gewesen wären.
Das Ergebnis bietet auch eine sehr einfache Installation für den privaten Anwender, ohne das hierfür eine eigenständige Linux-Installation nötig wäre. Auch wurden während der Entwicklung immer weitere Verbesserungen vorgenommen welche die Anwenderfreundlichkeit der endgültigen Lösung weiter optimierten. Die Möglichkeiten und Ideen sind dabei auch so vielfältig, dass sich die Arbeitsgruppe noch weiter mit diesem Thema beschäftigen wird und weitere Optimierungen vorgenommen werden können.

[1] Virtual Network User Mode Linux

2. Problemstellung

Bereits in der Vergangenheit war eine Installation der VNUML-Software in den Rechnerpools angedacht, was aber aus mehreren Gründen immer wieder scheitern musste. Bis einschließlich Version 1.5 von VNUML war es nicht möglich die Software mit eingeschränkten Benutzerrechten zu betreiben. Da man aber keinem Studenten so genannte „root"-Rechte auf einem Uni-Rechner im Rechnerpool zur Verfügung stellen kann, wäre eine einfache Installation der VNUML-Software auf das vorhandene Linux nicht möglich gewesen.
Mit Version 1.6 wurde es erstmals möglich VNUML auch mit eingeschränkten Benutzerrechten zu benutzen. Auch wenn hierbei einige Einschränkungen im Gegensatz zur gewohnten Nutzung der Simulationssoftware auftreten, kam mit dieser Möglichkeit erstmal wieder der Gedanke, eine Installation auf den Uni-Rechnern bereitzustellen. Und an dieser Stelle wurde mir als Ziel dieser Arbeit diese Aufgabe aufgetragen.
Jedoch kam im ersten Moment für mich eine Installation mit eingeschränkten Benutzerrechten nicht in Frage, da meines Erachtens damit eine nicht so effiziente Nutzung der Software möglich ist. Außerdem wäre es für die Studierenden eine Umstellung, wenn sie es von ihren privaten Rechnern gewohnt sind ohne Einschränkungen arbeiten zu können, an den Uni-Rechnern aber ein eingeschränktes System vorfinden.
Daher schaute ich, welche Alternativen sich zu dieser Variante bieten würden und in wie weit die vorhandenen Ressourcen die Anforderungen erfüllen würden. Neben der Leistung, welche die Rechner in den Rechnerpools besitzen, war besonders der Speicherplatz für eine Installation ein entscheidendes Kriterium. Eine Installation auf einem Netzlaufwerk, welche dann über das lokale Netzwerk gestartet wird wäre den Verantwortlichen im Rechenzentrum sehr entgegengekommen, da hier der Speicherplatz nahezu keine Rolle gespielt hätte. Wie sich aber herausstellte wäre dies nur bei einer Integration in die vorhandenen Linux-Systeme möglich gewesen. Als Alternative zu dieser, in den Benutzerrechten eingeschränkten, Variante gab es noch die Möglichkeit die VNUML Software in ein virtualisiertes Linux zu integrieren. Hierfür boten sich zwei Möglichkeiten, zum einen, eine Virtualisierung mit dem VM-Ware Player[2] und zum anderen die Verwendung von coLinux, einer speziell auf Linux-Systeme angepasste Virtualisierungsumgebung unter Windows-Systemen.
Welche Unterschiede es bei diesen drei Möglichkeiten zu beachten gibt wird im folgenden Kapitel ausführlich erläutert.

[2] Virtualisierungssoftware für nahe zu alle verfügbaren Systeme (http://www.vmware.com)

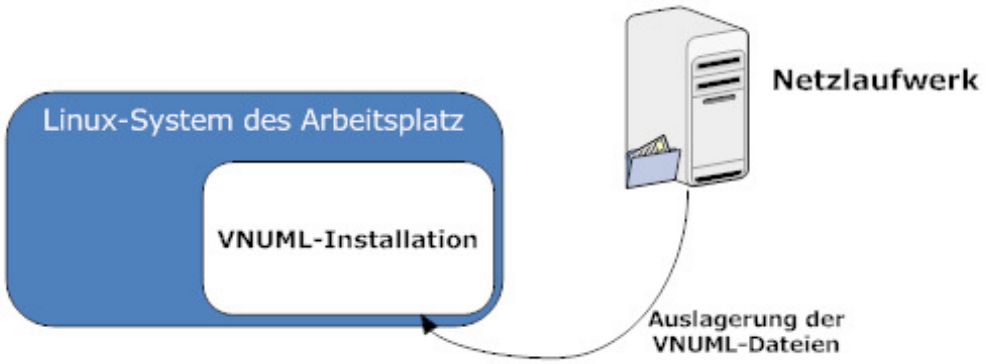
3. Die Möglichkeiten

Wie schon erwähnt, standen am Anfang meiner Überlegungen drei verschiedene Möglichkeiten zur Verfügung, um VNUML auf den Uni-Rechnern in den Rechnerpools zu installieren.

- Installation mit eingeschränkten Benutzerrechten im vorhandenen Linux-System
- VMWare Player mit einer Live-Version von VNUML
- coLinux[3] mit VNUML

3.1. Integration in das vorhandene Linux-System mit eingeschränkten Benutzerrechten

Dies wäre die einfachste Variante gewesen und hätte die wenigsten Ressourcen in Anspruch genommen, hätte aber den Nachteil mit sich gebracht, dass die Nutzer nur mit eingeschränkten Benutzerrechten unter VNUML arbeiten können.
Wie schon erwähnt, ist seit der VNUML Version 1.6 das Ausführen der Software auch in einem Linux-System ohne Root-Rechte möglich. Die Installation der Software wäre sehr einfach gewesen und hätte auch wenig Speicherplatz benötigt, da in diesem Fall eine Auslagerung der für VNUML benötigten Dateien auf ein Netzlaufwerk möglich gewesen wäre.



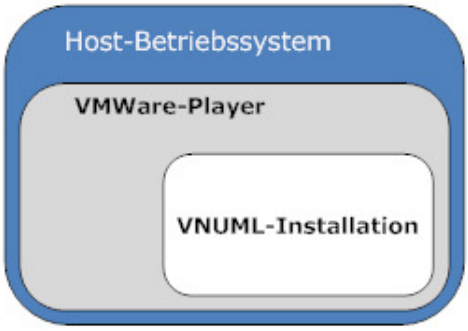
3.2 VMWare Player mit Live-VNUML-Image

Bereits in einer früheren Studienarbeit wurde eine LiveCD mit integrierter VNUML-Software entwickelt, welche für den Einsatz der VM-Ware Virtualisierungssoftware gedacht war. Inzwischen wird aber auch direkt von den Entwicklern von VNUML[4] eine LiveCD zum download angeboten, welche die neuste Version von VNUML beinhaltet. Die Grundlage für diese Art der Installation bietet der kostenlose VM-Player[5] der Firma VMWare Inc.
VM-Ware bietet für nahezu jede Plattform eine Virtualisierungssoftware an, so dass hierbei die freie Wahl bestanden hätte, unter welchem System die Installation am Ende erfolgt wäre.

---

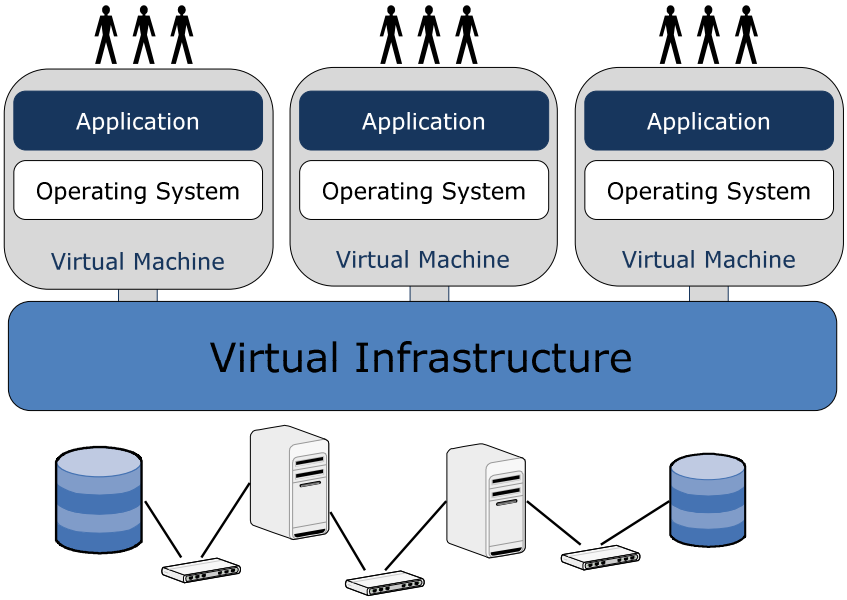
[3] Cooperative Linux, kurz coLinux - Virtualisierungssoftware für Linux-basierende Systeme unter Windows (http://www.colinux.org)
[4] Departamento de Ingeniería de Sistemas Telemáticos der Universidad Politécnica de Madrid (DIT-UPM) (http://www.dit.upm.es/vnumlwiki/)
[5] Nicht-kommerzielle Variante zum Betrieb einer VMWare Umgebung für den Desktopeinsatz (http://vmware.com/products/player/)

Die Idee, auf einer physischen Plattform, mehrere virtuelle Maschinen zu betreiben besitzt heutzutage eine zunehmend große Bedeutung. So ermöglicht diese Technik eine bessere Auslastung der vorhandenen Hardware und gibt den Benutzern die Möglichkeit mehrere unabhängige Systeme auf einem Rechner zu betreiben. Da diese virtuellen Maschinen völlig unabhängig voneinander betrieben werden, gewährleistet dies auch eine hohe Ausfallsicherheit. Stürzt eine virtuelle Maschine ab, arbeiten alle anderen unabhängig davon weiter. Eine Kommunikation mit anderen Systemen ist nur durch konfigurierte Netzwerkschnittstellen möglich.
Für die Installation der VNUML Software bedeutet dies, dass man so unter einem beliebigen Betriebssystem, in welchem man nur eingeschränkte Benutzerrechte besitzt, dem Benutzer trotzdem ein virtuelles System zur Verfügung stellen kann, auf welchem er uneingeschränkte Benutzerrechte genießt und somit eine VNUML-Installation ohne Beeinträchtigungen zur Verfügung hat.



Ein weiterer Vorteil bei der Verwendung einer Virtualisierungssoftware besteht darin, dass man ein vorhandenes VM-Ware-Image auf jedes beliebige System portieren kann, ohne weitere Anpassungen daran vorzunehmen. Der VM-Player verwaltet die benötigten Ressourcen automatisch und stellt in seiner virtuellen Maschine dem Image alles zur Verfügung was dieses benötigt. So wäre für jeden Rechner in den Uni-Pools nur ein einziges Image mit integrierter VNUML-Software zu erstellen gewesen, welches dann auf allen Rechnern gleichermaßen hätte genutzt werden können. Jedoch benötigt man für jede virtuelle Maschine eine eigene Image Datei, da eine gleichzeitige gemeinsame Nutzung eines Images nur von einer virtuellen Maschine erlaubt wird.

Ein Nachteil bei der Nutzung dieser Art der Virtualisierung ist jedoch der hohe Resourcenbedarf der virtuellen Infrastruktur selbst. Auch wenn ich vorhin eine optimale Ausnutzung vorhandener Hardware angesprochen habe, gilt dies nicht für den Einsatz im Desktop-Bereich. Der angesprochene Vorteil besteht in erster Linie bei leistungsstarken Serversystemen wobei sogar eine Auslagerung der Ressourcen über eine Netzwerkstruktur erfolgen kann.



Verwendet man den VMWare-Player jedoch auf einem Desktop-System, so werden die hier doch eher knapp bemessenen Hardwareeigenschaften, sehr beansprucht. So benötigt die Installation der Virtualisierungssoftware bereits über 200 Megabyte Festplattenspeicher auf dem Host-System und zusätzlich nochmals über ein Gigabyte für das Image des Linux-Systems mit integrierter VNUML-Installation. Verglichen mit einer Integration der VNUML Software in ein vorhandenes Linux-System wäre dies also etwa der doppelte Speicherplatz.
Viel entscheidender ist jedoch der auftretende Leistungsverlust auf einem Desktop-System bei der Verwendung des VMWare-Players. Betreibt man eine virtuelle Maschine auf einem einfachen Rechner benötigt dieser natürlich Ressourcen um das Host-System zu betreiben. Mit dem VM-Player werden weitere Ressourcen in Anspruch genommen, um den virtuellen Maschinen eine Infrastruktur bereitzustellen, auf welche diese zurückgreifen können. Und natürlich beansprucht die virtuelle Maschine, welche man mit dem VM-Player betreibt, nochmals die gleichen Ressourcen wie im Falle eines direkten Hardwarezugriffs.
So werden die vorhandenen Kapazitäten eines Systems bereits vor dem Start eines VNUML-Szenarios sehr in Anspruch genommen, wobei man nicht vergessen darf, dass ein solches Szenario dann wiederum eine virtuelle Umgebung aufbaut, in welcher jede simulierte Maschine die Teil dieses Szenarios ist, wiederum eine solche virtuelle Maschine darstellt wie das zugrunde liegende Systems selbst.
Dies führt dazu, dass bereits kleine Szenarien einer VNUML-Umgebung einen hohen Anspruch an die Hardware stellen und diese schnell an Ihre Grenzen bringen. So bekommt man z.B. auf einem Pentium Mobile System mit 1,6GHz Rechenleistung und 1,5GByte Arbeitsspeicher schon ein Timeout beim Starten eines Szenarios mit nur 5 virtuellen Maschinen.
Schlussendlich war dieser Umstand das maßgebliche Kriterium um diese Variante nicht zu verwenden, da die Anforderung auch komplexere Systeme zu betreiben gewährleistet sein sollte.
Als weiteren Kritikpunkt könnte man auch die etwas aufwendige Installation ansehen. Um einen Datenaustausch zwischen Host-System und virtueller Umgebung zu ermöglichen ist es beim VM-Ware-Player notwendig manuell eine Netzwerkschnittstelle zu konfigurieren, über welche dann der Datenaustausch erfolgen kann.

3.3 coLinux und VNUML

Die dritte Variante, zwischen welcher die Entscheidung zur Realisierung dieses Projekts gefallen ist, bietet die Installation einer anderen Virtualisierungssoftware. Hierbei handelt es sich um coLinux, was speziell dafür entwickelt wurde, um ein Linux-System auf einem Windows-System zu emulieren. Unterstützt werden dabei die 32-Bit Versionen 2000, XP und auch Vista. Cooperative Linux ist eine frei verfügbare Software, welche die Ressourcen eines Desktop PC's viel besser ausnutzt als das im vorherigen Kapitel beschriebene VM-Ware-System. Diese Aussage soll jedoch keine Abwertung der VM-Ware-Software darstellen, da diese im Gegensatz zu coLinux ein viel breiteres Anwendungsspektrum bietet, wogegen coLinux wirklich nur für die Bedürfnisse eines auf Linux basierenden Betriebssystem entwickelt wurde das unter Windows verwendet werden kann.
Wie im vorherigen Kapitel beschrieben benutzt die VM-Ware eine virtuelle Infrastruktur, welche die Hardware selbst simuliert. Bei coLinux jedoch findet keine wirkliche Virtualisierung der Hardware statt, sondern die Befehle des virtuellen Systems werden mittels spezieller Treiber auf Windows-Prozesse umgewandelt. So wird der Kernel des coLinux System in einem privilegierten Modus ausgeführt (auch bekannt als supervisor mode oder ring 0 Modus).
Zusätzlich wird der Zustand der physikalischen Maschine ständig zwischen dem des Host-Systems und des virtuellen Systems gewechselt, so dass die virtuelle Maschine auch die volle Kontrolle über die MMU (memory management unit) in ihrem eigenen Adressraum besitzt. Damit wird eine Performance erreicht, genügend Arbeitsspeicher vorrausgesetzt, welche nahezu die identische Geschwindigkeit eines eigenständigen Linux System erreicht.

3.4. Zusammenfassung

	Integration in das Linux-System der Arbeitsplätze	**VM-Ware Player mit VNUML-Live Image**	**coLinux mit inte-- grierter VNUML-Installation**
Speicherplatz	geringste Belastung der lokalen Festplatten, die für VNUML benötigten Dateien können auf einem Netzlaufwerk abgelegt werden	große Belastung durch Installation des VM-Ware Player + Live-Image mit integrierter VNUML Umgebung	geringe Belastung durch coLinux, aber große Belastung durch die Image-Datei der virtuellen Maschine
Performance	die Ressourcen werden optimal Ausgenutzt	sehr schlecht, da großer Ressourcenbedarf für Virtualisierung	sehr gut, da coLinux die Ressourcen optimal verwaltet
Installation	zentral vom Rechenzentrum möglich	umständlich, Installation manuell auf jedem Rechner mit eigener Konfiguration notwendig	aufwendig aber einfach, Installation manuell auf jedem Rechner notwendig
Benutzer-freundlichkeit	VNUML nur mit eingeschränkten Benutzerrechten benutzbar	VNUML uneigenschränkt nutzbar, jedoch oftmals timeout wegen hohem Resourcenbedarf	VNUML uneingeschränkt und nahezu optimal nutzbar

4. Installationsmöglichkeiten von coLinux

Nachdem die Entscheidung gefallen war, coLinux in den Rechnerpools zum Einsatz zu bringen, um die Verwendungung der VNMUL-Software dort zu ermöglichen, suchte ich nach einer Möglichkeit den einzigen Nachteil dieser Variante zu verringern. Dieser Nachteil bezog sich auf die 2GByte große Image-Datei[6] der VNUML-Entwickler, welche die Idee für diese Installationsvariante hervorbrachte.
Die Mitarbeiter des Rechenszentrums wollten es vermeiden, die lokalen Festplatten der Arbeitsplätze mit einer solch großen Datenmenge zu belasten, da hier der verfügbare Speicherplatz eher knapp bemessen ist.
Es galt also nach einer Möglichkeit zu suchen, die Kapazitäten der einzelnen Rechner möglichst wenig zu beanspruchen, wofür ich mir mehrere Varianten überlegte:

- **eine einzelne Imagedatei** wird auf einem Netzlaufwerk abgelegt und von allen Arbeitsplätzen gestartet

- **alle Imagedateien** werden auf einem Netzwerklaufwerk abgelegt und die Arbeitsplätze besitzen exklusiven Zugriff auf ein Image

- für die Nutzungsdauer der Software wird eine **temporäre Kopie** der Imagedatei auf einem Netzlaufwerk auf der **lokalen Festplatte** erstellt

- es werden **temporäre Kopien** auf dem **Netzlaufwerk** erzeugt, welche von coLinux verwendet werden können

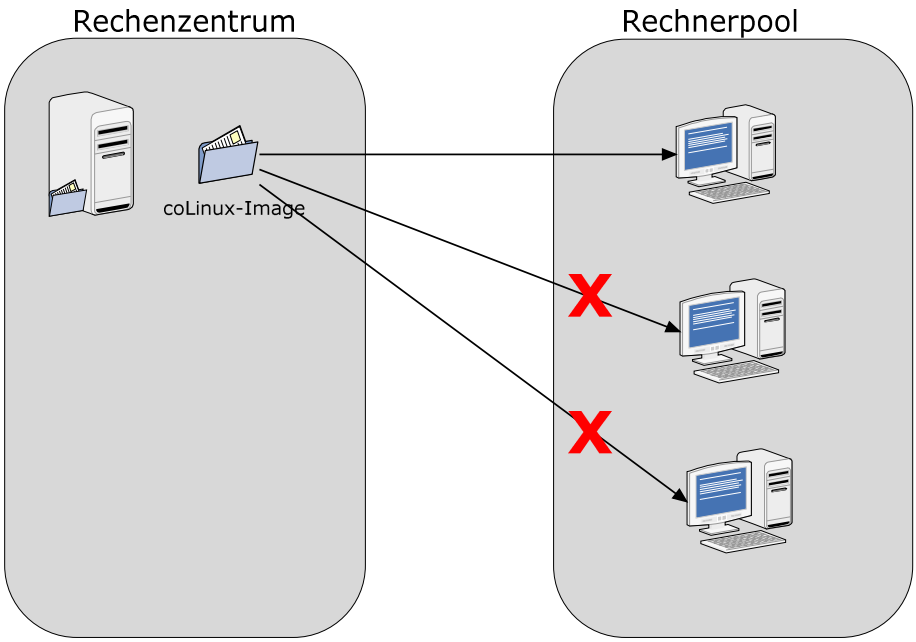
- die Imagedatei wird bestmöglich verkleinert und auf den **lokalen Festplatten** der Arbeitsplätze abgelegt

In den folgenden Abschnitten werden diese unterschiedlichen Möglichkeiten genauer vorgestellt und auf ihre Praxistauglichkeit hin untersucht.

4.1 Eine einzelne Imagedatei auf einem Netzlaufwerk

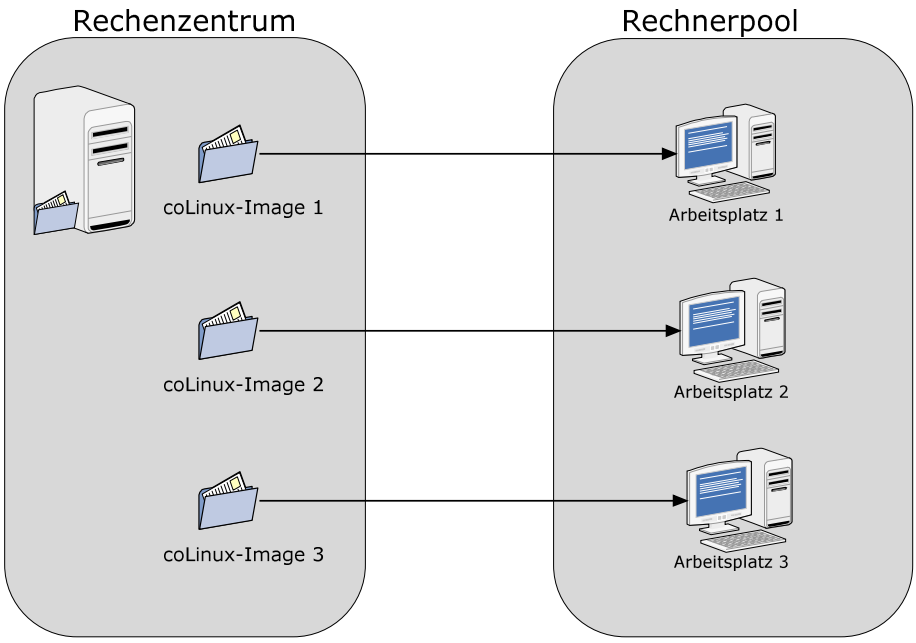
Diese Variante wäre optimal gewesen, da hier die kleinste Belastung der verfügbaren Ressourcen aufgetreten wäre. Wie meine Tests ergaben, ist ein Starten der coLinux-Imagedatei ohne essentiellen Leistungsverlust über ein 100Mbit/s Netzwerk möglich. Jedoch ist die Nutzung einer Imagedatei immer auf eine Sitzung beschränkt, d.h. wird die Imagedatei von einer virtuellen Maschine gestartet, besitzt diese exklusiven Zugriff darauf und verweigert anderen virtuellen Maschinen den Zugriff.

[6] Eine einzelne Datei welche sowohl Inhalt als auch Datenstruktur eines ganzen Datenspeicher enthält. Hier der Inhalt einer Partition mit einem Linux-System.



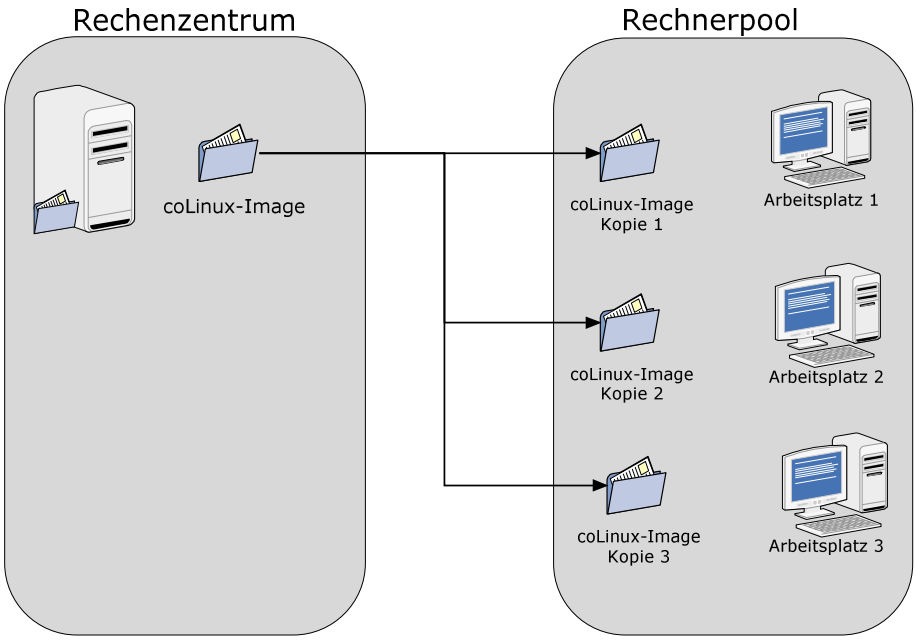
4.2 Alle Imagedateien auf einem Netzlaufwerk

Nachdem die im vorherigen Abschnitt vorgestellte Variante verworfen werden musste, kam mir der Gedanke, man könne für jeden Arbeitsplatz eine Imagedatei auf einem Netzwerklaufwerk ablegen, um so die lokalen Festplatten zu schonen. Theoretisch wäre dies auch zu realisieren, jedoch hätte dies eine sehr aufwendige Installation zu Folge gehabt, da jeder Imagedatei eine individuelle Konfigurationsdatei zur Verfügung gestellt werden muss die auf das Image verweist. Dabei hätte jegliche Änderung im Netzwerk dazu geführt, dass die Konfigurationsdateien der coLinux Imagedatei angepasst werden müsste.



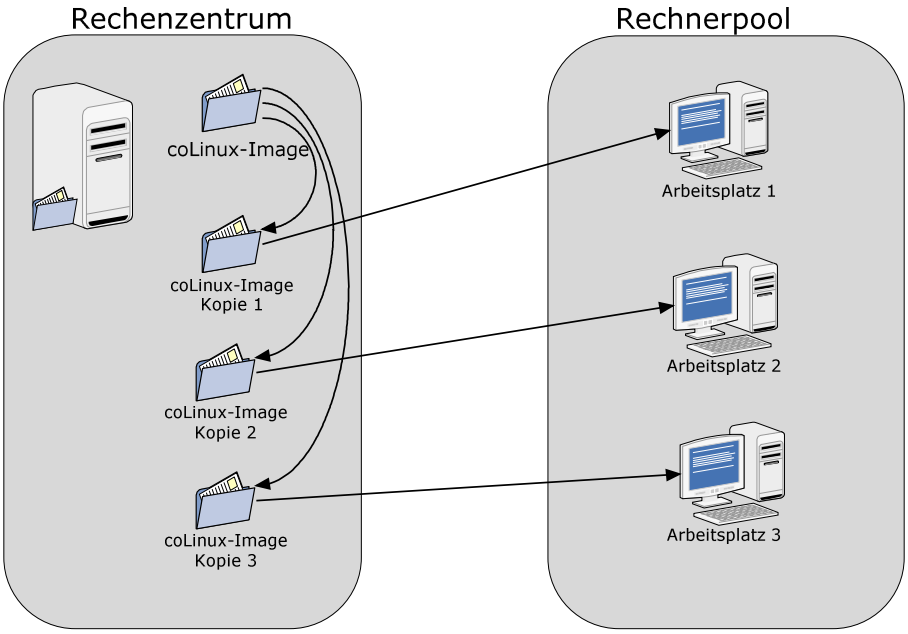
4.3 Erstellen einer temporären Kopie auf dem Arbeitsplatz vom Netzlaufwerk

Bei dieser Variante hätte ein kleines Programm dafür gesorgt, dass aus dem Netzwerk eine Kopie der Imagedatei auf dem Arbeitsplatz erstellt wird. Diese Kopie hätte der coLinux Umgebung des Rechners dann exklusiv für die Nutzungsdauer der VNUML Software zur Verfügung gestanden und hätte nicht dauerhaft die lokale Festplatte belastet.
Diese Möglichkeit hätte jedoch eine sehr große Verzögerung beim Start der Software zur Folge gehabt, denn selbst eine verkleinerte Imagedatei mit einer Größe von ~1,5GByte würde bei optimalen Vorraussetzungen noch ca. 2 Minuten benötigen um über das Netzwerk auf den lokalen Rechner kopiert zu werden. Da aber davon ausgegangen werden muss, dass z.B. in einer Übungsstunde der Rechnernetze-Vorlesung, mehrere Rechner gleichzeitig das Netzwerk mit diesem Kopiervorgang belasten würden, würde sich die Ladezeit noch um ein vielfaches erhöhen. Die daraus resultierenden Wartezeiten zum Starten der virtuellen Maschinen machten diese Möglichkeit daher zunichte.



4.4. Erstellen einer Kopie der Imagedatei auf dem Netzlaufwerk selbst

Die Idee, eine temporäre Kopie der Imagedatei zu erstellen, wurde damit aber noch nicht direkt verworfen. Als nächstes zog ich die Möglichkeit in betracht, auf einem Server im Netzwerk eine Kopie der Imagedatei zu erstellen, auf welche die coLinux Umgebung hätte zugreifen können.
Um dies mit der gewünschten Effektivität zu realisieren hätte jedoch ein Programm direkt auf dem Server installiert werden müssen, welches wiederum von einem Arbeitsplatz auf einem Rechner im Rechnerpool angestoßen werden müsste. Die Problematik der wechselnden Pfadangaben im Netzwerk stellte sich jedoch nicht, denn leider musste dieser Ansatz aus einem anderen Grund verworfen werden. Den Mitarbeitern im Rechenzentrum war das Sicherheitsrisiko zu groß, ein Programm auf einem Rechner laufen zu lassen, welches im Falle eines Missbrauchs ungeahnt große Datenmengen auf deren Festplatten verursachen könnte.



4.5. Imagedatei auf den lokalen Festplatten der Arbeitsplätze

Da alle anderen Möglichkeiten, die Festplatten der Arbeitsplatzrechner zu schonen, verworfen werden mussten, blieb am Ende nicht anderes übrig als die Image Dateien auf den lokalen Platten abzulegen.
Eigentlich sollte diese Belastung der lokalen Ressourcen ja vermieden werden, jedoch blieb dies als einzige Variante übrig, die effektiv zu realisieren war. Um jedoch die Festplatten nicht mit einer allzu großen Imagedatei zu belasten, musste dafür gesorgt werden, das dass Image auf die kleinstmögliche Größe gebracht wird. Der genaue Inhalt und eine Beschreibung der Imagedatei finden Sie im nächsten Kapitel.

5. Das Archiv „VNUMLcolinux.zip“

5.1. Die Image-Datei

Nachdem nun also die Imagedatei auf den lokalen Festplatten der Computer im Rechnerpool abgelegt werden sollte, musste darauf geachtet werden das dies mit einer möglichst geringen Belastung des Speichervolumens verbunden ist.
Als Grundlage für die später verwendete Imagedatei diente das Image für coLinux, welche von den VNUML-Entwicklern zur Verfügung gestellt wird[7]. In dieser Version der coLinux-Umgebung befinden sich folgende Dateien:

- colinux-vnuml-1.0.conf
- start.bat
- Debian-4.0r0-etch.ext3-vnuml.2gb

Bei den ersten beiden Dateien handelt es sich um eine Konfigurations- bzw. Batchdatei, welche zum Starten der coLinux Umgebung benötigt werden. In diesem Abschnitt liegt der Fokus jedoch auf der Datei Debian-4.0r0-etch.ext3-vnuml.2gb
Diese Imagedatei ist jedoch mehr als ein reines Abbild eines Speichervolumens. Es handelt sich dabei um eine so genannte Container-Datei, welche nicht nur die eigentlichen Dateien des Linux-Systems beinhaltet, sondern zusätzlich noch Speicherplatz zum Arbeiten in der virtuellen Umgebung reserviert.
Anhand des Dateinamens lässt sich erkennen, dass es sich hierbei um das Image einer Debian-Linuxdistribution handelt, welche auf einem ext3-Filesystem installiert wurde. Die Verwendung des ext3-Filesystems ist sinnvoll, da es sich hierbei um ein sogenanntes Journal-Dateisystem handelt, welches jede Dateiänderung protokolliert. Der Vorteil dabei ist, dass im Falle eines Systemabsturz nicht das ganze Image und somit das Dateisystem überprüft werden muss und so ein erneuter schneller Systemstart gewährleistet wird.
Desweiteren lässt die Dateiendung auf eine 2GByte große Datei schließen, was sich auch bei einem genauen Blick in das Verzeichnis bestätigt.

```
root@amdx2:~/colinux-vnuml-1.0# ls -l -h
insgesamt 2,1G
-rw-r--r-- 1 mario mario 1,7K 2008-02-24 02:41 colinux-vnuml-1.0.conf
-rw-r--r-- 1 mario mario 2,0G 2008-02-24 03:34 Debian-4.0r0-etch.ext3-
vnuml.2gb
-rw-r--r-- 1 mario mario   45 2008-02-23 21:08 start.bat
root@amdx2:~/colinux-vnuml-1.0#
```

[7] ftp://ftp.dit.upm.es/vnuml/colinux-vnuml-1.0.zip

Um die Auslastung dieser Image Datei festzustellen wird zuerst ein Verzeichnis `/mnt/image/` erstellt, in welches das Image anschließend gemountet wird.

```
root@amdx2:~/colinux-vnuml-1.0# mkdir /mnt/image
root@amdx2:~/colinux-vnuml-1.0# mount -o loop Debian-4.0r0-etch.ext3-
vnuml.2gb  /mnt/image/
root@amdx2:~/colinux-vnuml-1.0# df -h
Dateisystem            Größe Benut  Verf Ben% Eingehängt auf
/dev/sdb7              276G   12G  250G   5% /
tmpfs                 1013M     0 1013M   0% /lib/init/rw
varrun                1013M  104K 1012M   1% /var/run
varlock               1013M     0 1013M   0% /var/lock
udev                  1013M  2,9M 1010M   1% /dev
tmpfs                 1013M  404K 1012M   1% /dev/shm
lrm                   1013M  2,0M 1011M   1% /lib/modules/2.6.27-11-
generic/volatile
/dev/loop0             2,0G  1,1G  844M  56% /mnt/image
root@amdx2:~/colinux-vnuml-1.0#
```

Wie man in der letzten Zeile erkennt beträgt die relative Auslastung nur 56%, da von den reservierten 2,0 GByte nur etwa 1,1GByte benutzt werden. Derart viel Speicherplatz zu reservieren ist an dieser Stelle jedoch ineffizient und da das Ziel darin besteht, eine möglichst geringe Belastung des Speichervolumens zu verursachen, scheint es sinnvoll, die vorhandene Imagedatei zu verkleinern.
Um dies zu erreichen wird zunächst eine neue, leere Containerdatei mit dem Namen `VnumlImage.img` erzeugt. Dies geschieht unter Verwendung des Befehls `dd` (diskdump) mit den Parameter `bs=2k`, welcher eine Blockgröße von 2KByte beschreibt und `count=750k`, was dafür sorgt, dass 750.000 leere Blöcke in die neue Imagedatei geschrieben werden. Das Ergebnis ist demnach ein 1,5GByte großes, neues und noch leeres Image.

```
root@amdx2:~/colinux-vnuml-1.0# dd if=/dev/zero of=/home/mario/colinux-
vnuml-1.0/VnumlImage.img bs=2k count=750k
768000+0 Datensätze ein
768000+0 Datensätze aus
1572864000 Bytes (1,6 GB) kopiert, 19,4765 s, 80,8 MB/s
root@amdx2:~/colinux-vnuml-1.0# ls -l
insgesamt 3636716
-rw-r--r-- 1 mario mario       1728 2008-02-24 02:41 colinux-vnuml-
1.0.conf
-rw-r--r-- 1 mario mario 2147483648 2008-02-24 03:34 Debian-4.0r0-
etch.ext3-vnuml.2gb
-rw-r--r-- 1 mario mario         45 2008-02-23 21:08 start.bat
-rw-r--r-- 1 root  root  1572864000 2008-08-12 14:38 VnumlImage.img
root@amdx2:~/colinux-vnuml-1.0#
```

Um diese Imagedatei auch benutzen zu können, wird zuerst das Verzeichnis `/mnt/newimage/` erstellt, in welches später das Image gemountet wird. Damit dies jedoch überhaupt möglich ist, muss der neue Container zuvor mit dem ext3-Dateisystem formatiert werden.

```
root@amdx2:~/colinux-vnuml-1.0# mkdir /mnt/newimage/
root@amdx2:~/colinux-vnuml-1.0# mkfs.ext3 VnumlImage.img
mke2fs 1.41.3 (12-Oct-2008)
VnumlImage.img ist kein spezielles Block-Gerät.
Trotzdem fortsetzen? (j,n) j
Dateisystem-Label=
OS-Typ: Linux
Blockgröße=4096 (log=2)
Fragmentgröße=4096 (log=2)
96000 Inodes, 384000 Blöcke
19200 Blöcke (5.00%) reserviert für den Superuser
Erster Datenblock=0
Maximale Dateisystem-Blöcke=394264576
12 Blockgruppen
32768 Blöcke pro Gruppe, 32768 Fragmente pro Gruppe
8000 Inodes pro Gruppe
Superblock-Sicherungskopien gespeichert in den Blöcken:
      32768, 98304, 163840, 229376, 294912

Schreibe Inode-Tabellen: erledigt
Erstelle Journal (8192 Blöcke): erledigt
Schreibe Superblöcke und Dateisystem-Accountinginformationen: erledigt

Das Dateisystem wird automatisch nach jeweils 36 Einhäng-Vorgängen bzw.
alle 180 Tage überprüft, je nachdem, was zuerst eintritt. Veränderbar mit
tune2fs -c oder -t .
root@amdx2:~/colinux-vnuml-1.0# mount -o loop VnumlImage.img
/mnt/newimage
root@amdx2:~/colinux-vnuml-1.0#
```

Sieht man sich nun die Auslastung der eingehängten Dateisysteme an, sieht man die beiden Imagedateien an letzter Position mit den unterschiedlichen Größen und Benutzungsgraden.

```
root@amdx2:~/colinux-vnuml-1.0# df -h
Dateisystem            Größe Benut  Verf Ben% Eingehängt auf
/dev/sdb7              276G   13G  249G   5% /
tmpfs                 1013M     0 1013M   0% /lib/init/rw
varrun                1013M  104K 1012M   1% /var/run
varlock               1013M     0 1013M   0% /var/lock
udev                  1013M  2,9M 1010M   1% /dev
tmpfs                 1013M  404K 1012M   1% /dev/shm
lrm                   1013M  2,0M 1011M   1% /lib/modules/2.6.27-11-
generic/volatile
/dev/loop0             2,0G  1,1G  844M  56% /mnt/image
/dev/loop1             1,5G   35M  1,4G   3% /mnt/newimage
root@amdx2:~/colinux-vnuml-1.0#
```

Nun gilt es die Dateien aus der ursprünglichen Imagedatei in die neue zu übertragen, was mit einem einfachen copy-Befehl und den Parametern `-avx` realisiert wird.

```
root@amdx2:~/colinux-vnuml-1.0# cp -avx /mnt/image/* /mnt/newimage/
…
root@amdx2:~/colinux-vnuml-1.0# df -h
Dateisystem            Größe Benut  Verf Ben% Eingehängt auf
/dev/sdb7              276G   13G  249G   5% /
tmpfs                 1013M     0 1013M   0% /lib/init/rw
varrun                1013M  104K 1012M   1% /var/run
varlock               1013M     0 1013M   0% /var/lock
udev                  1013M  2,9M 1010M   1% /dev
tmpfs                 1013M  404K 1012M   1% /dev/shm
lrm                   1013M  2,0M 1011M   1% /lib/modules/2.6.27-11-
generic/volatile
/dev/loop0             2,0G  1,1G  844M  56% /mnt/image
/dev/loop1             1,5G  1,1G  365M  75% /mnt/newimage
root@amdx2:~/colinux-vnuml-1.0#
```

Eine anschließende erneute Überprüfung der Speicherplatzauslastung mit `df` (diskfree) zeigt, dass die neue Containerdatei nun ebenfalls 1,1GByte Daten enthält, mit einer Auslastung von 75% jedoch nur 1,5GByte reserviert und somit weniger Speicherplatz unbenutzt lässt. Auch wenn es möglich wäre, das Image noch weiter in seiner Größe zu reduzieren wurde bewusst darauf verzichtet, da später dem System dieser ungenutzte Speicherplatz als Arbeitsvolumen dient.

Die verkleinerte Imagedatei wurde also dadurch geschaffen, indem eine neue, zuerst leere Datei erzeugt wurde und anschließend der komplette Inhalt der vorhandenen Containerdatei in die neu erzeugte Datei kopiert wurde. Die neue Imagedatei belegt mit etwa 1,5GByte nur noch 75% des Speicherplatzes gegenüber der vorhandenen Datei, kann aber gleichwertig unter coLinux verwendet werden.

5.2. Installation des deutschen Tastatur-Layout

Nachdem nun eine verkleinerte Imagedatei erstellt wurde, muss diese noch weiter angepasst werden um eine komfortable Nutzung der VNUML-Software zu ermöglichen.
Da die späteren virtuellen Netzwerkumgebungen ausschließlich über die Konsole gesteuert werden und dies er Tastatureingabe erfolgt, ist es an dieser Stelle sinnvoll der Linux-Installation in der Imagedatei ein deutsches Tastaturlayout zu verpassen.
Um die fehlenden Packte für eine Installation herunterzuladen muss der coLinux-Umgebung als erstes der Zugriff auf das Internet ermöglicht werden. Dafür muss in der conf-Datei die Netzwerkschnittstelle umkonfiguriert werden, so dass anstelle der TUN/TAP[8]- eine SLIRP[9]-Schnittstelle verwendet wird, welche der coLinux Umgebung den Zugriff auf das Internet erlaubt.

[8] TUN und TAP sind virtuelle Netzwerk-Kerneltreiber, die Netzwerkgeräte über Software simulieren
[9] SLIRP ist ein TCP/IP-Emulator, mit dessen Hilfe über ein normales Terminal PPP-Dienste ohne wirkliche Einwahl verwendet können

```
…
# Slirp for internet connection (outgoing)
# Inside running coLinux configure eth0 with this static settings:
# ipaddress 10.0.2.15   broadcast  10.0.2.255   netmask 255.255.255.0
# gateway   10.0.2.2    nameserver 10.0.2.3
eth0=slirp

# Tuntap as private network between guest and host on second linux
device
#eth0=tuntap
```

Startet man die coLinux Umgebung nun mit dieser geänderten Konfigurationsdatei lassen sich die fehlenden Pakete aus dem Internet herunterladen. Für die Installation er neuen Sprachpakete wird der Befehl

```
debian:~# apt-get install console-data console-common console-tools
```

verwendet.
Dadurch werden alle benötigten Pakete erst heruntergeladen und installiert. Im Anschluss daran öffnet sich automatisch die Konfigurationsroutine um das gewünschte Tastaturlayout auszuwählen. In den einzelnen Schritten wird für das deutsche Layout dabei folgende Auswahl getroffen

```
=> Exit
      => Select keymap from arch list
            => qwertz
                  => German
```

Anschließend werden noch automatisch die Einstellungen die die entsprechenden Dateien geschrieben und das deutsche Tastaturlayout ist sofort verfügbar.

5.3. SWAP-Partition für leistungsschwache Systeme

Das von der DIT-UPM zur Verfügung gestellte Archiv beinhaltet noch keine Auslagerungsdatei, die sogenannte Swap-Partition, um auch auf leistungsschwachen Systemen die VNUML-Umgebung unter coLinux nutzen zu können. In der ursprünglichen Variante wurde nur der physisch vorhandene Arbeitsspeicher dazu verwendet, um die virtuellen Maschinen der VNUML Umgebung auszulagern. Mit Hilfe einer Swap-Partion, welche der coLinux Umgebung zur Verfügung gestellt werden kann, wird dem Linux-System die Möglichkeit gegeben, einen Teil der Festplatte als Arbeitsbereich zu verwalten.
Die Vorgehensweise, um eine solche Image-Datei zu erstellen, ist dabei ähnlich dem Verfahren, mit welchem bereits das Image der eigentlichen coLinux-Umgebung erstellt wurde. Es wird erneut eine Sparsedatei mit Hilfe des dd-Befehls erzeugt, welche nun aber als SWAP-Partition formatiert wird.

```
root@amdx2:~/colinux$ dd if=/dev/zero of=swap.img bs=2k count=512k
524288+0 Datensätze ein
524288+0 Datensätze aus
1073741824 Bytes (1,1 GB) kopiert, 10,9538 s, 98,0 MB/s
root@amdx2:~/colinux$ mkswap swap.img
Setting up swapspace version 1, size = 1048572 KiB
kein Label, UUID=e2cb5679-5316-464e-9eb2-257a16f50e29
root@amdx2:~/colinux$
```

Damit die coLinux Umgebung diese Datei auch als Swap-Partition nutzen kann, ist ein ensprechender Eintrag in der Datei `/etc/fstab` der virtellen Linux-Maschine nötig. Ein Blick in diese Datei zeigt, dass der benötigte Eintrag bereits vorhanden ist und beim Starten der coLinux-Umgebung nur noch den richtige Parameter übergeben werden muss, um unsere erzeugte Image-Datei als Swap-Partition nutzen zu können.

```
debian:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/cobd0      /               ext3    noatime,errors=remount-ro 0
1
/dev/cobd1       none            swap    sw              0       0
#/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto     0       0
debian:~#
```

Die coLinux-Umgebung hat unter dem Gerätenamen `/dev/cobd1` eine Swap-Partition definiert. Damit die erzeugte Sparse-Datei auch als solche genutzt werden kann, muss der coLinux-Umgebung beim Start diese Datei nur als Gerät cobd1 übergeben werden. Dies geschieht einfach in der Konfigurationsdatei, mit welcher die coLinux Umgebung aufgerufen wird, mit der Zeile:

```
cobd1="swap.img"
```

Da sowohl in der Konfigurations-Datei als auch in der fstab[10] des Linux-Systems die Verwendung der Swap-Partition aktiviert ist, erscheint eine Fehlermeldung beim Starten der coLinux-Umgebung wenn die erforderliche Image-Datei nicht bereitgestellt wird. Dies kann jedoch ignoriert werden und führt zu keiner weiteren Beeinträchtigung.

[10] Die Datei fstab enthält eine Liste aller beim Start zu ladenden Dateisysteme

5.4. Die Konfigurationsdatei für coLinux

Als Grundlage für die spätere Konfigurationsdatei diente auch hier die ursprüngliche conf-Datei aus dem Archiv der DIT-UPM.

```
…
# Full list of config params is listed in colinux-daemon.txt.

[1]
# The default kernel
kernel=..\vmlinux

[2]
# File contains the root file system.
# Download and extract preconfigured file from SF "Images for 2.6".
# cobd0="d:\coLinux\root_fs"
cobd0="Debian-4.0r0-etch.ext3-vnuml.2gb"

[3]
# Swap device, should be an empty file with 128..512MB.
#cobd1="c:\coLinux\swap_device"

[4]

# Tell kernel the name of root device (mostly /dev/cobd0,
# /dev/cobd/0 on Gentoo)
# This parameter will be forward to Linux kernel.
root=/dev/cobd0

# Additional kernel parameters (ro = rootfs mount read only)
ro

# Initrd installs modules into the root file system.
# Need only on first boot.
initrd=..\initrd.gz

# Maximal memory for linux guest
#mem=64

# Slirp for internet connection (outgoing)
# Inside running coLinux configure eth0 with this static settings:
# ipaddress 10.0.2.15   broadcast  10.0.2.255   netmask 255.255.255.0
# gateway   10.0.2.2    nameserver 10.0.2.3
#eth0=slirp

[5]
# Tuntap as private network between guest and host on second linux device
#eth1=tuntap
eth0=tuntap

# Setup for serial device
#ttys0=COM1,"BAUD=115200 PARITY=n DATA=8 STOP=1 dtr=on rts=on"

[6]
# Run an application on colinux start (Sample Xming, a Xserver)
#exec0=C:\Programs\Xming\Xming.exe,":0 -clipboard -multiwindow -ac"
```

Zu [1]: An dieser Stelle wird der Pfad zum Kernel festgelegt. Da bei der Installation der VNUML-Entwickler vorgegeben wird, das Archiv in einem Unterordner des coLinux-Stammverzeichnis zu entpacken, finden wir hier eine relative Pfadangabe zu dessen übergeordnetem Verzeichnis, welche den Speicherort des Kernel angibt.
Um jedoch eine höhere Flexibilität zu erreichen und die virtuelle Maschine später auch von einem USB-Stick starten zu können, wurden die kompletten coLinux –Systemdateien in das Archiv der coLinux-VNUML-Umgebung der Uni-Koblenz integriert. Relativ zum Verzeichnis, in welchem das Archiv entpackt wurde, finden sich diese Dateien im Ordner `.\colinux\colinuxsystemdata\`.
Dementsprechend musste die Angabe des Kernels wie folgt angepasst werden:

```
[1]
# The default kernel
kernel=colinux\kernel\vmlinux
```

Zu [2]: Hier erfolgt die Angabe der Image-Datei. Nachdem man das Archiv der VNUML-Entwickler entpackt hat, findet man Start-, Konfigurations- und Imagedatei im gleichen Verzeichnis, daher genügt es in deren Konfigurationsdatei an dieser Stelle nur den Namen des Image anzugeben.
Aus strukturellen Gründen wurde im neuen Archiv die Imagedatei im Verzeichnis `.\colinux\filesystem\` abgelegt und, wie schon im vorherigen Abschnitt beschrieben, umbenannt. Der neue Pfad zur Imagedatei lautet in diesem Archiv daher:

```
[2]
# File contains the root file system.
cobd0="colinux\filesystem\VnumlImage.img"
```

Zu [3]: Im vorherigen Abschnitt wurde die Erstellung einer zusätzlichen Image-Datei beschrieben, welche als SWAP-Partition integriert werden kann. Diese Datei liegt im Verzeichnis `.\colinux\swapfile\` und wird beim Start als Gerät cobd1 übergeben.

```
[3]
# Swap device, should be an empty file with 512..1024MB.
cobd1="colinux\swapfile\swap.img"
```

Zu [4]: Um eine einfache und direkte Möglichkeit zu haben, Dateien zwischen Hostsystem und virtueller Maschine auszutauschen, kann ein sogenanntes Co-Filesystem definiert werden. Dabei handelt es sich einfach um einen Ordner, welcher sowohl einen Zugriff aus der Windows-Oberfläche als auch aus der virtuellen Maschine zulässt. Dateien die im Host-System in den hier definierten Ordner kopiert werden, stehen später unter `/datatransfer` der coLinux-Umgebung zur Verfügung.

```
[4]
# cofs device, to transfer data between coLinux and windows
cofs0="datatransfer"
```

Zu [5]: Um eine Verbindung zwischen Host-Betriebsystem und der coLinux-Umgebung herzustellen, wird an dieser Stelle eine Netzwerkverbindung angegeben, welche diese Aufgabe erfüllt. Dazu wird hier ein virtueller Netzwerkadapter verwendet, welcher bei der Installation von coLinux eingerichtet wird. Eine Änderung der bestehenden Einstellung war hierbei nicht nötig, die Konfiguration dieser Schnittstelle muss jedoch bei einer Installation von coLunix angepasst werden. Eine genaue Beschreibung hierzu befindet sich im nächsten Kapitel im Abschnitt „Installation von coLinux und der VNUML-Umgebung“.

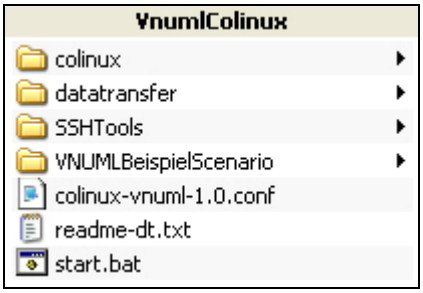
Zu [6]: Auf der Internetseite der DIT –UPM[11] wird zusätzlich noch die Installation eines X-Window Servers unter Windows erläutert. In unserem Fall haben wurde hierauf jedoch bewusst verzichtet, da keine weitere Programminstallation im Windowssystem erwünscht gewesen wäre.
Der Vorteil einer Installation eines X-Window Servers, in diesem Falle XMing, ist, dass sich automatisch beim Starten eines Szenarios unter VNUML für jede virtuelle Maschine ein eigenes Kontrollfenster öffnet und so ein komfortablerer Zugriff gegenüber des spärlichen coLinux Kontrollfenster gegeben ist.
Um dem Benutzer aber die Möglichkeit zu geben, manuell ein oder mehrere scrollbare und konfigurierbare Kontrollfenster zu benutzen, wurde PUTTY in das VNUML-Archiv integriert, welches sich beliebig oft starten lässt und so den gleichzeitigen Zugriff auf mehrere virtuelle Maschinen ermöglicht. Der Zugriff auf die coLinux Umgebung mittels Putty erfolgt dabei über die Netzwerkadresse 10.0.2.15, welche coLinux bei der Installation fest zugeordnet wird. Eine genaue Beschreibung findet sich im nächsten Kapitel, wo auch das Starten des im Archiv enthaltenen Beispielszenarios beschrieben wird.

[11] http://www.dit.upm.es/vnumlwiki/index.php/CoLinux-vnuml

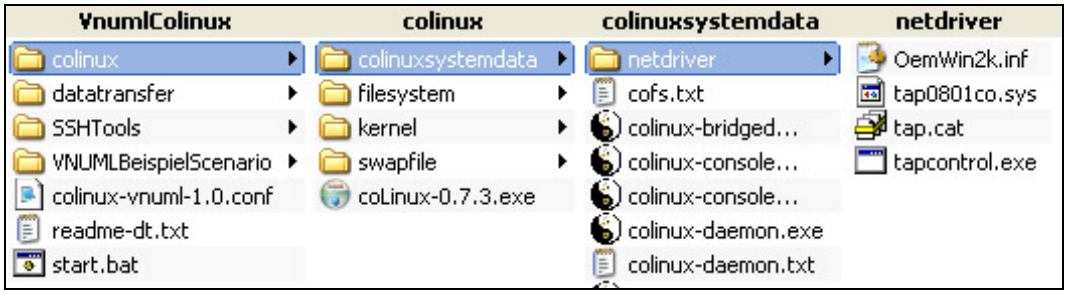
5.5. Überblick über das Archiv „VNUMLColinux.zip“

In den vorherigen Abschnitten wurde bereits beschrieben, wie die benötigten Dateien gemäß den Ansprüchen angepasst wurden. An dieser Stelle wollen wir uns nun den gesamten Inhalt des Archivs VnumlColinux.zip ansehen, um so auch einen besseren Überblick über die vorgenommenen Änderungen der Konfigurationsdateien zu bekommen.

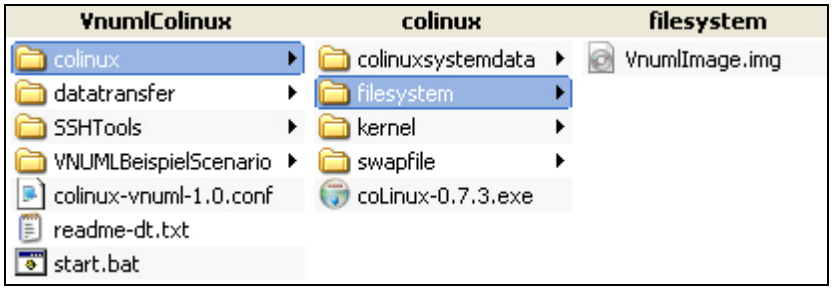
Im Wurzelverzeichnis finden wir die Dateien `start.bat`, `colinux-vnuml-1.0.conf` sowie eine Readme-Datei mit aktuellen Installationshinweisen. Zusätzlich finden wir hier die Ordner `/colinux`, `/datatranser`, `/SSHTools` und `/VNUMLBeispielScenario`.



Wie aus dem vorherigen Abschnitt bekannt, startet die Batchdatei `start.bat` den coLinux-Dämon im Verzeichnis `/colinux/colinuxsystemdata`, in welchem sich alle benötigten Systemdateien der coLinux-Umgebung befinden.

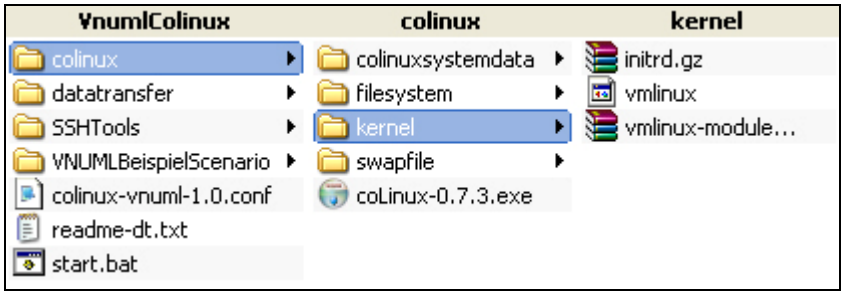


Im Verzeichnis `/colinux/filesystem` finden wir unsere verkleinerte Imagedatei mit der integrierten VNUML-Umgebung.

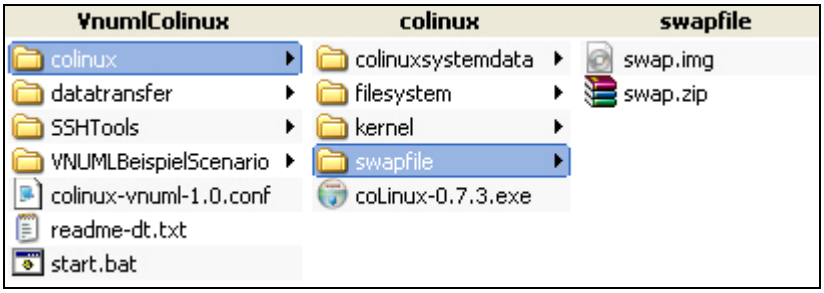


Der Kernel des coLinux-Systems wurde im Verzeichnis `/colinux/kernel` abgelegt, was wir so auch in der Konfigrationsdatei festgelegt haben. Zusätzlich findet man an dieser Stelle

noch die Datei `initrd.gz`, welche aber nur beim erstmaligen Start zu Konfigurationszwecken verwendet wird.



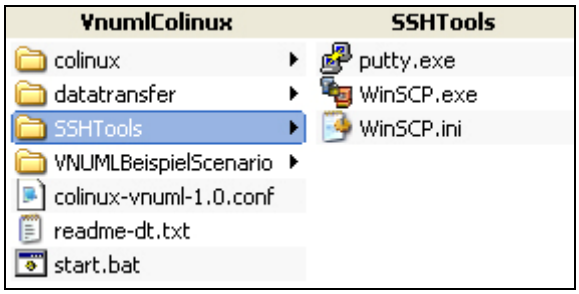
Im Verzeichnis `/colinux/swapfile` befindet sich zunächst die noch gepackte Image-Datei, welche als SWAP-Partition bereitgestellt werden kann. Im komprimierten Zustand benötigt diese Datei nur ~29MByte Speicherplatz, unkomprimiert dagegen stellt sie eine ~1GByte große Sparse-Datei dar, welche von der coLinux-Umgebung als Arbeitsvolumen benutzt werden kann. Eine derart große Komprimierungsrate ist möglich, da die leere Image-Datei nur null-byte Informationen enthält.



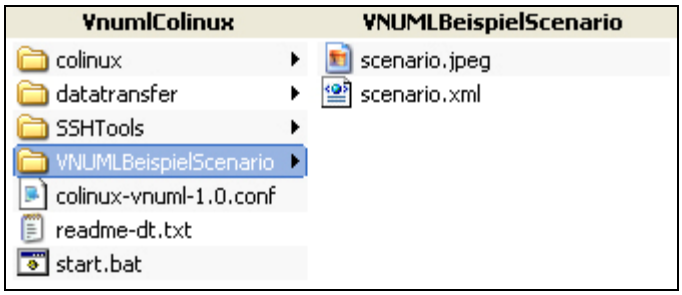
Der Ordner `/datatransfer` ist zu Anfang noch leer. Wie beschrieben, kann dieser zum Datenaustausch zwischen dem Host-System und der virtuellen Maschine genutzt werden. Dateien welche unter Windows in dieses Verzeichnis kopiert werden, stehen danach auch in der virtuellen coLinux-Umgebung zur Verfügung.

Das Verzeichnis `/SSHTools` beinhaltet mit den beiden Programmen *Putty*[12] und *WinSCP*[13] zwei Tools, welche die Nutzung der coLinux Umgebung erleichtern.
Putty ermöglicht es, mehrere Shell-Fenster zu öffnen und so eine parallele Kontrolle über die virtuellen Maschinen in der coLinux Umgebung zu haben. In unserem Fall ist dies die alternative zur Installation eines X-Window Systems, wie es von der DIT-UPM vorgeschlagen wird. Der Vorteil von Putty ist jedoch, dass es ohne Installation direkt ausgeführt werden kann.
Mit WinSCP bietet sich eine weitere Möglichkeit, den Datenaustausch zwischen Host-System und coLinux Umgebung zu realisieren. Es handelt sich dabei um einen grafischen Open-Source-SFTP Client für Windows, welcher über eine SSH Verbindung mit der virtuellen Maschine über dessen Netzwerkschnittstelle zum Host-System kommuniziert.

[12] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
[13] http://www.winscp.org/



Als letztes finden wir im Verzeichnis `/VNUMLBeispielScenario` eine XML-Datei welche die Konfiguration einer VNUML Umgebung beeinhaltet, sowie eine Grafik, die dieses Beispielszenario anschaulich darstellt.



6. Installation und Nutzung von coLinux und der VNUML-Umgebung

Nachdem nun eine Image-Datei für coLinux in einer akzeptablen Größe vorliegt und alle nötigen Konfgurationsdateien an die geforderten Ansprüche angepasst wurden, wird an dieser Stelle nun die Installation der coLinux Umgebung und das Starten der virtuellen Maschine mit integrierter VNUML-Umgebung beschrieben.

Vorraussetzungen:

- Windows XP/Vista (32 Bit) mit mindestens 768MByte Arbeitsspeicher und mind. 1,6GByte freiem Speicherplatz

- coLinux-0.7.3 Installationsdatei[14] (im Archiv der AG Steigner enthalten)

- coLinux –VNUML Archiv[15] der Arbeitsgruppe Steigner

6.1. Installation

- Führen Sie die coLinux Installationsdatei aus und installieren Sie das Programm in einem Verzeichnis Ihrer Wahl (z.B. C:\colinux).
  Das Installieren eines root-Filesystems während der Installation ist hierbei nicht erforderlich.

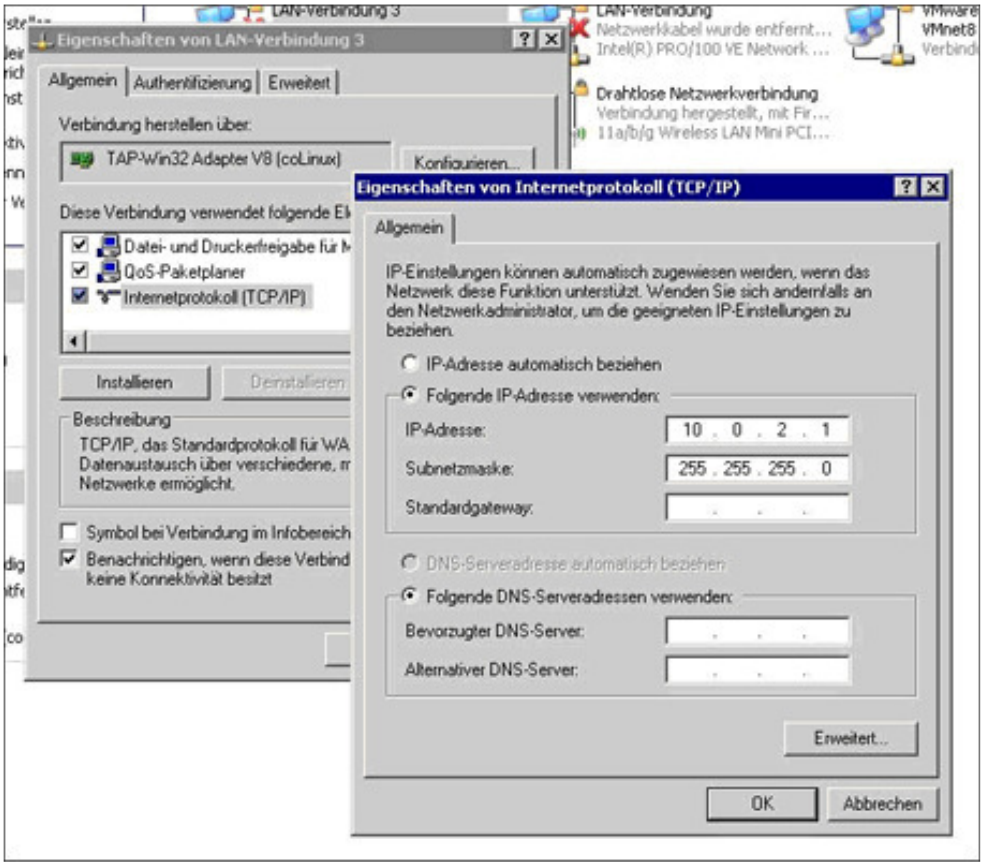
- Einrichten einer Netzwerkverbindung zwischen Windows und coLinux:

  Um später eine Netzwerkverbindung zwischen dem Host-System und der virtuellen Maschine benutzen zu können muss dem TAP-Interface, welches während der coLinux-Installation angelegt wurde, manuell eine IP-Adresse zugeordnet werden.
  Zu finden ist das virtuelle Interface unter dem Gerätenamen „TAP-Win32 Adapter V8 (colinux)“ in den Netzwerkverbindungen.

  Für die Konfiguration öffnen Sie die Eigenschaften des virtuellen Netzwerkadapters und setzen Sie dessen IP Adresse auf 10.0.2.1, unter Verwendung der Netzwerkmaske 255.255.255.0

- Die Benutzung der Swap-Partition ist besonders bei leistungsschwachen Systemen mit weniger als 768MByte Arbeitsspeicher zu empfehlen.
  Um der coLinux-Umgebung eine Swap-Partition zur Verfügung zu stellen, müssen Sie dafür das Archiv `swap.zip` im Verzeichnis `\colinux\swapfile\` entpacken.
  Dadurch wird eine ~1GByte große Datei erstellt, welche automatisch beim nächsten Start der virtuellen Maschine als Swap-Partition eingebunden wird. Ob dies erfolgreich geschehen ist können sie mit dem Befehl `free` unter coLinux testen.

[14] http://sourceforge.net/projects/colinux/files
[15] http://www.uni-koblenz.de/~vnuml/VnumlColinux.zip



- Entpacken Sie das VNUML-Image an einer beliebigen Stelle auf Ihrer Festplatte oder auf einem USB-Wechseldatenträger

- Durch öffnen der Datei start.bat im Stammverzeichniss wird nun die coLinux Umgebung gestartet. Zum Anmelden benutzen Sie dabei:

```
debian login: root
Password: root
```

- Ob das Einrichten der virtuellen Netzwerkverbindung zwischen Host-System und coLinux Umgebung erfolgreich war, lässt sich nun auch mit einem Ping-Befehl in der Kommandozeile von Windows testen.

```
C:\ping -n 2 10.0.2.15

Ping wird ausgeführt für 10.0.2.15 mit 32 Bytes Daten:

Antwort von 10.0.2.15: Bytes=32 Zeit<1ms TTL=64
Antwort von 10.0.2.15: Bytes=32 Zeit<1ms TTL=64

Ping-Statistik für 10.0.2.15:
    Pakete: Gesendet = 2, Empfangen = 2, Verloren = 0 (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 0ms, Maximum = <1ms, Mittelwert = 0.2147483647ms
```

6.2. Starten des Beispielszenarios

Nach erfolgreicher Anmeldung kann die VNUML-Software mit Hilfe des beigefügten Beispiel-Szenarios getestet werden. Hierfür wechseln Sie zuerst in das Unterverzeichnis `/examples` der VNUML-Software in welchem sich mehrere VNUML-Szenario –Dateien finden:

```
debian:~# cd /usr/share/vnuml/examples/
```

Hier kann nun das VNUML-Szenario nun mit dem bekannten Befehl gestartet werden:

```
vnumlparser.pl –t scenario.xml –v
```

Das Einloggen in die virtuellen Maschinen der VNUML-Umgebung funktioniert über eine SSH Verbindung wobei entweder die IP-Adresse oder der Name des UML-Rechners verwendet werden kann:

```
ssh root@uml1

ssh root@192.168.0.102
```

Sollte dabei nach einem Passwort gefragt werden so lautet dieses standardmäßig `xxxx`.

Von den einzelnen UML-Rechnern aus können nun die gewohnten Befehle zur Kontrolle des Netzwerks ausgeführt werden. Als Beispiel sehen wir uns den Ping-Befehl von UML1 (IP: 10.0.1.1) auf UML4 (IP: 10.0.5.2) einmal an:

```
uml1:~# ping –c 2 10.0.5.2
PING 10.0.5.2 (10.0.5.2) 56(84) bytes of data.
64 bytes from 10.0.5.2: icmp_seq=1 ttl=63 time=40.0 ms
64 bytes from 10.0.5.2: icmp_seq=2 ttl=63 time=0.000 ms

--- 10.0.5.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1020ms
rtt min/avg/max/mdev = 0.000/20.000/40.001/20.001 ms
```

Zum Stoppen des VNUML-Szenarios wird der Befehl

```
vnumlparser.pl –d scenario.xml –v
```

benutzt. Sollte es auf diesem Weg nicht gelingen kann es auch „gewaltsam“ mit

```
vnumlparser.pl –P scenario.xml –v
```

beendet werden, wodurch die Simulationsdateien einfach gelöscht werden.

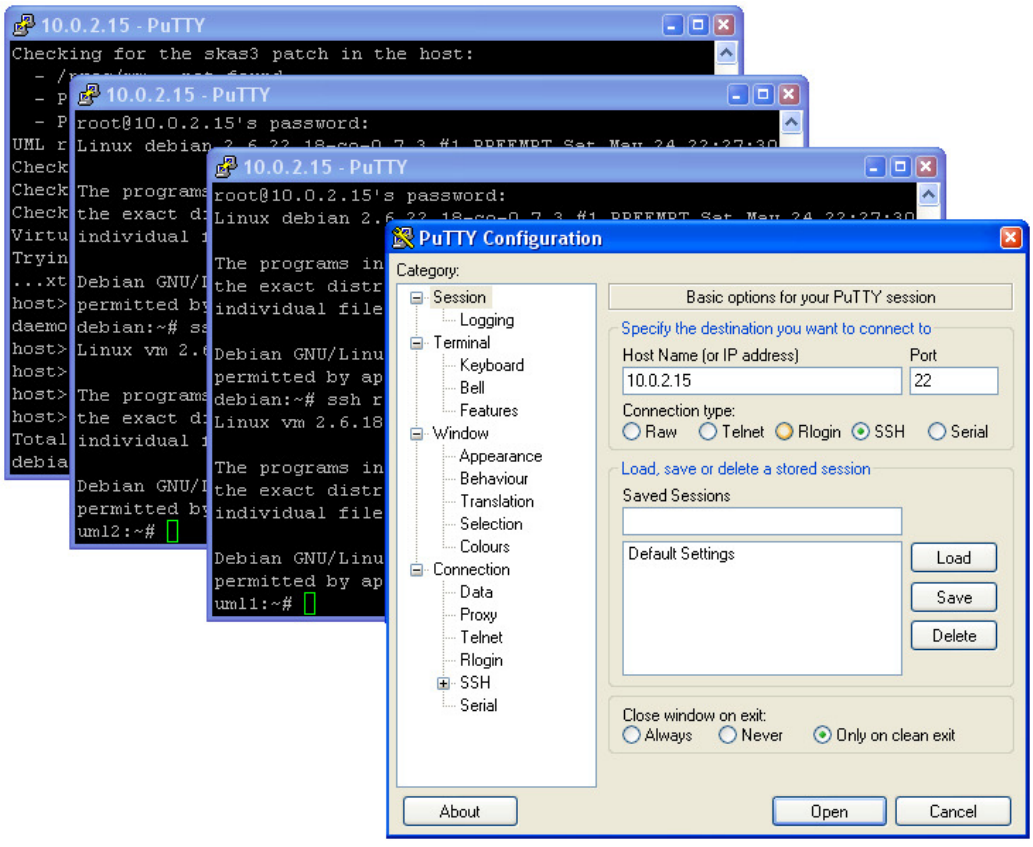
Um die coLinux Umgebung zu beenden kann der Befehl

```
Shutdown –h now
```

benutzt werden, oder indem man einfach die Konsole schließt.

6.3. PuTTY – ein komfortableres Terminal für coLinux

Nachdem die coLinux Umgebung gestartet wurde bemerkt man schnell, das dass integrierte Terminal zur Steuerung der virtuellen Maschine nicht besonders komfortable gestaltet wurde. Unter anderem bietet es keine Möglichkeit die Schriftgröße anzupassen und der Anzeigebereich ist auch nicht über eine Bildlaufschiene veränderbar. Dadurch wird die Anzeige immer auf die letzten 35 Zeilen beschränkt, was das Auslesen von Fehlermeldungen beim Starten eines VNUML Szenarios unmöglich macht.
Diesen Problemen wollten wohl die Entwickler der DIT-UPM bei der ursprünglichen coLinux Variante auch entgegenwirken, indem sie die Installation eines X-windows Server empfehlen. Wie man erkennt setzt dies eine weitere Installation im Host-System voraus, was es jedoch zu vermeiden gab. Daher wurde das SSH-Tool PuTTY in das Archiv aufgenommen, welches ohne Installation in jedem Windows-System ausführbar ist. Wie bereits bekannt, befindet sich das Programm im Unterverzeichnis `/SSHTools`.
Nachdem die coLinux Umgebung gestartet wurde kann man sich mit PuTTY in die virtuelle Maschine einloggen und sie gleichermaßen bedienen, wie dies auch über das coLinux Terminal möglich ist. Um die Verbindung korrekt herzustellen muss lediglich in der Adresszeile die IP-Adresse der coLinux Umgebung (10.0.2.15) angegeben. Weitere Einstellungen wie Schriftgröße etc. können natürlich nach belieben vorgenommen werden. Dadurch ist es auch möglich, die standardmäßig eher kleine Schrift derart anzupassen, so dass z.B. eine Präsentation des VNUML-Szenarios über einen Projektor möglich ist.
Das Programm PuTTY kann außerdem beliebig oft gestartet werden, was das gleichzeitige anmelden bei mehreren virtuellen Maschinen des VNUML-Szenarios ermöglicht. Dadurch können z.b. in einem Fenster mittels `tcpdump` die Aktivitäten an einer Netzwerkschnittstelle beobachtet werden, während man in einem anderen Terminal die Konfiguration verändert.



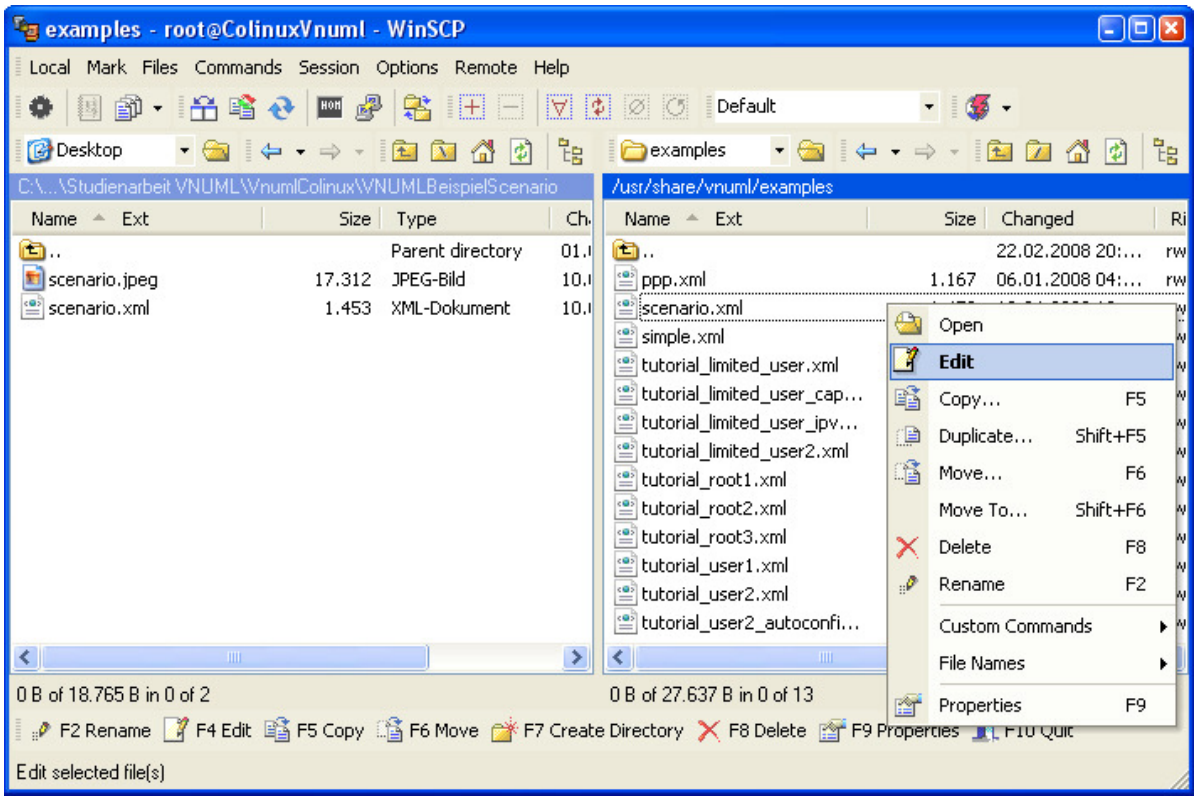
Beispiel mit drei bereits geöffneten Putty-Terminals sowie der Konfiguration zur Herstellung der Verbindung

6.4. Datentransfer zwischen Host-System und coLinux-Umgebung

Um Dateien zwischen dem Host-System und der virtuellen Maschine auszutauschen werden in diesem Abschnitt zwei Möglichkeiten beschrieben. Zum einen die Verwendung des integrierten Programm WinSCP und zum anderen mittels des Ordner `/datatransfer`.
Die Möglichkeit, Daten zwischen Host-System und coLinux auszutauschen ist für die effiziente Verwendung der VNUML-Software unerlässlich. So wird dadurch ermöglicht, bereits vorhandene Szenario-Dateien in die virtuelle Maschine zu kopieren und dort zu verwenden oder eigene Entwicklungen mit anderen VNUML-Benutzern zu teilen. Die für den Übungsbetrieb der AG-Steigner bereitgestellten Szenario-Dateien können auf diesem Weg also problemlos verwendet werden.

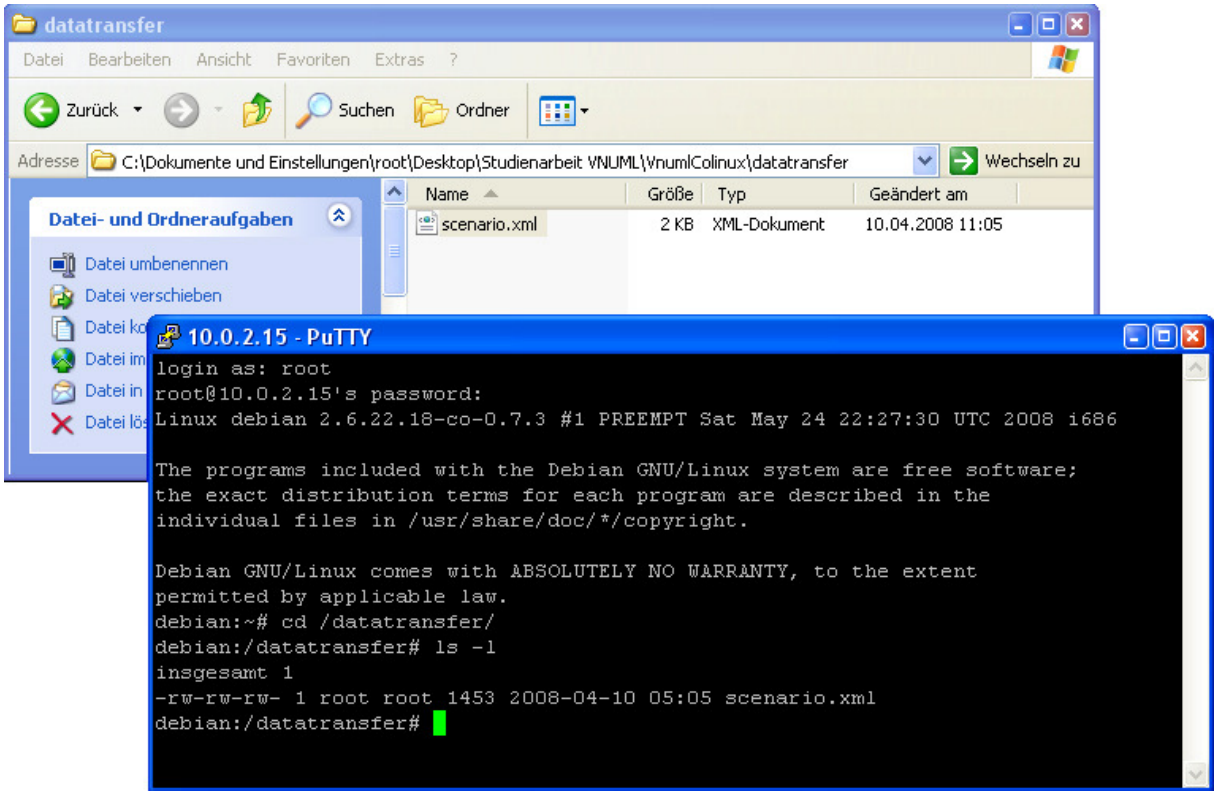
6.4.1. WinSCP

Mit WinSCP wird ein grafischer SFTP-Client zur Verfügung gestellt, welcher einen einfachen und intuitiv zu bedienenden Datentransfer zwischen coLinux und dem Host-System ermöglicht.
Nachdem die Datei WinSCP.EXE aus dem Verzeichnis `/SSHTools` gestartet wurde, muss der Benutzer nur noch den Login-Button betätigen und die Verbindung zur virtuellen Maschine wird automatisch hergestellt, vorausgesetzt natürlich die coLinux Umgebung wurde zuvor gestartet. Die benötigen Anmeldeinformationen und die Adresse sind bereits als Session unter dem Namen „root@CoLinuxVnuml“ gespeichert, so dass der Benutzer hier keine weiteren Einstellungen vornehmen muss.
Wurde die Verbindung hergestellt, kann der Benutzer in den beiden Navigationsbäumen in die gewünschten Verzeichnisse wechseln und per Drag&Drop die gewünschten Dateien austauschen. Zusätzlich können die XML-Dateien mit dem integrierten Editor über die rechte Maustaste geöffnet und editiert werden.



6.4.2. Der Ordner /datatransfer

Eine weitere Möglichkeit, Daten zwischen coLinux und Windows auszutauschen, bietet der Ordner `/datatransfer`. Wie bereits in Abschnitt 5.4. beschrieben, wurde dieser Ordner als so genanntes co-filesystem mit Hilfe der Konfigurationsdatei in die coLinux-Umgebung integriert.
Dadurch stehen alle Dateien die unter Windows in diesen Ordner kopiert werden auch anschließend in der coLinux Umgebung zur Verfügung. Von dort aus können die Dateien dann an einen beliebigen Ort innerhalb der virtuellen Maschine kopiert werden oder auch direkt in diesem Ordner verwendet werden.



6.5. Fehlermeldungen und Problembehebung

Nachdem diese coLinux-Installationsmöglichkeit mit integrierter VNUML-Umgebung ihren praktischen Einsatz fand, traten bei der Benutzung diverse Probleme auf. Hauptursache der Fehler waren dabei meist nicht ordnungsgemäß beendete VNUML-Szenarien, was dazu führt, dass sich die VNUML-Umgebung nicht starten lässt oder das Netzwerk der virtuellen Umgebung nicht wie erwartet reagiert.
Der Grund dafür liegt in nicht gelöschten temporären Dateien der einzelnen UML Maschinen, die unter umständen nicht vollständig gelöscht wurden. Wurde das VNUML-Szenario nicht heruntergefahren und die coLinux Umgebung einfach geschlossen, sind diese sogar noch vollständig vorhanden und verhindern das Anlegen neuer virtueller Maschinen mit der gleichen Bezeichnung.
Um diesem Fehlverhalten entgegenzuwirken wurden die bekannten Probleme zuerst in die Installationsanweisung aufgenommen und die entsprechenden Gegenmaßnahmen beschrieben. In der derzeitigen coLinux-VNUML Version der AG-Steigner wurde diesen Fehlerquellen bereits vorgebeugt, indem automatisch beim Starten der coLinux Umgebung die entsprechenden Verzeichnisse mit den temporären Daten gelöscht werden.
Hierfür wird beim Starten der virtuellen Maschine automatisch das Script *vnumlClean* ausgeführt, welches sich im Verzeichnis */etc/init.d/* befindet. Der dort eingetragene Befehl *rm –rf /root/.vnuml/networks/** sorgt dafür, dass bei jedem Start eventuell vorhandene alte Konfigurationsdateien von VNUML-Szenarien gelöscht werden. So steht dem Benutzer jedesmal ein völlig sauberes System zur Verfügung.
Die nun noch auftretenden Fehler lassen sich daher weitestgehend auf fehlerhafte XML-Dateien oder nicht ausreichende Systemressourcen zurückführen. Letzteren kann man jedoch in den meisten Fällen, wie beschrieben, mit dem Anlegen der Swap-Partition entgegenwirken.

7. Fazit

Das Ergebnis dieser Arbeit hat das eigentliche Ziel, eine Installationsmöglichkeit der VNUML-Software für die Rechnerpools zu finden, sogar noch übertroffen.
Auch wenn die Speicherkapazität der Rechner stärker als anfangs gewünscht belastet wurde, steht mit dieser Lösung eine uneingeschränkt nutzbare VNUML-Umgebung zur Verfügung. Darüber hinaus bietet die Verwendung der coLinux-Umgebung auch denjenigen die Möglichkeit mit VNUML zu arbeiten, die normalerweise kein Linux System benutzen. Aus eigener Erfahrung weis ich, dass viele Studierende auf die Verwendung von VNUML verzichten, weil sie neben dem Standartbetriebssystem kein Linux System installieren möchten. Diese Möglichkeit, VNUML mittels coLinux in einer virtuellen Linux-Umgebung zu verwenden, wird meiner Meinung nach vielen die ersten Berührungsängste mit diesem System nehmen können. Auch wenn die, von der Arbeitsgruppe Steigner zur Verfügung gestellten Offline-Installationsmöglichkeiten von VNUML unter Linux dem Benutzer fast jedes Konfigurations- und Installationsproblem ersparen, ist dies keine Lösung für Benutzer eines Windows-Betriebssystems. Und in Zeiten wo eine 500GByte große Festplatte schon zur Standartausrüstung eines Heimrechners zählt, dürften die erforderlichen 1,5GByte der virtuellen coLinux Maschine kein Speicherplatzproblem mit sich bringen. Die Möglichkeit seine eigene VNUML-Installation mit dieser Variante auf einem USB-Stick mit sich herumzutragen stellt eine zusätzliche praktische Lösung dar. Einzige Vorraussetzung für dessen Verwendung ist dabei ein installiertes coLinux System auf dem jeweiligen Rechner. Ein weiterer Vorteil dieser Lösung ist, dass zukünftige Updates und Änderungen recht einfach und schnell eingespielt werden können. Hierfür muss nur eine einzige neue Imagedatei erstellt werden, welche dann problemlos über das Netzwerk auf die Rechner in den Rechnerpools kopiert werden kann.
Aber nicht nur für die Nutzung einer VNUML-Umgebung stellt dieses System eine praktische Lösung dar. Auf der Homepage der coLinux Entwickler finden sich für Verschiedene Linux Systeme vorgefertigte Imagedateien, welche nach belieben Konfiguriert werden können. So wäre es auch denkbar, coLinux für andere Projekte neben VNUML zu nutzen, wo die Benutzer auf ein Linux-System mit vollen Administratorrechten zurückgreifen müssen.

8. Quellenangaben

- Jeff Dike: User Mode Linux, Prentice Hall International; Auflage: illustrated edition (20. April 2006)

- Trent Hein, Evi Nemeth, Garth Snyder: Linux Administration Handbook, Prentice Hall International; Auflage: 2nd ed. (31. Januar 2005)

- Universität Koblenz-Landau, Arbeitsgruppe Steigner, http://www.uni-koblenz.de/~vnuml,

- Departamento de Ingeniería de Sistemas Telemáticos, (DIT)http://www.dit.upm.es/vnuml,

- Cooperative Linux, http://www.colinux.org

- VMware Inc. http://www.vmware.com

- Debian GNU/Linux http://www.debian.org

9. Anhang

Inhalt der CD:

- Ausarbeitung zum Thema:
  „VNUML-Installation im Windows-Pool
  an der Universität Koblenz-Landau“

- Powerpoint-Präsentation zur Vorstellung
  meiner Arbeit

- Tutorial zur Installation von coLinux und
  Benutzung der VNUML-Umgebung

- ZIP-Archiv incl. coLinux 0.7.3 & VNUML-Installation

CD startet automatisch unter Windows oder öffnen Sie die Datei „index.html“.