Leopold-Franzens-Universität

Innsbruck

Institut für Informatik

Verteilte und Parallele Systeme

Evaluierung von Netzwerk – Testumgebungen, Teil 2

Bakkalaureatsarbeit

eingereicht bei Dr. Ing. Michael Welzl

Florian Matous

Innsbruck, 01. Mai 2005

Danksagung

Ich möchte mich herzlich bei jenen bedanken, die mir bei dieser Bakkalaureatsarbeit mit Rat und Tat zur Seite gestanden sind.

Besonders bedanken möchte ich mich natürlich bei meinem Betreuer Dr. Ing. Michael Welzl.

Weiters möchte ich mich natürlich auch bei den Mitarbeitern der einzelnen Netzwerktestumgebungen und den Mitgliedern der jeweiligen Mailing Lists bedanken, die mir bei diversen Problemen zur Seite standen.

Ein Dank gilt natürlich auch dem Team der Informatik an der Universität Innsbruck, die es durch das Schaffen dieses Studienzweiges ermöglichen, dass ich mir dieses Wissen aneignen konnte.

Florian Matous

Inhaltsverzeichnis

Kurzfassung

Das vorliegende Dokument wurde im Rahmen der Bakkalaureatsarbeit zum Thema „Evaluierung von Netzwerk-Testumgebungen, Teil 2“ erstellt und repräsentiert die Weiterführung der ersten Bakkalaureatsarbeit zum Thema „Evaluierung von Netzwerk-Testumgebungen“. Dabei geht es darum, Netzwerktestumgebungen zu begutachten, wenn möglich praktisch mit diesen zu arbeiten und anschließend zu evaluieren. Im Rahmen dieser Arbeit wurden die folgenden drei Netzwerktestumgebungen untersucht:

- DummyNet
- Internet 2
- PlanetLab

Die Aufgaben im Rahmen dieser Bakkalaureatsarbeit waren:

- Einarbeitung in die jeweilige Testumgebung
- Wenn möglich, Installation und Inbetriebnahme der Testumgebung
- Hinweise für die Installation und Benützung der Testumgebung zu erstellen
- Die Vor- und Nachteile der einzelnen Testumgebungen zu dokumentieren
- Eine Entscheidungshilfe zu geben, wann welche Testumgebung für die jeweiligen Zwecke am Besten geeignet ist
- Ein abschliessendes Resümee über alle Testumgebungen

Summary

This document bears the title “Evaluation of network-testbeds, part 2” and is meant as a bachelor thesis.

The work’s main objective is to test and evaluate network-testbeds, and in doing so three network-testbeds have been analysed, namely

- DummyNet
- Internet 2
- PlanetLab

My tasks regarding this bachelor thesis included:

- understanding how the testbeds work and operate
- if possible, installing and evaluating the testbed
- writing a guideline for the installation and use of the testbed
- determining the advantages and disadvantages of each testbed
- giving help in decision making, when to use which testbed for a specific task
- drawing a conclusion on all evaluated testbeds

Kapitel 1

Einleitung

1.1. Motivation der Arbeit

Da es viele Netzwerktestumgebungen zur freien Verfügung gibt, ist es für einen Anwender nicht einfach, eine Geeignete für dessen Anforderungen und vor allem auch Kenntnisse zu finden. Besonders eine schlechte bzw. dürftige Dokumentation erschwert eine Auswahl zusätzlich und führt wiederum zu einer sehr langen Einarbeitungszeit. Aufgrund dieser Problematik entstand diese Arbeit. Sie soll dem Leser den Einstieg in die jeweilige Netzwerktestumgebung erleichtern und seine Einarbeitungszeit verkürzen. Des Weiteren soll mit Hilfe dieser Arbeit die Installation, die Benützung und die Auswahl der richtigen Umgebung für die jeweilige Situation erleichtert werden.

In dieser Bakkalaureatsarbeit werden mit den Testumgebungen Internet 2 und PlanetLab zwei Plattformen betrachtet, die für sehr viele Forschungsbereiche eingesetzt werden. Hier ist es wichtig, einen grundlegenden Überblick zu geben, damit der Leser einen Eindruck von den Einsatzmöglichkeiten der jeweiligen Testumgebung gewinnt.

1.2. Problemdefinition

Die Problemstellung bei dieser Bakkalaureatsarbeit bestand darin, die oben genannten Netzwerktestumgebungen zu untersuchen, diese wenn möglich praktisch auszuprobieren und anschliessend zu evaluieren. Die Probleme und Besonderheiten, die im Rahmen der Evaluierung aufgetreten sind, werden am Ende dieser Arbeit noch genauer erläutert.

1.3. Zweck der Arbeit

Da es im Internet teilweise wenig Informationen über die Netzwerktestumgebungen, geschweige denn einen Vergleich beziehungsweise eine Empfehlung für die Verwendung einer bestimmten Testumgebung gibt, bestand die Aufgabe, gleich wie bei der ersten Arbeit zu diesem Projekt, darin eine Dokumentation zu verfassen, die es dem Leser erleichtern soll, eine passende Testumgebung zu wählen. Mit Hilfe dieses Dokuments kann sich der Interessierte zum einen mit der jeweiligen Testumgebung vertraut machen und eben auch die passende für die eigene Situation auswählen. Zum anderen wird mit dieser Arbeit die Installation der benötigten Tools erleichtert und hoffentlich verständlich erläutert.

Kapitel 2

DummyNet

2.1. Einführung

2.1.1. Was ist DummyNet?

DummyNet ist ein flexibles Tool das ursprünglich zum Testen von Netzwerkprotokollen entwickelt wurde, aber heute vor allem im Bereich Bandwidth-Management verwendet wird. DummyNet ist ausschliesslich auf FreeBSD-Systemen verfügbar und ist Teil des FreeBSD-Kernels; es kann auf normalen Hosts, aber auch auf einem Router verwendet werden. DummyNet wurde von Luigi Rizzo an der Universität von Pisa entwickelt, wobei Luigi Rizzo unter der E-Mail Adresse luigi@iet.unipi.it erreichbar ist. Die offizielle DummyNet-Homepage ist http://info.iet.unipi.it/~luigi/ip_dummynet/. Weitere gute Informationsquellen zum Thema DummyNet sind die Manpages `ipfw(8)`, `dummynet(4)`, `bridge(4)`. Die offizielle FreeBSD-Homepage ist http://www.freebsd.org/.

2.1.2. Arbeitsweise von DummyNet

IPFW - Firewall

DummyNet ist ein Feature der IPFW-Firewall von FreeBSD und somit ein fixer Bestandteil eines jeden FreeBSD-Basissytems. IPFW arbeitet regelbasiert, d.h. jedes

Paket durchläuft eine Liste in der die für Firewalls typischen Regeln definiert sind. Sobald eine Regel auf ein Paket zutrifft, wird die darin definierte Aktion durchgeführt. Tatsächlich ist DummyNet nur ein "Zusatzfeature" von IPFW und DummyNet ist die Bezeichnung dieses Features. DummyNet ist kein eigenständig ausführbares Programm.

Einige einfache Beispiele zur Konfiguration der IPFW-Firewall:

IPFW – Befehl	**Wirkung**
`ipfw add allow tcp from any to any 80`	TCP-Traffic auf Port 80 wird erlaubt.
`ipfw add allow icmp from any to any`	ICMP-Traffic (für z.b. Ping) wird nicht blockiert.
`ipfw add deny any from any to any`	Jeglicher Traffic wird blockiert.
`ipfw add allow ip from any to any`	Jeglicher IP-Traffic wird erlaubt.
`ipfw add allow icmp from any to any`	Jeglicher ICMP-Traffic wird erlaubt.

Achtung: Standardmäßig wird von IPFW jeglicher Verkehr blockiert! Dies muss auf Wunsch explizit mit z.B. der Regel `ipfw add allow ip from any to any` geändert werden.

Die Konfiguration der IPFW-Firewall ist ein ziemlich komplexes Thema und wird für die Verwendung von DummyNet nicht benötigt. Es muss nur dafür gesorgt werden, dass nicht standardmäßig der gesamte Traffic blockiert wird (siehe `ipfw -` Kommando weiter oben). Für detaillierte Informationen zur Konfiguration von Firewall-Regeln sei auf http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html verwiesen.

DummyNet - Funktionsweise

DummyNet fängt Pakete auf dem Weg durch den Protokollstack ab und schiebt diese in sogenannte Pipes. Eine Pipe simuliert einen Netzwerklink mit fest zugewiesener Bandbreite, Propagation Delay und Packet Loss. Pipes werden mit dem `ipfw` – Kommando erstellt und konfiguriert. Die Syntax des `ipfw` – Kommandos wird später genau erläutert.

Mit DummyNet können die folgenden Parameter bzw. Eigenschaften von Netzwerklinks manipuliert bzw. simuliert werden:

- Bandbreite auf fixe Werte beschränken
- Queue – Size für eine Pipe festlegen
- Propagation Delay für eine Pipe festlegen
- Random Packet Loss für eine Pipe festlegen

2.2. Installation

2.2.1. Installation von FreeBSD und DummyNet

Die folgenden Installationsangaben beziehen sich auf die verwendete FreeBSD-Version 5.2.1. Jedoch laufen die grundlegenden Installationsschritte in späteren Versionen gleich ab. Alle durchgeführten Tests wurden auf einem Samsung-Notebook durchgeführt. Aufgrund der Tatsache, dass ein Notebook verwendet wurde, ist es nicht gelungen, eine lauffähige graphische Oberfläche (X-Window-System) zu installieren, was jedoch für den Einsatz von DummyNet auch nicht nötig ist.

Die Installation gliedert sich in die folgenden Abschnitte:

1. Download der Iso-Images und Installation von FreeBSD
2. IPFW - Firewall aktivieren und konfigurieren
3. DummyNet in Kernelkonfigurationsdatei aktivieren
4. Kernel neu kompilieren und installieren

1. Download der Iso-Images und Installation von FreeBSD

Die benötigten FreeBSD ISO-Images können auf ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/ heruntergeladen werden. Für den Betrieb von DummyNet wird eigentlich nur das FreeBSD-Basissystem benötigt, da DummyNet fix im Kernel von FreeBSD integriert ist. Daher genügt es, nur

die CD 1 herunterzuladen. Das offizielle FreeBSD-Handbuch und eine Installationsanleitung sind auf http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/index.html zu finden. Die Installation von FreeBSD ist ziemlich einfach, da ein graphischer Installationsassistent ("sysinstall") zur Verfügung gestellt wird und einfach den Installationsanweisungen am Bildschirm gefolgt werden muss. Die Konfiguration des Netzwerk-Interfaces funktionierte bei problemlos. Lediglich der Versuch der Installation einer X-Window-Umgebung scheiterte.

Wenn FreeBSD korrekt installiert wurde, kann DummyNet noch nicht eingesetzt werden, da die IPFW – Firewall und somit auch DummyNet standardmäßig deaktiviert sind!

2. IPFW – Firewall aktivieren und konfigurieren

Als nächstes muss die IPFW – Firewall aktiviert und konfiguriert werden .Dies geschieht, indem die Datei `/etc/rc.conf` editiert wird. In der `rc.conf` werden die meisten Netzwerkeinstellungen unter FreeBSD vorgenommen. Für das Editieren von Dateien wurde der `ee` – Editor verwendet. Es müssen die untenstehenden Zeilen hinzugefügt werden:

```
firewall_enable="YES"
firewall_script="/etc/ipfw.rules"
```

Das erste Kommando aktiviert die Firewall. Das zweite Kommando gibt den Pfad zu einer Shell-Skript-Datei an, in der die Firewall-Regeln und DummyNet-Operationen definiert werden können. Der Name und das Verzeichnis der Skript-Datei können selber gewählt werden. Die Verwendung einer Skript-Datei vereinfacht das Arbeiten mit DummyNet, da nicht mehr jedes Kommando einzeln über die Konsole eingeben werden muss. Das Shell-Skript wird automatisch beim Starten von FreeBSD ausgeführt und kann mit dem Befehl `sh ipfw.rules` ausgeführt werden. Um nicht immer explizit `sh` angeben zu müssen, kann in die erste Zeile des Skripts folgendes geschrieben werden: `#!/bin/sh`. Dadurch kann das Skript direkt mit Angabe des Namens ausführt werden. Es wurden die oben angeführten Einstellungen verwendet.

3. DummyNet in Kernelkonfigurationsdatei aktivieren

Als nächstes muss DummyNet aktiviert werden, da es standardmäßig deaktiviert ist. Dazu wird die Kernelkonfigurationsdatei `/usr/src/sys/i386/conf/GENERIC` editiert. In dieser Datei werden sämtliche Kernelkonfigurationen vorgenommen. Es ist empfehlenswert, vor der Modifizierung eine Kopie der Datei anzufertigen und einen Link auf die Kopie zu erstellen.

Dies wird folgendermassen durchgeführt:

```
# cp GENERIC /root/kernels/DER_NEUE_KERNEL
# ln -s /root/kernels/DER_NEUE_KERNEL
```

Jetzt kann die Datei *DER_NEUE_KERNEL* im Verzeichnis `/root/kernels` modifiziert werden.

Es werden folgende Zeilen hinzugefügt:

```
options IPFIREWALL
options DUMMYNET
options HZ=1000
```

Die erste und zweite Anweisung aktivieren die Firewall und DummyNet. Mit der dritten Anweisung wird die Timergranularität eingestellt. Dies ist wichtig, damit DummyNet Delays korrekt simulieren kann.

4. Kernel neu kompilieren und installieren

Der letzte Schritt besteht darin, den FreeBSD-Kernel neu zu kompilieren und zu installieren. Der Kernel kann relativ einfach neu kompiliert und installiert werden.

Folgende Befehle werden hierfür verwendet:

```
# /usr/sbin/config DER_NEUE_KERNEL
# cd ../compile/DER_NEUE_KERNEL
# make depend
# make
# make install
```

Mit der zweiten Anweisung wird in das Bauverzeichnis gewechselt und anschließend wird mit `make` der Kernel kompiliert und installiert.

Falls alle Installationsschritte korrekt durchgeführt wurden, ist DummyNet nach einem Reboot einsatzbereit.

DummyNet auf einem Router

Für den Einsatz von DummyNet auf einem Router gilt die oben angeführte Installationsanleitung. Zusätzlich muss noch eine Einstellung in der `/etc/rc.conf` gemacht werden.

Es muss folgende Zeile hinzugefügt werden:

```
gateway_enable=YES
```

FreeBSD verwendet den Standard-BSD-Routing-Daemon `routed`, der das RIP verwendet. BGP und OSPF werden von `net/zebra` unterstützt. Falls statische Routen verwendet werden möchten, kann dies in der Datei `/etc/rc.conf` konfiguriert werden.

Folgendes Beispiel wurde dem FreeBSD-Handbuch entnommen:

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

Die Variable `static_routes` enthält eine Reihe von Strings, die durch Leerzeichen getrennt sind. Jeder String bezieht sich auf den Namen einer Route. In diesem Beispiel hat `static_routes` “internalnet2“ als einzigen String. Zusätzlich wird die Konfigurationsvariable `route_internalnet2` verwendet, in der alle sonstigen an `route` zu übergebenden Parameter festgelegt werden.

Im obigen Beispiel hätte alternativ folgender Befehl verwendet werden können:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

2.2.2. Verwendung von PicoBSD

Falls FreeBSD nicht auf dem Rechner installiert werden kann oder DummyNet nur kurz ausprobiert werden soll, kann PicoBSD verwendet werden. PicoBSD ist ein lauffähiges FreeBSD-Sytem mit voller DummyNet-Funktionalität, gepackt auf eine bootfähige Floppy – Diskette. Das Image File von PicoBSD befindet sich auf http://info.iet.unipi.it/~luigi/ip_dummynet/#PicoBSD. Es braucht nur im BIOS das Diskettenlaufwerk als First-Boot-Device einstellt werden und dann kann von der Diskette PicoBSD gestartet werden. Während des Startvorgangs muss ein Hostname und eine IP-Adresse angegeben werden. Anschließend ist es möglich, sich als root mit dem Passwort "setup" einzuloggen um mit DummyNet zu spielen.
Jedoch handelt es sich um eine sehr abgespeckte Version von FreeBSD die im Wesentlichen nur das `ipfw`-Kommando und die gebräuchlichsten Netzwerkbefehle (`route, tcpdump, netstat, ifconfig` usw.) zur Verfügung stellt.

2.3. Arbeiten mit DummyNet

2.3.1. Anlegen und Konfigurieren von Pipes

Wie vorher bereits erwähnt funktioniert DummyNet so, dass der Netzwerkverkehr in sogenannte Pipes geschoben wird, welche einen Netzwerklink mit bestimmten Eigenschaften(Bandbreite, Delay usw.) simulieren. Pipes werden mit dem `ipfw`-Kommando angelegt und konfiguriert.

Sämtliche Pipes in DummyNet sind Half-Duplex-Pipes; soll ein normaler Full-Duplex-Link simuliert werden, müssen zwei Pipes definiert werden, die dann jeweils Download- und Uploadparameter des Links definieren (mit den Schlüsselwörtern `in` und `out` möglich). Falls explizit Half-Duplex-Links simuliert werden sollen, sind die Schlüsselwörter `in, out` nicht notwendig. Eckige Klammern in den Syntaxdefinitionen weisen auf optionale Parameter hin.

Anlegen von Pipes

Die Syntax des `ipfw`-Kommandos zum Anlegen einer Pipe lautet wie folgt:

```
# ipfw add pipe N proto from src to dest [in/out]
```

Parameter	**Bedeutung**
N	Beliebige Nummer zur Identifikation der Pipe (von 1 bis 65534).
Proto	Genau ein Protokolltyp für die Pipe (IP, ICMP, TCP, UDP).
src, dest	Quell- und Ziel-IP-Adressen(-bereiche) für die Pipe.
in, out	Gibt an ob die Pipe die Download- oder Uploadkapazität eines Netzwerklinks definiert.

Als Quell- und Ziel-IP-Adressen können entweder einzelne IP-Adressen oder IP-Adressen-Bereiche in der Form X.X.X.X/Y angegeben werden, wobei Y gleich der Anzahl der Netzbits ist. Einige Beispiele für Adressenbereiche: 10.1.2.0/24, 131.130.0.0/16, 192.35.244.0/24 usw. Weiters kann für src und dest auch `any` benutzt werden. Dies bedeutet, dass als Quell- und Ziel-IP-Adressen alle Adressen in Frage kommen.

Konfigurieren von Pipes

Die Syntax des `ipfw`-Kommandos zum Konfigurieren einer Pipe lautet wie folgt (die Reihenfolge der Konfigurationsparameter spielt keine Rolle):

```
# ipfw pipe N config param
```

Parameter	**Bedeutung**
N	Nummer der Pipe (von 1 bis 65534).

Param	Folgende Paramter sind möglich: • `bw` (Bandwidth) • `queue` (Queue-Size) • `delay` (Propagation-Delay) • `plr` (Random Packet Loss)

Nützliche ipfw-Befehle für die Arbeit mit DummyNet

Die folgende Tabelle zeigt die nützlichsten Befehle, die im Umgang mit DummyNet benötigt werden:

Befehl	**Bedeutung**
`# ipfw –a list`	Anzeigen der konfigurierten `ipfw`-Rules
`# ipfw pipe list`	Auflistung der vorhandenen Pipes
`# ipfw delete <rulenumber>`	Löschen einer `ipfw`-Rule
`# ipfw delete <pipenumber>`	Löschen einer Pipe
`# ipfw pipe flush`	Löschen aller Pipes (Pakete, die an eine nicht existente Pipe gehen, werden einfach gedroppt!)

2.3.2. Konfigurationsparameter von Pipes

Bandwidth

Mit dem Konfigurationsparameter Bandwidth kann die verfügbare Bandbreite für die Pipe (also einen simulierten Netzwerklink) festgelegt werden.

Die Bandbreite wird folgendermassen festgelegt:

```
# ipfw pipe XY config bw NNunit
```

Nach dem `config`-Parameter folgt der `bw`-Parameter, wobei `NN` für die Bandbreitenkapazität steht und `unit` eine der folgenden Einheiten sein kann: `bit/s`, `Kbit/s`, `Mbit/s`, `Byte/s`, `KByte/s`, `MByte/s`. Eine Angabe von 0 bedeutet keine Bandbreiteneinschränkung.

Einige Beispiele für die Konfiguration der Bandbreite einer Pipe mit der Nr. “XY“:

```
# ipfw pipe XY config bw 128Kbit/s
# ipfw pipe XY config bw 56Kbit/s
# ipfw pipe XY config bw 2MByte/s
```

Ein Beispiel für das Erstellen und Konfigurieren von zwei Pipes die einen Full-Duplex-ADSL-Link simulieren (Pipe 3 = Upload, Pipe 4 = Download):

```
# ipfw add pipe 3 ip from any to any out
# ipfw add pipe 4 ip from any to any in
# ipfw pipe 3 config bw 128Kbit/s
# ipfw pipe 4 config bw 640Kbit/s
```

Delay

Mit dem Konfigurationsparameter Delay kann das Propagation Delay für die Pipe in Millisekunden festgelegt werden. Das Propagation Delay ist unabhängig vom Queue Delay.

Das Propagation Delay wird folgendermassen festgelegt:

```
# ipfw pipe XY config delay NNms
```

Nach dem `config`-Parameter folgt der `delay`-Parameter, wobei `NN` für das Propagation Delay in Millisekunden steht.

Einige Beispiele für die Konfiguration der Bandbreite und des Delays einer Pipe mit der Nr. “XY“:

```
# ipfw pipe XY config bw 128Kbit/s delay 100ms
# ipfw pipe XY config bw 56Kbit/s delay 100ms
# ipfw pipe XY config bw 2MByte/ delay 1000ms
```

Ein Beispiel für das Erstellen und Konfigurieren von zwei Pipes die einen Full-Duplex-ADSL-Link simulieren (Pipe 3 = Upload, Pipe 4 = Download, Delay = 500ms):

```
# ipfw add pipe 3 ip from any to any out
# ipfw add pipe 4 ip from any to any in
```

```
# ipfw pipe 3 config bw 128Kbit/s delay 500ms
# ipfw pipe 4 config bw 640Kbit/s delay 500ms
```

Queue

Mit dem Konfigurationsparameter Queue kann die Queue-Size für die Pipe festgelegt werden.

Die Queue-Size wird folgendermassen festgelegt:

```
# ipfw pipe XY config queue NN[unit]
```

Nach dem `config`-Parameter folgt der `queue`-Parameter, wobei `NN` für die Grösse der Queue steht. Folgende Einheiten sind für `unit` zulässig: `Bytes, Kbytes`. Wird für `unit` nichts angegeben, so wird die Grösse in Slots interpretiert. Wird die Queue-Size nicht explizit definiert, so wird fünfzig Slots als Default-Size verwendet. Bei niedrigen Bandbreiten und grossen Queues können hohe Queue-Delays auftreten!

Einige Beispiele für die Konfiguration der Bandbreite, der Queue-Size und des Delays einer Pipe mit der Nr. “XY“:

```
# ipfw pipe XY config bw 128Kbit/s queue 10Kbytes delay 100ms
# ipfw pipe XY config bw 56Kbit/s queue 30 delay 100ms
# ipfw pipe XY config bw 2MByte/ queue 20Kbytes delay 1000ms
```

Ein Beispiel für das Erstellen und Konfigurieren von zwei Pipes die einen Full-Duplex-ADSL-Link simulieren (Pipe 3 = Upload, Pipe 4 = Download, Delay = 500ms, Queue-Size f. Pipe 3 = 10 Slots, Queue-Size f. Pipe 4 = 30 Slots):

```
# ipfw add pipe 3 ip from any to any out
# ipfw add pipe 4 ip from any to any in
# ipfw pipe 3 config bw 128Kbit/s queue 10 delay 500ms
# ipfw pipe 4 config bw 640Kbit/s queue 30 delay 500ms
```

Random Packet Loss

Mit dem Konfigurationsparameter Random Packet Loss kann eine Wahrscheinlichkeit angegeben werden, mit der Pakete auf dieser Pipe verloren gehen. Dieser Parameter ist nützlich um fehlerhafte Links zu simulieren.

Die Random Packet Loss Rate wird folgendermassen festgelegt:

```
# ipfw pipe XY config plr X
```

Nach dem `config`-Parameter folgt der `plr`-Parameter, wobei `X` eine Gleitkommazahl zwischen 0 und 1 ist und die Wahrscheinlichkeit eines Verlusts angibt. Wird 0 angegeben, so bedeutet dies eine Wahrscheinlichkeit von 0, also keinen Verlust von Paketen.

Ein Beispiel für das Erstellen und Konfigurieren von zwei Pipes die einen Full-Duplex-ADSL-Link simulieren (Pipe 3 = Upload, Pipe 4 = Download, Delay = 500ms, Queue-Size f. Pipe 3 = 10 Slots, Queue-Size f. Pipe 4 = 30 Slots, Loss-Rate f. Pipe 3 = 0.2, Loss-Rate f. Pipe 4 = 0.3):

```
# ipfw add pipe 3 ip from any to any out
# ipfw add pipe 4 ip from any to any in
# ipfw pipe 3 config bw 128Kbit/s queue 10 delay 500ms plr 0.2
# ipfw pipe 4 config bw 640Kbit/s queue 30 delay 500ms plr 0.3
```

2.3.3. Beispiel-Skripte für DummyNet

Wie bereits erwähnt kann in der Datei `/etc/rc.conf` ein Pfad zu einer Shell-Skript-Datei (Firewall-Script) angegeben werden, in der Firewall- und DummyNet-Regeln definiert werden können. Da dieses Skript ein Standard-Shellskript ist, stehen einem alle Möglichkeiten der Shell-Programmierung zur Verfügung.

Die folgenden zwei Skripte zeigen einfache Beispiele für DummyNet.

ICMP – Delay, Ping

Das unten aufgeführte Skript verzögert sämtlichen ICMP-Traffic um 100ms. Um die Funktionsweise dieses Skripts zu demonstrieren wurde folgendes Testszenario verwendet:

- Rechner A (IP: 192.168.0.2) macht Ping auf Rechner B (IP: 192.168.0.4)
- Rechner B verzögert sämtlichen ICMP-Traffic um 100ms mit Hilfe von DummyNet
- Rechner A verwendet als Betriebssystem Windows XP und Rechner B verwendet FreeBSD mit DummyNet.

Für das obige Szenario kann folgendes Skript verwendet werden:

```
#!/bin/sh       #Shebang

# DUMMYNET – SCRIPTFILE – ICMP - DELAY

ipfw pipe flush                         #delete all existing pipes
ipfw flush                              #delete all existing rules

ipfw add pipe 1 icmp from any to any out #create pipe for icmp-traffic
ipfw pipe 1 config delay 100ms            #config delay for icmp-traffic
ipfw add allow icmp from any to any      #allow icmp traffic
```

Wird dieses Skript auf Rechner B ausgeführt, so ergibt sich folgendes Ergebnis (Abbildung 1) nach Durchführung des Pings auf Rechner A:

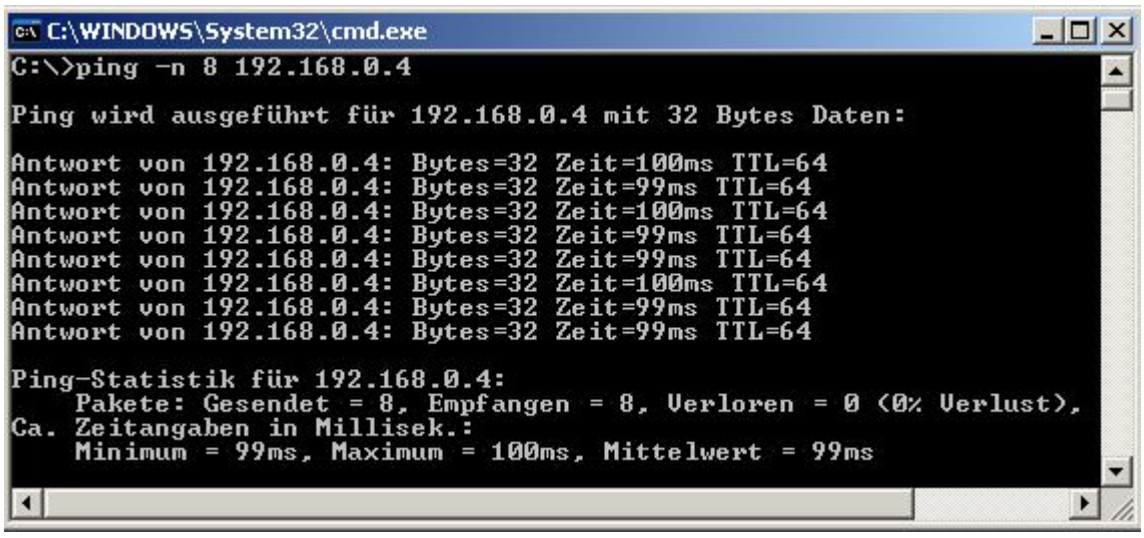

Abbildung 1: Ergebnis mit DummyNet

Zu beachten ist, dass bei der Erstellung der Pipe der Parameter `out` verwendet wird. Dadurch wird erreicht, dass nur der ausgehende ICMP-Traffic (also der ICMP_ECHO_REPLY) um 100ms verzögert wird. Sollte das `out` weggelassen werden, so wird im Falle eines einfachen Pings die ICMP_ECHO_REQUEST- und die ICMP_ECHO_REPLY-Nachricht um 100ms verzögert. Daraus würde sich ein sichtbares Delay von 200ms ergeben!

Zum Vergleich zeigt die untenfolgende Abbildung das Ergebnis eines Pings ohne DummyNet:

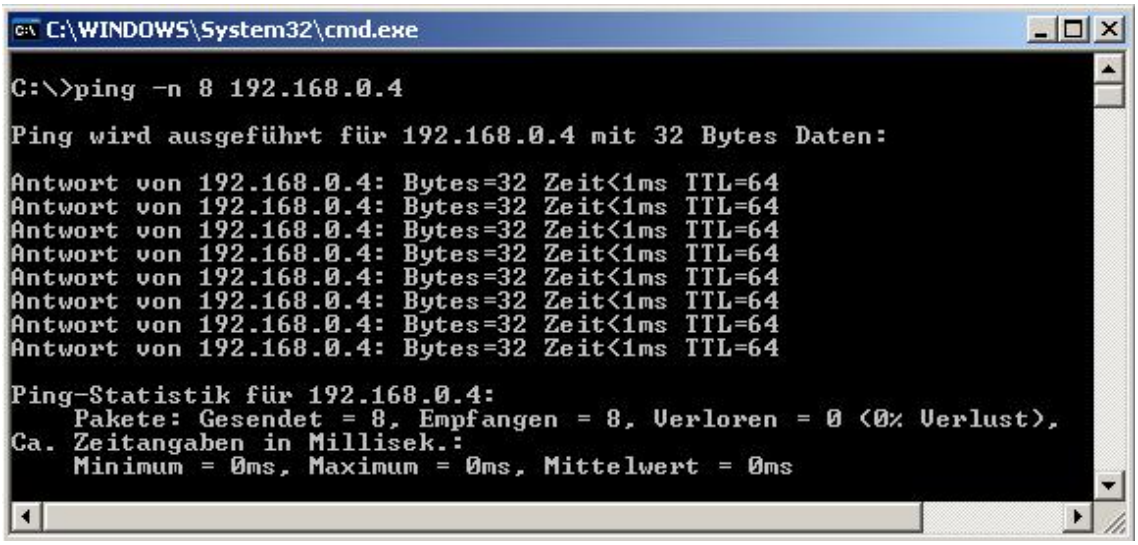

Abbildung 2: Ergebnis ohne DummyNet

Bandbreitenbeschränkung

DummyNet wird vor allem für Bandwidth-Management verwendet. Soll z.B. der Web-Traffic auf einem Rechner beschränkt werden oder allgemein sämtlicher IP-Traffic zu einem bestimmten IP-Adressenbereich beschränkt werden, so kann eine Konfiguration wie in dem unten aufgeführtem Skript verwendet werden.

Dieses Skript simuliert folgendes Szenario:

- Die Bandbreite für Web-Traffic vom Rechner mit der IP 192.168.0.2 wird auf 128 Kbit/s beschränkt.
- Die Bandbreite von sämtlichem ausgehenden IP-Traffic vom Rechner mit der IP 192.168.0.2 zu den Rechnern 131.130.0.0/16 wird auf 256 Kbit/s beschränkt. Weiters wird die Queue-Size der Pipe auf 10 Slots gesetzt und sämtlicher Traffic um 200ms verzögert.

```
#!/bin/sh       #Shebang

# DUMMYNET – SCRIPTFILE
# BANDWIDTH - LIMITATION

ipfw pipe flush                         #delete all existing pipes
ipfw flush                              #delete all existing rules


#create pipe for web-traffic from ip 192.168.0.2
ipfw add pipe 1 tcp from 192.168.0.2 port 80 to any

#config pipe, set bandwidth limitation
ipfw pipe 1 config bw 128Kbit/s

#create pipe for outgoing ip-traffic to 131.130.0.0/16
ipfw add pipe 2 ip from 192.168.0.2 to 131.130.0.0/16 out

#config pipe, set bandwitdth, delay and queue-size in slots
ipfw pipe 2 config bw 256Kbit/s delay 200ms queue 10

ipfw add allow ip from any to any     #allow all traffic
```

Kapitel 3

Internet 2

3.1. Einführung

3.1.1. Was ist Internet 2?

Internet 2 ist ein von amerikanischen Universitäten gegründetes Consortium, welches sich zum Ziel gesetzt hat, Netzwerkapplikationen und –Netzwerktechnologien zu entwickeln, die die Entwicklung des Internets massgeblich beeinflussen bzw. beschleunigen sollen. Um dies zu erreichen gibt es zahlreiche Kooperationen zwischen Akademischen Institutionen, Regierung und großen Unternehmen. Die offizielle Webseite von Internet 2 ist http://www.internet2.edu/.

Die Hauptziele des Internet 2 - Consortiums sind:

- Eine “state-of-the-art“ Netzwerktestplattform für die nationale Forschungsgemeinde von Internet 2 zur Verfügung zu stellen
- Das Entwickeln von Next-Generation Internet-Applikationen zu ermöglichen
- Die Nutzung von neuen Netzwerk-Services bzw. Internet-Services einer breiten Masse von Internet-Usern zu ermöglichen

3.1.2. Ziele von Internet 2

Die Mission bzw. das Ziel von Internet 2 besteht in der Entwicklung von Netzwerkapplikationen, welche die Entwicklung des "Nachfolgers des Internet" (eben das Internet 2) beschleunigen sollen.

Folgende Betrachtung bzw. Analyse der Entwicklung des Internet stammt aus einer Präsentation über Internet 2 und verdeutlicht den Standpunkt bzw. die Ziele von Internet 2 ziemlich gut:

Yesterday's Internet

- Thousands of users
- Remote login, file transfer
- Interconnect mainframe computers
- Applications capitalize on underlying technology

Today's Internet

- Millions of users
- Web, email, low-quality audio & video
- Interconnect personal computers and servers
- Applications adapt to underlying technology

Tomorrow's Internet (= Ziele von Internet 2)

- Billions of users and devices
- Convergence of today`s applications with multimedia (telephony, video-conference, HDTV)
- Interconnect personal computers, servers and embedded computers
- New technologies enable unanticipated applications (and create new challenges)

3.1.3. Wer ist Mitglied bei Internet 2?

Die Mitglieder von Internet 2 kommen im Wesentlichen aus den 3 Bereichen Universitäten (university), Privatwirtschaft (corporate) und Regierung (government). Zum Zeitpunkt der Verfassung dieser Arbeit sind 207 Universitäten im Internet 2 – Consortium vertreten. Die Universitäten übernehmen die leitende Funktion bei Internet 2. Für eine vollständige Liste aller Universitäten sei auf http://members.internet2.edu/university/universities.cfm verwiesen.

Im Bereich der Privatwirtschaft sind einige bekannte Unternehmen aus dem IT-Sektor vertreten. Beispiele hierfür sind: IBM, Cisco Systems, Microsoft Research und Sun Microsystems. Für eine vollständige Liste der Unternehmen sei auf http://members.internet2.edu/corporate/corporate.cfm#sponsors verwiesen. Die Universitäten werden von den Unternehmen gesponsert, jedoch gibt es natürlich eine enge Zusammenarbeit zwischen den Unternehmen und den Universitäten.

Es existiert auch eine enge Zusammenarbeit mit der amerikanischen Regierung. Die Regierung hat ein eigenes Projekt mit dem Namen NGI[1] gestartet, welches eine ähnliche Zielsetzung wie Internet 2 hat, wobei enge Kooperationen zwischen Intenet 2 und NGI vorhanden sind.

Weiters gibt es zahlreiche internationale Kooperationen. Die neueste Zusammenarbeit besteht mit China. Es findet jedes Jahr das CANS (Chinese-American Networking Symposium) statt, bei dem die Forschungsteams jeweils ihre Ergebnisse präsentieren.

3.1.4. Wie werde ich Mitglied bei Internet 2?

Zum Zeitpunkt der Erstellung dieser Arbeit konnten nur Universitäten bzw. Institutionen innerhalb der U.S.A Mitglied bei Internet 2 werden. So weit ermittelt werden konnte, ist es z.B. für die Universität Innsbruck nicht möglich, Mitglied bei Internet 2 zu werden! Jedoch existieren internationale Zusammenarbeiten mit gleichgesinnten Organisationen und Forschungsnetzen. Eine Auflistung aller

[1] Next Generation Internet, Webseite: http://www.ngi.gov

internationalen Partner von Internet 2 kann auf http://international.internet2.edu/partners/ gefunden werden. Es gibt beispielsweise Kooperationen mit dem DFN-Verein (Deutsches Forschungsnetz, http://www.dfn.de/content/de/) oder mit DANTE (Delivery of Advanced Network Technology to Europe http://www.dante.org.uk/). Jedoch wurden keine Belege für eine Zusammenarbeit zwischen einer vergleichbaren österreichischen Organisation und Internet 2 gefunden.

3.1.5. Kontakt zur Internet 2 - Community

Mailing Lists

Alle öffentlichen Mailing Lists sind über das "Mailing List Online Interface" unter der URL https://mail.internet2.edu/wws/ zugänglich. Dieses "Interface" ist ähnlich wie ein Forum aufgebaut. Die einzelnen Mailing Lists sind nach Bereichen unterteilt und Nachrichten können wie in einem Forum gepostet werden. Um Zugang zu bekommen ist es notwendig, sich unter https://mail.internet2.edu/wws/subrequest/i2-news zu registrieren.

Die Mailing Lists sind in die folgenden Bereiche unterteilt:

- Application
- Corporate
- E2E (End 2 End Initiative)
- Engineering
- Middleware
- MITC
- Network
- News and Media
- Security
- Technical
- Administrative
- Others

Um generelle Informationen und Neuigkeiten über Internet 2 zu erhalten, gibt es ausserdem noch die “I2-INFO“ Mailing List. Für diese können sich Interessierte unter https://mail.internet2.edu/wws/subrequest/i2-info registrieren.

Für eine genaue Auflistung sämtlicher Mailing Listen für jeden Bereich wird auf das Forum unter https://mail.internet2.edu/wws/ verwiesen.

Veranstaltungen

Die Internet 2- Community organisiert einige Meetings bzw. Workshops in denen die Mitglieder sämtlicher Bereiche (Universitäten, Wirtschaft und Regierung) persönlich Erfahrungen austauschen.

Folgende Veranstaltungen sind zum Zeitpunkt der Verfassung der Arbeit fixiert (und finden jährlich statt):

Internet 2 Events:

- Spring Internet 2 Member Meeting
- Fall Internet 2 Member Meeting

Internet 2 Workshops:

- Performance and Master Class Production Workshop
- CAMP Med: Identity and Access Management for Medical Applications Workshop
- ESCC/Internet 2 Joint Techs Workshop

Nähere Informationen zu sämtlichen Events können im “Internet 2 Online Calendar“ auf http://events.internet2.edu/ nachgelesen werden.

3.1.6. Struktur von Internet 2

Internet 2 ist im Wesentlichen in die unten aufgeführten Forschungsbereiche bzw. Tätigkeitsbereiche eingeteilt. Die Mitglieder arbeiten in sogennanten Working Groups an den jeweiligen Projekten in den verschiedenen Forschungsbereichen.

Forschungs- bzw. Tätigkeitsbereiche:

- Backbone Network
- Engineering
- Applications
- Initiatives

In den folgenden Kapiteln wird pro Tätigkeitsbereich eine Working Group detaillierter beschreiben. Die einzelnen Tätigkeitsbereiche bzw. Working Groups sind zu komplex, um sie in dieser Arbeit alle genau zu beschreiben. Es wurde aus den Tätigkeitsbereichen jeweils eine Working Group ausgewählt, die am interessantesten erschien.
Für einen Überblick über alle existierenden Working Groups sei auf http://www.internet2.edu/working-groups.html verwiesen. Weiters gibt es noch die sogenannten Special Interest Groups (SIG) und die Advisory Groups (AG). Eine Übersicht über alle existierenden SIGs und AGs findet sich ebenfalls auf http://www.internet2.edu/working-groups.html. Der Unterschied zu den normalen Working Groups besteht darin, dass SIGs rein informativer Natur sind. Die Mitglieder treffen sich nur zwei mal im Jahr bei den Internet 2 Member Meetings. Es werden keine konkreten Forschungen miteinander betrieben. AGs dienen in erster Linie der Koordination von Zusammenarbeiten zwischen den Working Groups.

3.2. Backbone Network von Internet 2

3.2.1. Abilene Network

Das Abilene Network ist das Hauptforschungsnetzwerk von Internet 2. Abilene verbindet innerhalb der USA 220 Institutionen mit einer Kapazität von derzeit 10 Gbit/s, jedoch gibt es auch Zusammenarbeiten mit internationalen High Performance Netzwerken (Peer Networks von Abilene). Eine Übersicht über diese kann auf http://abilene.internet2.edu/peernetworks/ gefunden werden. Eine Liste aller Institutionen innerhalb der USA, die mit dem Abilene verbunden sind, kann auf http://abilene.internet2.edu/community/participants/list.html eruiert werden.

Das Abilene Network ist eine nicht-kommerzielle Einrichtung; d.h. das gesamte Netzwerk wird ausschliesslich für Forschungszwecke verwendet. Die “Conditions of Use“ von Abilene auf http://abilene.internet2.edu/policies/cou.html enthalten detaillierte Informationen über die Nutzung von Abilene.

Abilene Architektur und Fakten

Das Abilene Network verbindet die Mitgliedsinstitutionen über sogenannte gigaPoPs, welche die Kernstruktur vom Abilene Network bilden. Ein gigaPoP ist der Verbindungspunkt zum Abilene Network und besteht aus High Performance Geräten, wie zum Beispiel dem Juniper Networks T640 Router (der sogenannte Core Router bzw. Backbone). Diese Router bilden die sogenannte Core Architecture von Abilene und verbinden die Institutionen untereinander. Insgesamt gibt es 11 dieser Highspeed Core Router.

Einige Fakten zum Abilene Network:

- Die Kapazität des Abilene Networks liegt derzeit bei 10 Gbit/s. Diese Kapazität wird durch den Einsatz von Glasfaserkabeln (optische Netzwerktechnologien) erreicht.
- Das Abilene Network besteht aus ca. 13.000 Meilen Glasfaserkabel

- Das Abilene Network arbeitet ca. 180.000 – mal schneller als ein 56k Modem
- Der monatliche Traffic liegt bei ca. 1.600 Terabyte
- Jeder Router unterstützt IPv6 und Multicast
- Die Administration vom Abilene Network wird von der Indiana University in Indianapolis durchgeführt (Network Operations Center – NOC).
- Für detaillierte Statistiken zu Traffic- und Netzwerktests siehe http://www.abilene.iu.edu/ndoc.html.

Abilene Verfügbarkeit über ACOnet

Eines der internationalen Peer Networks von Abilene ist das GEANT Network, (Website: http://www.geant.net/). Das GEANT Network ist ein Zusammenschluss von 26 europäischen Forschungsnetzen. Das österreichische Forschungsnetz ACOnet (Website: http://www.aco.net/) ist eines dieser 26 Forschungsnetze und somit indirekt mit dem Abilene Network verbunden.

Um genauere Informationen zu erhalten, wurde eine Mail an die Abilene-Administration verfasst. In dieser Mail wurde ein erfolgreicher Traceroute von einem Abilene-Server zur Website der Universität Innsbruck (http://info.uibk.ac.at) mitgeteilt:

```
traceroute to info.uibk.ac.at (138.232.1.216), 30 hops max, 38 byte packets

1  207.75.164.1 (207.75.164.1)  0.489 ms  0.413 ms  0.317 ms
2  e0.aadl.mich.net (198.108.90.129)  0.480 ms  0.479 ms  0.474 ms
3  abilene-iplsng.mich.net (192.122.183.10)  69.598 ms  11.004 ms  11.035 ms
4  chinng-iplsng.abilene.ucaid.edu (198.32.8.76)  14.852 ms  19.118 ms 14.844 ms
5  abilene.nl1.nl.geant.net (62.40.103.165)  116.394 ms  116.393 ms  116.414 ms
6  nl.de1.de.geant.net (62.40.96.101)  127.765 ms  127.765 ms  127.752 ms
7  de1-1.de2.de.geant.net (62.40.96.130)  127.775 ms  127.936 ms  127.770 ms
8  de.at1.at.geant.net (62.40.96.57)  136.760 ms  136.783 ms  167.184 ms
9  aconet-gw.at1.at.geant.net (62.40.103.2)  136.974 ms  136.979 ms  137.016 ms
10 Wien2.ACO.net (193.171.23.34)  249.906 ms  255.655 ms  326.930 ms
11 Ibk-GBS.ACO.net (193.171.12.210)  146.398 ms  146.440 ms  146.397 ms
12 rdmz.uibk.ac.at (193.171.19.2)  146.820 ms  146.811 ms  146.796 ms
13 sr1.uibk.ac.at (138.232.10.125)  146.466 ms  146.486 ms  146.327 ms
14 info.uibk.ac.at (138.232.1.216)  146.589 ms  146.683 ms  146.758 ms
```

Es wurden jedoch keine Informationen über Zusammenarbeiten bzw. Projekte zwischen dem Abilene Network und ACOnet gefunden.

Die folgende Abbildung zeigt die Architektur vom Abilene Network mit den 11 Core Router, sowie den Traffic und die Auslastung (gekennzeichnet durch die Farben) am 4.Jänner 2005:

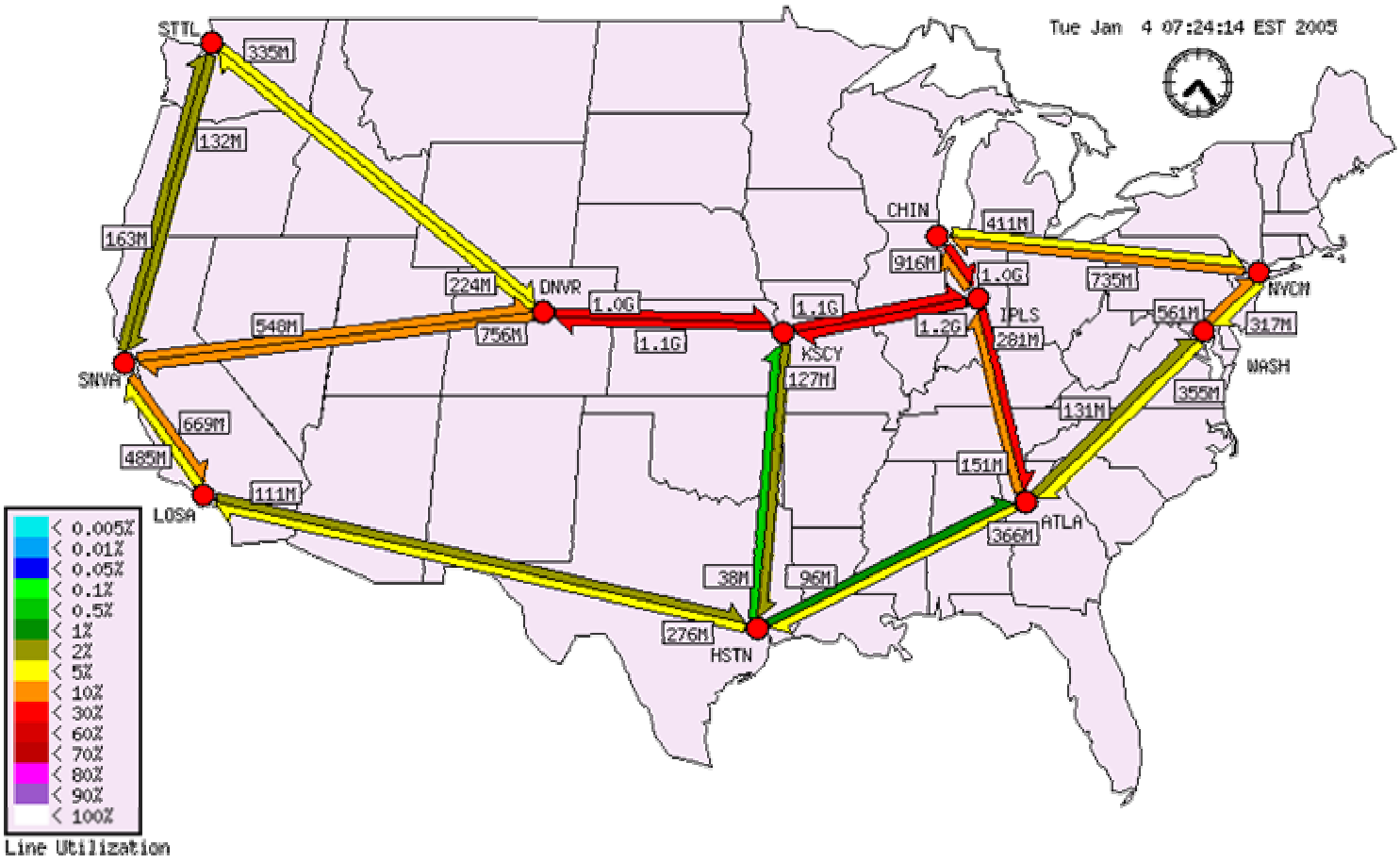

Abbildung 3: Architektur vom Abilene Network und Trafficauslastung[2]

[2] Quelle: http://abilene.internet2.edu/maps-lists/

Die folgende Abbildung zeigt die internationale Anbindung von Abilene bzw. der Core Router an andere Highspeed – Netzwerke:

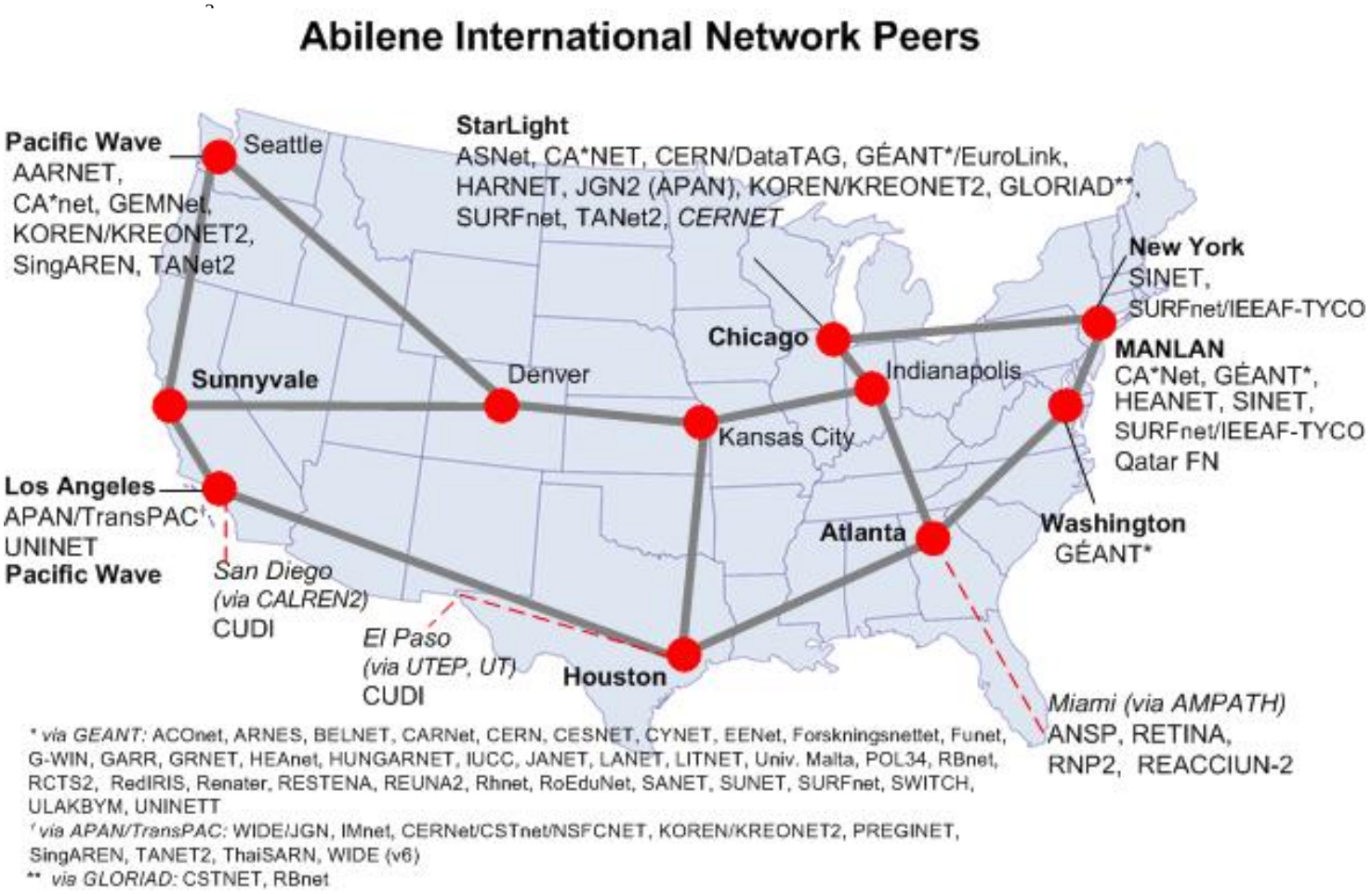

Abbildung 4: Internationale Anbindung vom Abilene Network[3]

[3] Quelle: http://abilene.internet2.edu/maps-lists/

3.3. Engineering

3.3.1. IPv6 Working Group

Diese Working Group beschäftigt sich, wie der Name schon sagt, mit der Entwicklung und Verbreitung von IPv6 innerhalb der High-Performance Netzwerke von Internet 2, vor allem dem Abilene Netzwerk. Alle Hosts, die mit IPv6 erreichbar sind, können in einer Liste, aufgeschlüsselt nach Netzwerken unter der URL http://ipv6.internet2.edu/ipv6hosts.shtml identifiziert werden. Nähere Informationen zur Working Group befinden sich unter http://ipv6.internet2.edu/.

Die Aktivitäten der IPv6 WG werden in die folgenden 3 Areas unterteilt:

Engineering/Operational Area

Dieser Bereich kümmert sich um die tatsächliche Realisierung von IPv6 und Installation von IPv6 fähigen Hosts im Internet 2 Netzwerk. Wichtige Punkte sind beispielsweise die IPv6 – Adressierung und die Aufteilung der Adressen auf die Universitäten innerhalb des Abilene Netzwerks. Hierzu gibt es einen „“IPv6 Adressing Plan“, zu finden unter http://ipv6.internet2.edu/Abilene_IPv6_Addressing.shtml, und eine Aufteilung der Adressen auf die Universitäten unter http://ipv6.internet2.edu/Abilene_Allocations.shtml.

Educational Area

Dieser Bereich kümmert sich um die administrative Arbeit die im Zusammenhang mit der Verwendung von IPv6 anfällt. Dazu gehört beispielsweise die Einschulung der Administratoren der jeweiligen Universitäten. Weiters sind diese Personen die Ansprechspartner bei Problemen mit IPv6 innerhalb des eigenen Netzwerkes.

Motivational Area

Dieser Bereich ist sozusagen die “Marketing Abteilung von IPv6”, welcher den Mitgliedsinstitutionen von Internet 2 die Vorteile und den Einsatz von IPv6 näherbringen soll.

Kontakt zur Working Group

Nähere Informationen zum Beitritt lassen sich unter http://ipv6.internet2.edu/wg-ipv6-join.shtml nachlesen. Um der Mailing List beizutreten, wird eine Mail an die Adresse sympa@internet2.edu gesendet. Der Betreff der Mail muss folgendem Format genügen: “sub wg-ipv6 FirstName LastName“. Weitere Informationen zur Working Group gibt es in den Mailing List Archiven unter https://mail.internet2.edu/wws/arc/wg-ipv6.

3.4. Applications

3.4.1. Voice Over IP Working Group

Diese Working Group beschäftigt sich, wie der Name schon sagt, mit der Entwicklung von Applikationen zur Kommunikation über High-Performance Netzwerke (vor allem dem Abilene-Netzwerk).

Es gibt enge Zusammenarbeiten mit den anderen Real-Time-Communication Working Groups (z.B. Digital Video WG).

Folgende Projekte laufen in dieser Working Group:

- Internet 2 SIP.edu, nähere Infos unter: http://voip.internet2.edu/SIP.edu/
- Voice Disaster Recovery, nähere Infos unter: http://voip.internet2.edu/dr/
- H.323 VoIP Testbed, nähere Infos unter http://voip.internet2.edu/h323/

Kontakt zur Working Group

Es existiert eine eigene VoIP - Mailing List: wg-voip@internet2.edu, welcher unter http://voip.internet2.edu/participate.html beigetreten werden kann. Weitere Informationen zu dieser Working Group können in den Mailing List Archiven unter https://mail.internet2.edu/wws/arc/wg-voip nachgelesen werden.

3.5. Initiatives

3.5.1. End-to-End Performance Initiative (E2Epi)

E2E piPEs Framework

Die Entwicklung des E2E piPEs Framework stellt das wichtigste Projekt der E2Epi dar. Die Bezeichnung E2E piPEs ist eine Abkürzung für: End to End Performance Initiative Performance Environment System. Der Hauptzweck vom piPEs ist das Auffinden von Performanceproblemen und -kapazitäten in Netzwerken (z.B. Abilene) zwischen zwei Endpunkten. Dies wird erreicht, indem der gesamte E2E Weg in kleine Segmente unterteilt wird. In diesen Segmenten werden dann Netzwerktests wie Bandbreitenmessung, Latenzzeitmessung und Routingtests durchgeführt. Die Resultate von jedem Segment werden in einer zentralen Datenbank gespeichert. Nähere Informationen hierzu unter http://e2epi.internet2.edu/e2epipes/index.html.

Die Hauptziele von E2E piPEs sind:

- Dem User das Auffinden von E2E – Problemen zu ermöglichen und die betreffenden Personen zur Problemlösung zu kontaktieren
- Fernsteuerung von Performance Tests über Teile des gesamten E2E Wegs
- Performancedaten über einzelne Teile eines E2E Wegs öffentlich zugänglich machen

Die folgende Abbildung zeigt die Funktionsweise vom E2E piPEs Framework:

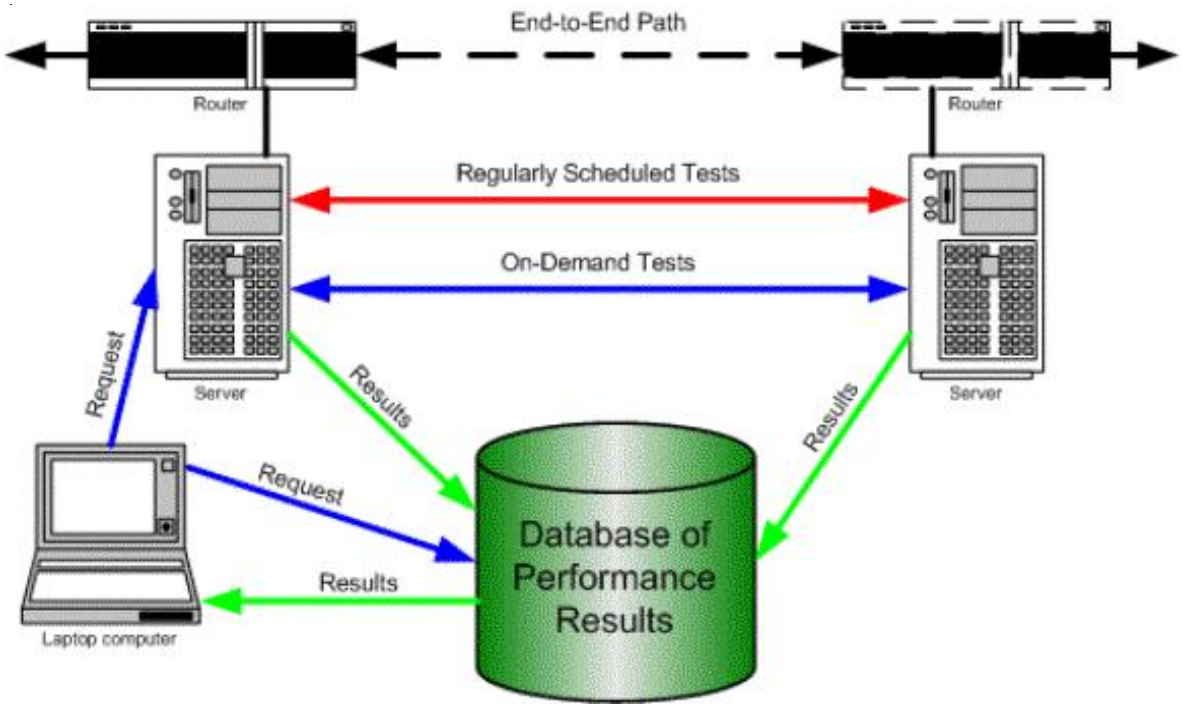

Abbildung 5: Funktionsweise von E2E piPEs[4]

[4] Quelle: http://e2epi.internet2.edu/e2epipes/

Kapitel 4

PlanetLab

4.1. Einführung

4.1.1. Was ist PlanetLab?

PlanetLab ist eine, quer über den ganzen Planeten vernetze Testplattform, welche als Entwicklungsplattform für geographisch verteilte Netz-Applikationen (die sogenannten PlanetLab-Services) sowie als Testumgebung für Netzwerkszenarien dient. Jedoch ist die PlanetLab-Umgebung weit mehr als eine Netzwerktestumgebung. Es wird zum Beispiel auch in den Forschungsbereichen mit den Schwerpunkten Router Design oder Distributed Hash Tables eingesetzt. Nähere Details hierzu im Kapitel 4.1.7, “Anwendungen, Services in PlanetLab“.

Zurzeit besteht PlanetLab aus 458 Maschinen (den PlanetLab-Nodes) in 25 verschiedenen Ländern (den PlanetLab-Sites), wovon sich die meisten in Universitäten bzw. Forschungseinrichtungen in den USA befinden. Es gibt es jedoch auch Kooperationen mit anderen Testbeds wie Emulab oder Internet 2. Es ist zum Beispiel möglich in Emulab einen PlanetLabSlice zu erstellen. Weiters sind einige der Maschinen von PlanetLab Teil des Abilene-Backbone-Networks von Internet 2.

Alle nötigen Informationen zu PlanetLab, die auch zu einem grossen Teil die Grundlage dieses Kapitels bildeten finden sich auf der Website http://www.planet-lab.org/. Leider war die Möglichkeit PlanetLab praktisch zu testen nicht gegeben, jedoch konnte aufgrund der doch recht umfangreichen Dokumentation ziemlich genau auf das Thema PlanetLab eingegangen werden.

PlanetLab ist eine sehr umfangreiche Testplattform, die den Usern zahlreiche Services zu den unterschiedlichsten Forschungsbereichen bereitstellt. Aufgrund des Themas dieser Arbeit, ist vor allem der Network-Measurement-Service Scriptroute relevant und wird im Kapitel 4.4, “Arbeiten mit PlanetLab“- Netzwerktests mit Scriptroute näher vorgestellt.

Folgende Abbildung illustriert das PlanetLab-Network:

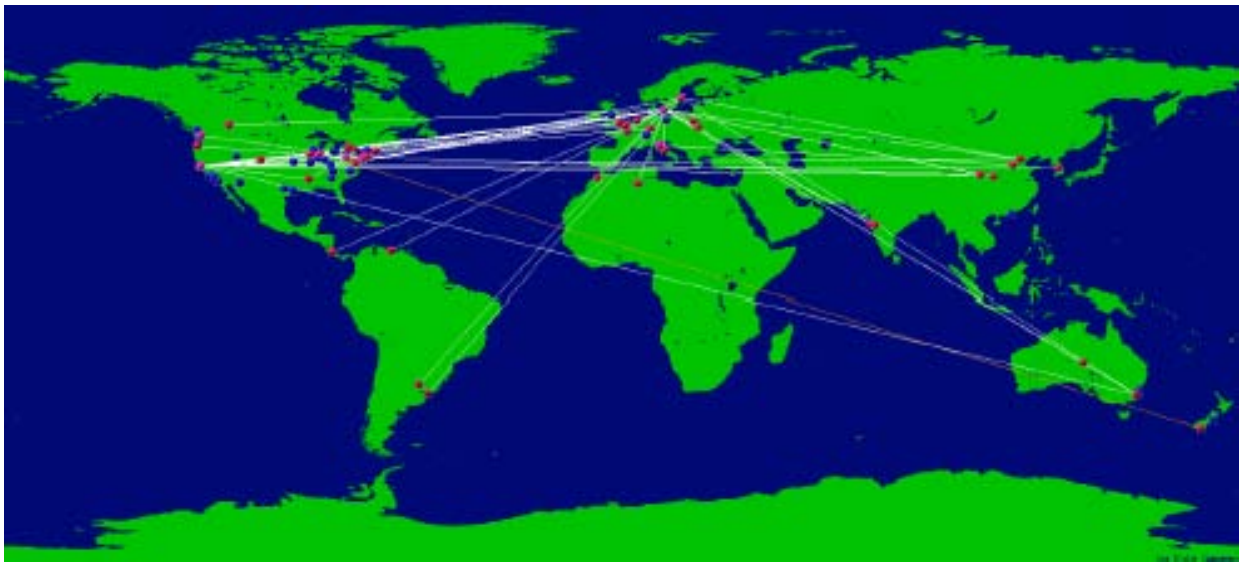

Abbildung 6: Weltweite Verteilung der PlanetLab-Sites[5]

[5] Quelle: http://www.planet-lab.org/

4.1.2. Wer ist Mitglied bei PlanetLab?

Die Mitglieder von PlanetLab umfassen viele renommierte Universitäten und Forschungszentren, aber auch grosse Unternehmen aus dem IT-Sektor wie Intel oder HP. Die meisten dieser Institutionen befinden sich in den USA, jedoch sind auch einige europäische Institutionen vertreten.
Alle Institutionen, die zum Zeitpunkt der Verfassung dieser Arbeit Mitglieder von PlanetLab sind, können auf http://www.planet-lab.org/php/institutions.php nachgelesen werden.

Welche Arten der Mitgliedschaft gibt es?

PlanetLab unterscheidet zwischen akademischen und kommerziellen Organisationen. Je nach Art der Institution ist die Mitgliedschaft gebührenpflichtig oder gebührenfrei. Die folgende Tabelle gibt einen Überblick über die Arten der Mitgliedschaft:

Mitgliedslevel	**Gebühren/Jahr**
Industrial	
Charter	$300,000 U.S.
Full	$75,000 U.S.
Associate	$25,000 U.S.
Sponsor	$10.000 U.S.
Academic	
Managing Party	0$
Academic	0$

Erwähnenswert ist, dass akademische Mitglieder keine Gebühren zahlen müssen, jedoch über die gleichen Leistungen wie Mitglieder des Levels Charter und Full verfügen.

Für eine ausführliche Beschreibung sei auf das Dokument https://www.planet-lab.org/consortium/Governance_1203.pdf mit detaillierten Beschreibungen der Mitgliedsarten verwiesen.

4.1.3. Wie werde ich Mitglied bei PlanetLab?

PlanetLab stellt im Vergleich zu Emulab oder NistNet hohe Ansprüche technischer und organisatorischer Natur an die Mitglieder. Es müssen zum Beispiel mind. zwei Rechner zur Verfügung gestellt werden, die ausschliesslich für PlanetLab verwendet werden. Der gesamte Prozess für Institutionen die beitreten wollen, gliedert sich in die folgenden Abschnitte:

1. Rechner für PlanetLab festlegen
2. Ansuchen um Mitgliedschaft (Apply for Membership)
3. Connection Request auf der Webseite beantragen

Rechner für PlanetLab festlegen

Der erste Schritt besteht darin festzulegen, welche Rechner für PlanetLab zur Verfügung gestellt werden. Es müssen mindestens zwei Geräte zur Verfügung gestellt werden. Sehr wichtig ist, dass jene Rechner nur für PlanetLab verwendet werden können, da mit der Installation der PlanetLab-Software alle Daten auf der Festplatte gelöscht werden! PlanetLab schreibt für die Rechner minimale Hardwareanforderungen vor, welche ziemlich hoch sind.

Folgende Auflistung zeigt die Hardwareanforderungen für neue Mitglieder:

- 1.5 GHz IA32, PIII class processor
- 1 GByte RAM
- 160 GByte total disk space (no single disk smaller than 2 GByte)
- CD-ROM drive that supports bootable CD
- Floppy drive
- Fast Ethernet interface

Sehr wichtig bei der Auswahl der Rechner ist, dass die jeweiligen Modelle der Rechner mit dem PlanetLab-Betriebssystem kompatibel sind. PlanetLab schlägt eine Reihe von Hardware-Konfigurationen vor, die bereits verifizierte PlanetLab-Nodes sind und von denen bekannt ist, dass sie funktionieren. Eine gute Wahl sind Rechner der Marke Dell, da in Zusammenhang mit diesen keine Konfigurationsprobleme bekannt sind. Jeder der

festgelegten Rechner benötigt eine statische IP-Adresse, die entweder manuell oder mit DHCP zugewiesen werden kann. Weiters müssen die Rechner ausserhalb der hauseigenen Firewall sein.

Für eine detaillierte Beschreibung der Hardwareanforderungen und der funktionsfähigen Konfigurationen sei auf http://www.planet-lab.org/consortium/hardware.php verwiesen.

Ansuchen um Mitgliedschaft

Als Nächstes muss eine Institution beim PlanetLab Consortium um Mitgliedschaft ansuchen. Dies geschieht, indem das Membership Agreement http://www.planet-lab.org/consortium/pl_member_20040218.pdf ausgefüllt wird. Zuerst muss entschieden werden, welches Mitgliedslevel gewählt wird. Konkret würde das für die Universität Innsbruck den Mitgliedslevel Academic oder Managing Party bedeuten.
Nun müssen der Administrator (Primary Administrative Contact) und eine Ansprechperson für technische Probleme (Primary Technical Contact) festgelegt werden. Weiters muss eine vom Institut authorisierte Person unterschreiben, damit der Mitgliedsvertrag gültig wird. Es ist möglich, dass zum Beispiel Administrator und Unterzeichnender dieselbe Person sind.
Als nächstes sollten die Terms and Conditions of Membership sorgfältig durchlesen werden. Die Mitgliedschaft kann innerhalb von 30 Tagen von beiden Seiten gekündigt werden, läuft dann aber für ein Jahr (initial year) und verlängert sich auf Wunsch auf drei Jahre.
Das vollständig ausgefüllte Dokument muss nun an die Princeton University in den USA übermittelt werden. Dazu besteht die Möglichkeit, die Originale einzuscannen und per E-Mail an die folgende Adresse zu senden: mthompso@princeton.edu. Alternativ kann auch der normale Postweg benutzt werden. PlanetLab weist ausdrücklich darauf hin, dass alle Anträge ohne entsprechende Unterschriften nicht bearbeitet werden.

Die Rechner wurden festgelegt und das Ansuchen versendet, was nun?

Wenn die oben beschriebenen Punkte erledigt wurden, muss gewartet werden bis das Antwortschreiben von PlanetLab eintrifft und feststeht, ob die Mitgliedschaft genehmigt wurde. In den Terms and Conditions of Membership wird angegeben, dass die Bearbeitung eines Ansuchens innerhalb von 30 Tagen erfolgt.

Im Rahmen des gesamten Beitrittsprozesses, sind sehr viele Dokumente zu lesen, die umfangreiche Informationen zu PlanetLab bereitstellen. Alle relevanten Dokumente befinden sich auf http://www.planet-lab.org/consortium/.

Jene Dokumente, die auf alle Fälle gelesen werden sollten sind die folgenden:

Governance Document
Beschreibt die Organisation und den Aufbau des PlanetLab Consortiums näher. Dieses Dokument befindet sich auf http://www.planet-lab.org/consortium/Governance_1203.pdf.

Hosting a Node
Beschreibt die Rechte und Pflichten von Mitgliedern, die in Zusammenhang mit PlanetLab-Nodes auftreten. Dieses Dokument befindet sich auf http://www.planet-lab.org/php/host/.

Acceptable Use Policy
Beschreibt Regeln zur Nutzung vom PlanetLab-Network und der Nodes. Zum Beispiel darf auf alle PlanetLab-Nodes nur via SSH zugegriffen werden. Dieses Dokument findet befindet sich auf http://www.planet-lab.org/php/aup/.

Administrator's Guide
Beschreibt alle wichtigen Aspekte im Zusammenhang mit der Wartung von PlanetLab-Nodes wie Node-Configuration usw. Dieses Dokument befindet sich auf http://www.planet-lab.org/doc/AdminGuide.php.

Connection Request auf der Webseite

Der nächste Schritt besteht darin, einen Connection Request auf http://www.planet-lab.org/consortium/join_request.php anzufordern. Dieser Connection Request muss für jeden Node einzeln angefordert werden. Dort müssen die Kontaktdaten für den Principal Investigator (PI) und eine Anspruchsperson für technische Probleme (Shutdown von Nodes usw.) angegeben werden. Die wichtigen Daten sind die Site-Informationen. Als erstes wird die URL der Site angegeben (z.B. http://informatik.uibk.ac.at), wobei bei

dieser Eingabe keine IP-Adressen erlaubt sind. Danach wird ein Site-Name angegeben (z.B. Universität Innsbruck). Als nächstes wird die IP-Adresse des Nodes innerhalb des eigenen LANs angegeben (z.B. 192.168.1.3). Ausserdem wird noch die Subnet-Maske des Nodes benötigt (z.B. 255.255.255.0).

4.1.4. Welche Rollen gibt es bei PlanetLab?

Principal Investigator (PI)

Der PI einer Site ist für jegliches Verhalten der Slices (siehe Abschnitt 4.1.6) dieser Site verantwortlich. Weiters muss der PI sicherstellen, dass sämtliche Nodes seiner Site den Anforderungen von PlanetLab gerecht werden. Er ist dafür verantwortlich, dass die Nodes ausserhalb von Firewalls liegen und ausreichend Bandbreite zur Verfügung haben.
Der PI ist ausserdem für die gesamte User- und Sliceverwaltung verantwortlich. Slices neu anlegen, User zu Slices hinzufügen, Ressourcen für Slices anfordern und neue User freischalten zählen zu den Aufgaben des PIs.

Administrator

Der Administrator übernimmt, wie der Name schon sagt die Administration der Nodes. Zu den Aufgaben eines Administrators zählen zum Beispiel die Konfiguration der Nodes (Installation, Bandwidth-Management, Starten und Beenden eines Nodes usw.). Natürlich kann es sich bei dem PI und dem Administrator um dieselbe Person handeln. Jedoch kann es auch sein, dass der PI die Administrationsarbeiten sozusagen “outsourcen“ möchte.

User

Alle Personen, die die Services von PlanetLab nutzen oder Anwendungen für PlanetLab entwickeln. User haben geringere Rechte als der Administrator oder der PI. Ein User kann zum Beispiel keinen Slice erzeugen.

4.1.5. Kontakt zu PlanetLab

Mailing Lists

PlanetLab stellt eine ganze Reihe von Mailing Lists zur Verfügung, sodass es am Anfang schwierig ist, die richtige für sein Problem auszuwählen.

Folgende Tabelle gibt einen Überblick über die Mailing Lists:

Mail – Adresse	**In welchem Zusammenhang verwenden?**
support@planet-lab.org	Fragen die Account-Management und Node-Installation betreffen.
info@planet-lab.org	Generelle Anfragen an das PlanetLab-Consortium (z.B. Presseanfragen).
announce@lists.planet-lab.org	Generelle Ankündigungen über PlanetLab (z.B. Nodeausfälle).
users@lists.planet-lab.org	Für den normalen User die wichtigste Liste. Hier können Fragen der Art “Wie mache ich dies und das in PlanetLab?“ gestellt werden.
arch@lists.planet-lab.org	Fragen, Vorschläge die die Architektur, Programmierung von PlanetLab - Software betreffen.
devel@lists.planet-lab.org	Fragen zur Programmierung von Systemen, die den PlanetLab-Kern betreffen.

Für die Listen announce@lists.planet-lab.org und users@lists.planet-lab.org erfolgt eine automatische Anmeldung.

Working Groups

Die gesamte PlanetLab-Community ist in sogenannte Working Groups unterteilt. Jede Gruppe konzentriert sich auf einen bestimmten Aspekt von PlanetLab.

Die Mailing Listen und Kontaktpersonen der wichtigsten Working Groups werden kurz dargestellt:

Architecture Working Group	
Mailing List:	arch@lists.planet-lab.org
Kontaktperson 1:	Larry Peterson, llp@cs.princeton.edu
Kontaktperson 2:	Timothy Roscoe, troscoe@intel-research.net

Planetary-Scale Commercial Services Working Group	
Mailing List:	pscs@lists.planet-lab.org
Kontaktperson 1:	Mic Bowman, mic.bowman@intel.com
Kontaktperson 2:	Rick McGeer, rick.mcgeer@hplcom

IT Advisory Board	
Mailing List:	it-advisory-board@lists.planet-lab.org
Kontaktperson 1:	Jeff Sedayao, jeff.sedayao@intel.com
Kontaktperson 2:	Scott Karlin, scott@cs.princeton.edu

Tools Working Group	
Mailing List:	tools@lists.planet-lab.org
Kontaktperson:	Robert Adams, robert.adams@intel.com

4.1.6. Wichtige Begriffe im Zusammenhang mit PlanetLab

Site

Das Unternehmen oder die Institution, in der sich die PlanetLab-Nodes befinden. Zurzeit sind es 209 Sites weltweit, wie zum Beispiel die Princeton University oder Intel Research in Berkeley.

Node

Ein Rechner, der zum PlanetLab-Network gehört und auf dem Anwendungen von PlanetLab-Services ausgeführt werden können. Jedes Mitglied muss mindestens zwei Nodes zur Verfügung stellen.

Slice

Eine über das PlanetLab-Network verteilte Anzahl an Ressourcen, die einer Anwendung eines PlanetLab-Service zugeordnet sind. Ein Slice ist sozusagen ein Subnet vom gesamten PlanetLab-Network, in dem der User seine Anwendungen ausführt bzw. implementiert.
Konkret wird ein Slice durch einen UNIX Login Account auf mehreren Nodes dargestellt. Der Slice wird vom PI erstellt und dieser teilt die User dem Slice zu. Der User kann dann bestimmen, welche Nodes er für seinen Slice haben will. Dadurch wird auf jedem dieser Nodes ein Login Account in einem Virtual Server erstellt.

Virtual Server

Die Umgebung, in der ein Slice ausgeführt wird. Sind einem Node mehrere Slices zugeordnet, so läuft für jeden Slice ein eigener Virtual Server. Die Virtual Server laufen dabei völlig unabhängig voneinander. Jeder Virtual Server verfügt über eine eigene Ressourcenverwaltung (z.B. Memory-Management).

Sliver

Der Teil eines Slice bzw. einer Anwendung, der auf einem Node in einem Virtual Server ausgeführt wird.

Sensor

Sensors bilden die Verbindung zwischen Slices die auf einem Node laufen, da diese nach Slices gruppierte Informationen zu einem Node liefern. Es gibt verschiedene Arten von Sensors wie z.B. den NOC-Sensor oder den Slice Stat Sensor. Der Slice Stat Sensor liefert beispielsweise Statistiken, nach Slices gruppiert (ähnlich wie “ps“ unter Linux). Allgemein kann gesagt werden, dass Sensors Slice-unabhängige und Slice-abhängige Informationen über einen oder mehrere Nodes liefern. Jedes Mitglied hat natürlich die Möglichkeit, Sensors nach individuellen Anforderungen zu entwickeln. Sensors werden im Kapitel 4.3, “Arbeiten mit PlanetLab“ noch genauer beschrieben.

Die folgende Abbildung illustriert den Zusammenhang zwischen Virtual Servers und Slivers. Die pinkfarbenen Quadrate repräsentieren dabei die unterschiedlichen Slivers, also eine Anwendung, die in dem Virtual Server ausgeführt wird.

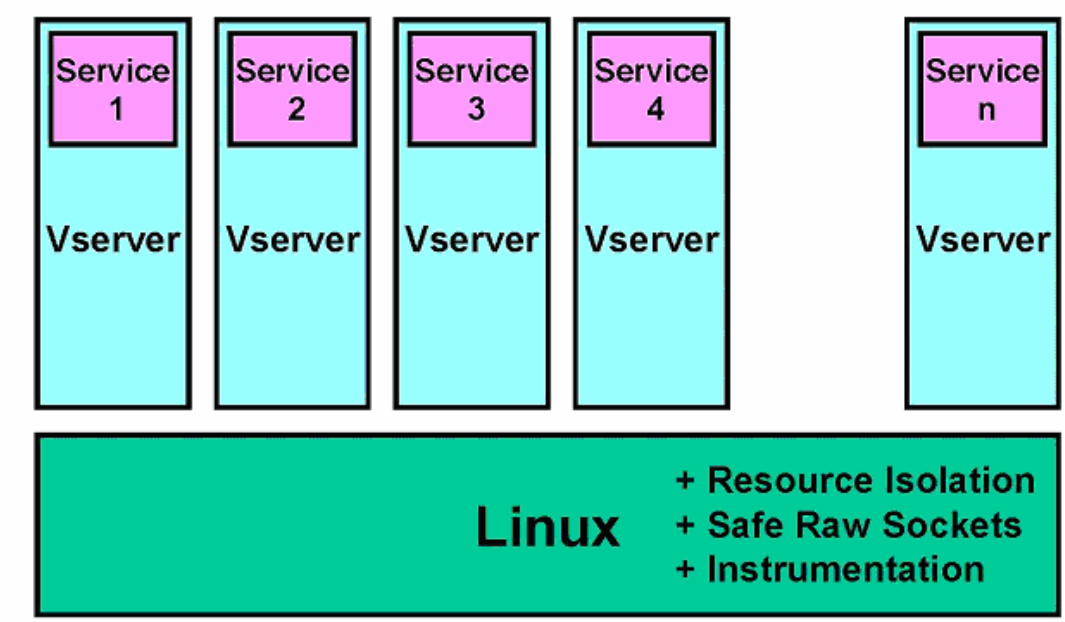

Abbildung 7: Modell eines PlanetLab-Nodes[6]

[6] Quelle: www.planet-lab.org/Talks/2003-11-20-PlanetLab-IEEE.pdf

Die folgende Abbildung verdeutlicht das grundlegende Konzept der Slices. Die pinkfarbenen und grünen Rechtecke repräsentieren die Slivers (die in einem Virtual Server laufen) und gehören jeweils zu einem Slice.

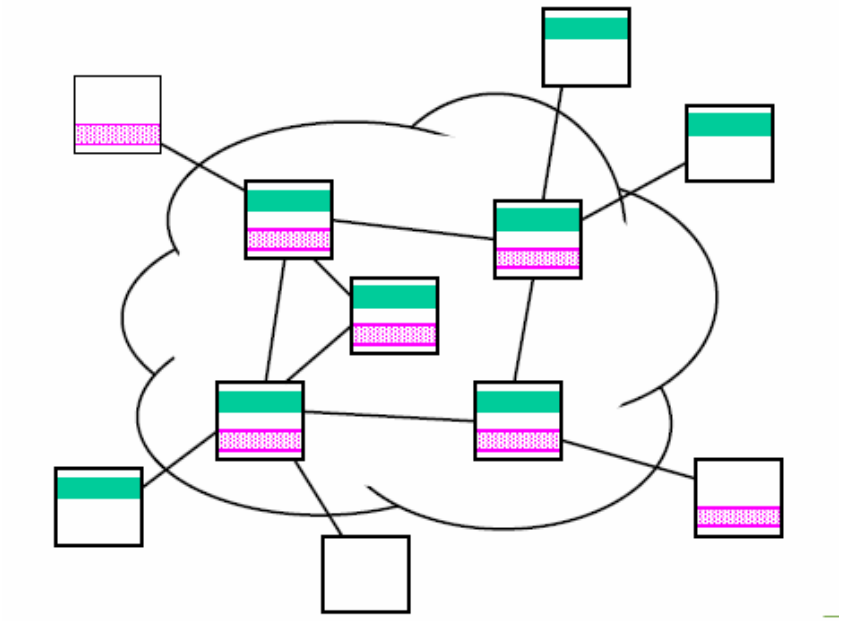

Abbildung 8: Konzept der Slices von PlanetLab[7]

4.1.7. Services, Anwendungen in PlanetLab

Die Anzahl an Services bzw. Anwendungen, die als Mitglied von PlanetLab genutzt werden können ist sehr gross und wächst ständig. Es gibt sehr viele verschiedene Bereiche, in denen PlanetLab verwendet wird; alle Services vorzustellen, würde den Rahmen dieser Arbeit sprengen. Aus diesem Grund werde in diesem Kapitel ein grundlegender Überblick über die Services gegeben. In Bezug auf das Thema dieser Arbeit sind besonders die Services im Bereich Network-Measurement, wie z.B. Scriptroute interessant. Mit diesen Services ist es möglich, Netzwerk-Tests durchzuführen. Scriptroute wird im Kapitel 4.3, “Arbeiten mit PlanetLab-Netzwerktests mit Scriptroute“ noch genauer beschrieben.

[7] Quelle: www.planet-lab.org/Talks/2003-11-20-PlanetLab-IEEE.pdf

Unterschied zwischen Service und Anwendung?

In den PlanetLab – Dokumentationen werden die Begriffe Service und Anwendung nicht eindeutig verwendet. Generell kann gesagt werden, dass eine Anwendung eine Teilmenge eines Service darstellt. Ein Service stellt allgemein ausgedrückt einen Dienst zur Verfügung, der von den Usern, Anwendungen oder anderen Services genutzt werden kann.

Die folgende Tabelle gibt einen Überblick über die verschiedenen Einsatzbereiche von PlanetLab und den dazugehörigen Services. Diese Übersicht wurde dem Dokument auf http://www.planet-lab.org/Talks/2003-04-15-HP-PlanetLab-Users.pdf entnommen.

Network measurement

- ScriptRoute (UWashington)
- PlanetProbe (Cambridge)
- I3 (UCBerkeley)
- Network Weather Service (UTK)
- BGP multiviews (Princeton)
- NetBait (Intel Berkeley)
- Ping (*everyone*)

Application-level multicast

- End-System Multicast (CMU)
- Scribe (Rice)
- TACT (Duke)
- Internet Backplane (UTK)

Distributed Hash Tables

- Chord / Koorde (MIT)
- Tapestry (UCB)
- Pastry (Rice)
- Kademlia (NYU)
- + cast of thousands using them

Storage

- Oceanstore (UCB)
- SFS, CFS, Ivy (MIT)
- Logistical Network (UTK)
- Palimpsest (IRB/Cambridge)

Resource Allocation

- SHARP (Duke, Intel Berkeley)
- Slices (Intel Berkeley, Princeton)
- Automated Contracts (UCB)
- DSlice (Intel Berkeley, Princeton)
- XenoCorp (Cambridge)

Distributed Query Processing

- PIER (UCB, ICIR, IRB)
- Sophia (Princeton)
- IrisNet (Intel Pittsburgh/CMU)
- TAG (Intel Berkeley)
- Distributed Search (MIT)

Content Distribution Networks

- CoDeeN (Princeton)
- Infranet/IRIS (MIT)
- ESM (CMU)
- UltraPeer emulation (UCB)
- Gnutella mapping (UWash)
- Gnutella measurement (UChicago)

Management and Monitoring

- Ganglia (UCB, Intel Berkeley)
- InfoSpect (Intel Berkeley)
- Scout Monitor (Intel SSL, Princeton)
- BGP Sensors (Princeton)
- Service Utilities (Duke)
- PlanetLab NMS (IRB/UWash)

Overlay Networks

- RON (MIT)
- RON++ (Princeton)
- XBone (Internet 2)
- ABone (Internet 2)
- ESM (CMU)

Virtualization and Isolation

- Xen (Cambridge)
- Denali (UWash)
- Vservers (Intel Berkeley)
- Mgmt VMs (Intel SSL)
- SILK/Scout (Princeton)
- DSlice (Intel Berkeley)

Router Design

- NetBind (Columbia)
- Scout/SILK (Princeton)
- NewArch (MIT)
- Icarus (UWash)

Testbed Federation

- PlanetLab (IRB, Princeton, UW)
- Emulab/Netbed (Utah)
- RON (MIT)
- Grid
- Xenoservers (Cambridge)

4.2. Installation von PlanetLab

4.2.1. Rechner mit PlanetLab verbinden

Falls die Mitgliedschaft genehmigt wurde und für jeden Rechner ein Connection Request gestellt wurde, kann begonnen werden die PlanetLab-Software auf den zukünftigen Nodes zu installieren.

Dabei wird wie folgt vorgegangen:

1. Lokale Konfigurationseinstellungen vornehmen
2. BootCD erstellen
3. Configuration File für jeden Node erstellen
4. PlanetLab-Betriebssystem installieren
5. PI-Account aktivieren lassen

1. Lokale Konfigurationseinstellungen vornehmen

Als erstes muss die Bootsequenz im BIOS einstellt werden. Die Rechner müssen als erstes von der CD booten und nicht von der Floppy oder Hard Disk! Alle anderen Bootfähigen Laufwerke können aus der Bootsequenz entfernt werden. Falls mehrere Festplatten vorhanden sind und ein RAID-Controller verwendet wird, muss sicherstellt werden, dass der Controller auf RAID 0[8] konfiguriert ist.

[8] zusammengeschlossene Festplatten ohne Fehlertoleranz (Redundanz)

Bevor mit der Installation begonnen werden kann, müssen die IP-Adressen, Hostnamen und Domain names für die Nodes identifiziert werden. PlanetLab schlägt bei der Namensvergabe der Nodes ein Schema wie "planetlab1", "planetlab2" usw. vor.

2. BootCD erstellen

Zum Zeitpunkt des Verfassens dieser Arbeit war die PlanetLab-Version 2.0 gerade aktuell. Daher beziehen sich sämtliche Angaben auf diese Version.

Für die Erstellung der BootCD werden folgende Files, die von http://www.planet-lab.org/download/ heruntergeladen werden können benötigt:

Filename	**Beschreibung**
PlanetLab-BootCD-2.0.iso.gz.sgn	Gepacktes 130 MB ISO-Image mit dem PlanetLab-Betriebssystem. Ist mit einem GPG[9]-Key verschlüsselt.
PlanetLab-BootCD-2.0.md5	Enthält die MD5-Checksummen zur Verifikation des Image-Files.
PlanetLab-Public-Key	Enthält den GPG-Public-Key zur Entschlüsselung des Image-Files.

[9] GPG = GNU-Privacy-Guard, Webseite: http://www.gnupg.org/

Die folgende Installationsanleitung wurde von http://www.planet-lab.org/consortium/setup_procedure.php übernommen und übersetzt. Zeilen, die mit # beginnen sind Kommentare. Die Zeilen mit fetter Schrift sind Eingaben an einer Linux-Shell.

```
# 1. PlanetLab-Public-Key importieren mit gpg
# (die Warnungen können ignoriert werden)
gpg --import PlanetLab-Public-Key

# 2. Image-File entschlüsseln
# (erstellt ein 123 MB gepacktes File, Warnungen könne wiederum
# ignoriert werden)
gpg --output PlanetLab-BootCD-2.0.iso.gz \
    --decrypt PlanetLab-BootCD-2.0.iso.gz.sgn

# 3. Image-file entpacken
# (erzeugt ein 450 MB ISO-Image-File)
gzip -d PlanetLab-BootCD-2.0.iso.gz

# 4. Verifikation der Installationsschritte
# (die Warnungen können wiederum ignoriert werden)
md5sum -c PlanetLab-BootCD-2.0.md5
```

Nach erfolgreicher Erstellung des ISO-Images muss dieses für jeden Node auf CD gebrannt werden.

3. Configuration File für jeden Node erstellen

Der nächste Schritt besteht darin, für jeden Node ein Configuration File (File-Extension: .cnf) zu erstellen. Dazu müssen auf http://www.planet-lab.org/consortium/bootcdconfig.php die geforderten Daten eingegeben werden. Es muss angegeben werden, ob die IP-Adressen statisch zugeteilt sind oder per DHCP zugeteilt werden. Weiters muss im Falle der statischen Zuweisung die IP-Adressen, Gateway-Adressen usw. angegeben werden. Weiters müssen der Hostname und die Domain für den Node angegeben werden. Abbildung 9 zeigt eine beispielhafte Konfiguration für einen Node, der seine Adressen per DHCP bezieht:

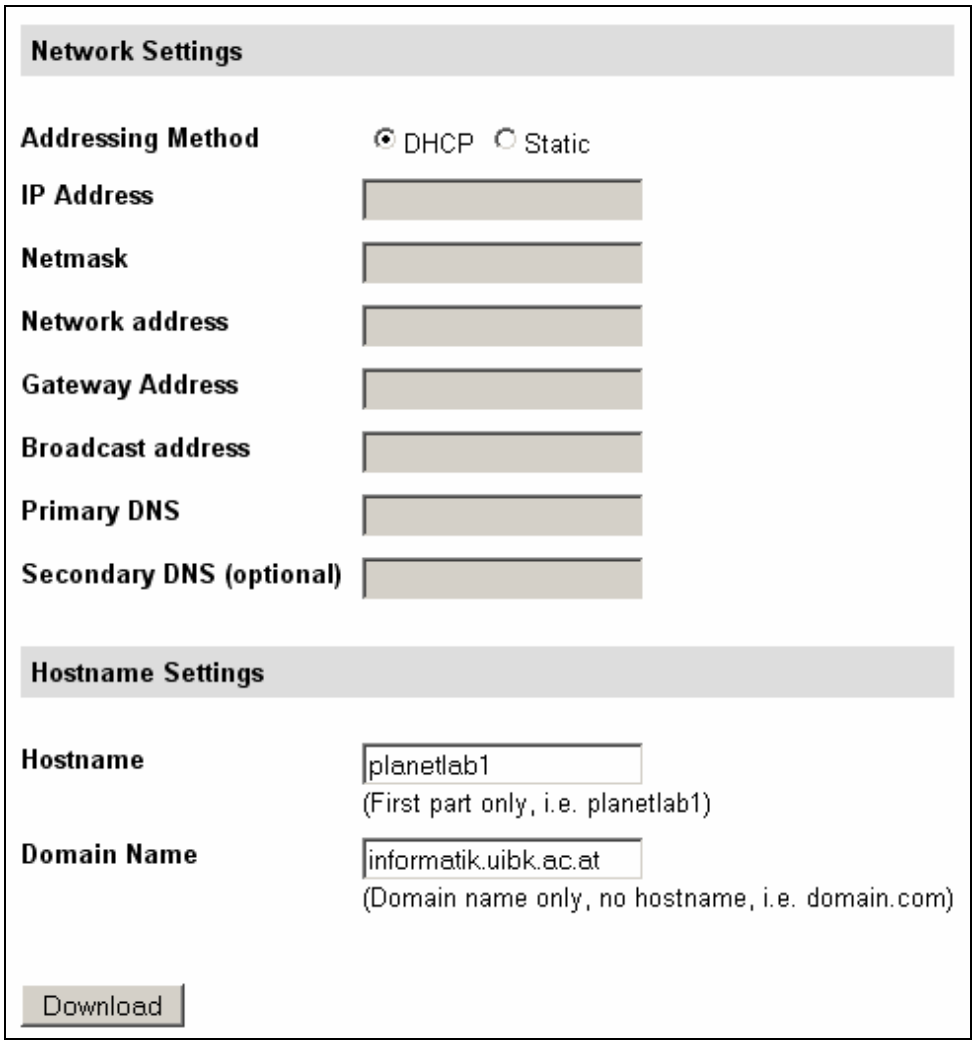

Abbildung 9: Formular für Configuration File[10]

[10] Quelle: http://www.planet-lab.org/consortium/join_request.php

Wenn alles fertig ausgefüllt ist, kann das .cnf – File über den Download-Button heruntergeladen und auf eine Diskette kopiert werden. Wichtig ist, dass das .cnf – File nicht mit einem Editor wie Wordpad bearbeitet wird, da sonst die UNIX-Zeilenumbrüche in Windows-Zeilenumbrüche konvertiert werden! Diese Diskette muss dann, gleich wie die BootCD, während des Betriebes des Nodes immer im Laufwerk bleiben!

Das .cnf - File für das Beispiel von Abbildung 9 hat folgende Einträge:

```
# DO NOT edit this file with a text editor that is
# not aware of linux/unix line feeds, like notepad
IP_METHOD="dhcp"
HOST_NAME="planetlab1"
DOMAIN_NAME="informatik.uibk.ac.at"
```

4. PlanetLab-Betriebssystem installieren

Nach dem Erstellen der BootCD und der Configuration Files kann nun das Betriebssystem installiert werden. Hierzu ist es notwendig, dass wie schon erwähnt vom CD-Laufwerk gebootet wird. Während der Installation und später während des Betriebs des Nodes müssen die BootCD und die Floppy mit dem Configuration File immer in den Laufwerken bleiben!
Bei dem PlanetLab-Betriebssystem handelt es sich zurzeit um eine RedHat 9 Linux-Distribution, die auf die Anforderungen von PlanetLab konfiguriert wurde. Zum Zeitpunkt der Verfassung dieser Arbeit wird ein modifizierter Linux 2.4.22 Kernel verwendet. Die wichtigste Modifikation am Kernel ist der VServer Patch, der es ermöglicht mehrere VServer auf einem Node auszuführen.

Die folgenden PlanetLab - Anwendungen bilden den Kern eines jeden PlanetLab-Systems auf einem Node:

- Modifizierter Linux 2.4.22 Kernel
- Plkmod
- Node Manager
- Installer

Bei Plkmod handelt es sich um ein Kernel-Modul, das für die Ressourcenaufteilung (z.B. CPU-Scheduling) zwischen verschiedenen Slices auf einem Node benötigt wird (slice isolation). Weiters wird durch Plkmod die Programmierung von Anwendungen mit Hilfe der SafeRawSockets ermöglicht. SafeRawSockets werden im Kapitel 4.3, "Arbeiten mit PlanetLab – Grundlagen" noch näher erläutert.

Der Node Manager (läuft auf jedem Node) ist ein authorisierter Prozess, der die Aktivitäten der anderen Slivers auf dem Node überwacht. Er überwacht und regelt die Ressourcenaufteilung zwischen den Slivers auf einem Node. Der Node Manager läuft auf jedem Node in einem eigenen Virtual Server, um die anderen Virtual Servers bzw. Slivers überwachen zu können.

Der Installer ist ein Tool, mit dem globale PlanetLab Software – Updates installiert werden. Mit diesem Tool werden Installationen durchgeführt, die alle PlanetLab Nodes betreffen und auf allen Nodes identisch sein sollten.

5. PI-Account aktivieren lassen

Wenn die Installation der Nodes fehlerfrei abgelaufen und diese online sind, wird der PlanetLab-Support mit der Mail Adresse support@planet-lab.org kontaktiert, damit der PI-Account aktiviert wird. Erst nach Aktivieren des PI-Accounts ist die Installation abgeschlossen.

4.2.2. PlanetLab User Accounts erstellen und freischalten

Jeder User muss seinen eigenen Account auf https://www.planet-lab.org/db/accounts/register.php erstellen. Um sich später auf den PlanetLab Nodes via SSH einloggen zu können, muss ein Public/Private Key erstellt werden, da SSH eine RSA-Authentifizierung verwendet. Um den Public/Private Key zu erstellen, wird SSH-Keygen folgendermassen verwendet:

```
ssh-keygen -t rsa -f ~/.ssh/identity
```

SSH-Keygen erstellt einen Private Key mit dem Namen identity und eine Public Key mit dem Namen identity.pub. Der Public Key kann nun im Zuge der Accounterstellung upgeloaded werden.

Dieser Account muss dann vom jeweiligen PI der Site bestätigt bzw. freigeschalten werden. Dies passiert, indem sich der PI auf https://www.planet-lab.org/db/login/login.php einloggt und den Account freischaltet.

4.2.3. PlanetLab-Slice erstellen

Um mit PlanetLab arbeiten zu können, muss für das eigene Projekt ein Slice erstellt werden, in dem dann die eigenen Netzwerktests durchgeführt werden. Das Erstellen eines Slice und das anschließende Zuteilen von Usern zu den Slices ist Aufgabe des PI. Es ist üblich für jedes Projekt einen eigenen Slice zu erstellen und dann die Bearbeiter dieses Projekts dem Slice zuzuordnen.

Falls der PI nun einen Slice erstellen möchte geht er folgendermassen vor:

- Die Seite https://www.planet-lab.org/db/login/login.php laden und einloggen
- Im Memberbereich die Seite “Dynamic Slices“ öffnen (befindet sich unterhalb von Account/Slices)
- Oben auf der Seite `Create` auswählen
- Bezeichnung für Slice eingeben. Es wird automatisch ein Kürzel für den eigenen Site-Namen vorangestellt.
- Oben auf der Seite `Assign users` auswählen und die User dem Slice zuordnen

Eine Übersicht über die aktuellen existierenden Slices findet sich auf https://www.planet-lab.org/php/slices.php. Ein Slice hat eine Lebensspanne von 14 Tagen, kann aber vom PI verlängert werden, indem er auf der “Dynamic Slices“ Seite die Renew - Option benützt.

4.2.4. Node-Konfiguration

Da es sich bei dem PlanetLab-Betriebssystem um ein “abgespecktes“ RedHat 9 Linux handelt, können Programme mit den üblichen Tools wie beispielsweise YUM (yellowdog updater modified) installiert werden. YUM ist ein Tool zur automatischen Installation von rpm-Paketen; eine ausführliche Dokumentation befindet sich auf http://linux.duke.edu/projects/yum/. Im Folgenden werden kurz die nötigen Schritte beschrieben, um YUM auf PlanetLab-Nodes nützen zu können.

Zuerst muss Python 2.2 installiert werden. Die Installation von Python und YUM kann mittels eines Skripts (setup_yum.sh) von der PlanetLab-Website geschehen. Um das Skript zu starten wird die folgende Eingabe verwendet:

```
curl http://boot.planet-lab.org/alpina/otherscripts/setup_yum.sh | bash
```

Nach dem Ausführen des Skripts ist YUM installiert und es können Tools installiert werden, um den Node nach individuellen Wünschen zu konfigurieren. Für Infos zur Verwendung von YUM sei wiederum auf http://linux.duke.edu/projects/yum/ verwiesen.

4.3. Arbeiten mit PlanetLab - Grundlagen

4.3.1. Wie kann ich mich auf meinem Slice einloggen?

Der User hat die Möglichkeit, sich per SSH in jeden Node einzuloggen, der zu seinem Slice gehört (vorausgesetzt, er hat seinen Public Key hochgeladen). Unter der Annahme, dass der Slice site_exp2 existiert und der Node “planetlab1.informatik.uibk.ac.at“ dem Slice zugeordnet ist, kann sich der User folgendermassen einloggen:

```
ssh site_exp2@planetlab1.informatik.uibk.ac.at
```

Alle User, die im selben Slice arbeiten besitzen die gleiche UID in der Virtual Server Umgebung in der der Slice läuft. Der User hat auf den Nodes `sudo` root-Privilegien, wobei das Passwort das leere Wort ist.

4.3.2. Wie kann ich Ping und Traceroute in meinem Slice nutzen?

Die Standardimplementierungen von Ping und Traceroute funktionieren in PlanetLab nicht, da die Standardversionen keine Safe Raw Sockets benützen. Ping und Traceroute sind aber nützlich um Statusinformationen über bestimmte Nodes in PlanetLab zu erhalten. Es müssen also Implementierungen der zwei Tools installiert werden die Safe Raw Sockets benützen.

Die Tools (bzw. die RPM-Pakete) können von der PlanetLab-Website folgendermassen installiert werden:

```
rpm -Uvh http://boot.planet-lab.org/install-rpms/planetlab/iputils-pl-
20020927-3.planetlab.i386.rpm

rpm -Uvh http://boot.planet-lab.org/install-rpms/planetlab/traceroute-
pl-1.4a12-10.planetlab.i386.rpm
```

Nach erfolgreicher Installation kann dann mit z.B. “ping planetlab1.informatik.uibk.ac.at“ ein Ping auf den Node mit der Bezeichnung planetlab1 der Site informatik.uibk.ac.at ausgeführt werden.

Eine Besonderheit mit Ping gibt es in PlanetLab zu beachten! Es kann sein, dass das Einloggen mit SSH in einen Node zwar funktioniert, jedoch dieser mit Ping nicht erreicht wird. Der Grund dafür liegt darin, dass manche PlanetLab-Sites aus Gründen der Sicherheit Pings filtern. Falls also festgestellt werden soll, ob ein Node im Slice verfügbar ist und mit Ping kein Erfolg erzielt wurde, sollte um sicherzugehen auch noch versucht werden, sich direkt per SSH auf dem Node einzuloggen. Falls auch dies scheitert ist der Node tatsächlich nicht verfügbar.

4.3.3. Wie kann ich die PlanetLab-Sensors nutzen?

In diesem Kapitel werden die wichtigsten Sensors vorgestellt. Die bedeutendsten Sensors sind der Slice-Stat-Sensor, der NOC-Sensor und der Proc-Sensor.

Die in den folgenden Überschriften angegebenen Zahlen in den Klammern beziehen sich jeweils auf den Port, auf dem der Sensor arbeitet.

1. Slice-Stat-Sensor (2840)

Der Slice-Stat-Sensor stellt dem User umfangreiche Statistiken zu den PlanetLab-Slices zur Verfügung. Der Slice-Stat-Sensor arbeitet auf Port 2840. Die Dokumentation zu diesem Sensor findet sich auf http://berkeley.intel-research.net/bnc/slicestat/. Um den Sensor zu verwenden wird folgendes eingegeben:

```
curl -s http://127.0.0.1:2840/slicestat
```

Der Slice-Stat-Senor liefert nun für jeden Slice ein Tupel von Feldern mit Informationen. Zum Zeitpunkt der Erstellung dieser Arbeit umfasst ein Tupel folgende Felder in dieser Reihenfolge:

Feld	**Information/Slice**
Slice	Slice Login-Name (z.b. irb2)
Ctx	VServer Context-ID
%cpu%	Gibt die Nutzung der CPU/Slice in Prozent an
%mem%	Gibt die Nutzung von Physical-Memory/Slice in Prozent an
Pmem	Gibt die Nutzung von Physical-Memory/Slice in KB an
Vmem	Gibt die Nutzung von Virtual-Memory/Slice in KB an
Ntasks	Gibt die Anzahl der Tasks/Slice an
send_bw1	Gibt die Bandbreite (in Kbit/s) des Download-Traffics der letzten Minute an
send_bw5	Gibt die Bandbreite (in Kbit/s) des Download-Traffics der letzten 5 Minuten an
send_bw15	Gibt die Bandbreite (in Kbit/s) des Download-Traffics der letzten 15 Minuten an
recv_bw1	Gibt die Bandbreite (in Kbit/s) des Upload-Traffics der letzten Minute an
recv_bw5	Gibt die Bandbreite (in Kbit/s) des Upload-

	Traffics der letzten 5 Minuten an
recv_bw15	Gibt die Bandbreite (in Kbit/s) des Upload-Traffics der letzten 15 Minuten an
Ip	Gibt die lokale IP-Adresse des Nodes an, von wo der Sensor ausgeführt wurde

Da der Output von Slicestat ziemlich unübersichtlich ist, sollte die Ausgabe von Slicestat mit Hilfe einer Pipe auf z.B. Perl umgeleitet werden. Dann kann mit Perl die Ausgabe lesefreundlich formatiert werden. Am einfachsten ist es, eine Bash-Funktion zu schreiben, welche die Formatierung automatisch durchführt.

Die Bash-Funktion könnte folgendermassen aussehen:

```
ptop ()
  {
    curl -s http://127.0.0.1:2840/slicestat |
    perl -ne 'chomp; @F=split /,/; printf "%-20s %4d %4.1f %4.1f %5d
     %5d  %3d | %6.2f %6.2f %6.2f | %6.2f %6.2f %6.2f\n", @F';
  }
```

Die Ausgabe von curl wird an Perl weitergeleitet, wo dann mit der printf()-Funktion die Ausgabe formatiert wird. Die lokale IP-Adresse wird nicht angegeben.

Folgendes Beispiel zeigt einen möglichen Output nach dem Aufruf von ptop():

```
irb_x1         632  0.1  3.4 12288 23960   1 |   0.18   0.24   0.27 |   0.65   1.02   1.18
torino_http    600  0.0  0.1   576  2596   2 |   0.00   0.00   0.00 |   0.00   0.00   0.00
irb_idmef      606  0.0  0.0    40   108   1 |   0.00   0.00   0.00 |   0.00   0.00   0.00
mit_dht        568  2.4  1.6  7284  7460   5 |  38.45  39.30  39.45 |  36.81  38.59  38.73
idsl1          502 14.2  1.2  4912  6040   1 |   0.00   0.02   0.03 | 202.75 176.42 173.96
irb2           503  2.0  4.0 16168 47548   7 |   0.00   0.00   0.00 |   0.00   0.00   0.00
ucb_pier_2     522 13.8  8.7 34164 61148   6 |  23.01  14.20  12.72 |  20.73  15.69  14.66
```

2. NOC-Sensor (33080)

Der NOC-Sensor liefert globale Informationen zu den PlanetLab-Nodes. Der NOC-Sensor arbeitet auf Port 33080. Der Sensor wird folgendermassen verwendet:

```
curl http://localhost:33080/nodes/ + Parameter i. d. Form: /parameter1/..
```

Folgende Parameter sind verfügbar:

Parameter	**Bedeutung**
Name	DNS-Name des Nodes
Lat	Geographische Breite des Nodes (der Site)
Long	Geographische Länge des Nodes (der Site)
Model	Hersteller bzw. Modell des Rechners
Category	Gibt Verwendungstatus des Nodes an; folgende sind möglich: • Production – voll einsatzfähig • Beta – wird konfiguriert und getestet • Alpha – f. interne Tests verwendet
Ip	IP-Adresse des Nodes
url	URL der Webseite der Institution, die den Node zur Verfügung stellen
Site	Name der PlanetLab-Site der Institution
bootCD	Version der verwendeten BootCD

Um den NOC-Sensor mit den Parametern name, lat und long aufzurufen wird die folgende Eingabe verwendet:

```
curl http://localhost:33080/nodes/name/lat/long
```

Der Output könnte beispielsweise so aussehen:

```
planetlab2.eurecom.fr,46.0,2.0
planetlab1.eurecom.fr,46.0,2.0
planetlab3.cs.uoregon.edu,44.04,-123.06
planetlab2.cs.uoregon.edu,44.04,-123.06
planetlab1.cs.uoregon.edu,44.04,-123.06
planetlab2.cs.cornell.edu,42.4478,-76.476
planetlab1.cs.cornell.edu,42.4478,-76.476
planetlab1.inria.fr,46.0,2.0
```

3. Proc-Sensor (7890)

Der Proc-Sensor liefert Systeminformationen und -statistiken pro PlanetLab-Node. Der Proc-Sensor arbeitet auf Port 7890. Der Sensor wird folgendermassen verwendet:

```
curl http://localhost:7890/proc?verbose
```

Für den obigen Aufruf könnte beispielsweise folgende Ausgabe erfolgen:

```
Host=planetlab-1.cmcl.cs.cmu.edu,
CurrentTime=1077565036289,
cpu_MHz=1263.611 ,
mem_total(KB)=1031264,
mem_free(KB)=16864,
mem_shared(KB)=0,
mem_buffers(KB)=3300,
swap_total(KB)=1048568,
swap_free(KB)=8,
load_one=0.99,
load_five=0.65,
load_fifteen=0.34,
proc_run=1.0,
proc_total=643.0,
boot_time=1074795784,
cpu_user=11.25,
cpu_nice=0.00,
cpu_idle=72.13,
cpu_system=16.62,
cpu_aidle=42.26,
bytes_in(/sec)=37820.00,
pkts_in(/sec)=219.53,
bytes_out(/sec)=33243.53,
pkts_out(/sec)=182.06,
TotalDisk(MB)=62427,
FreeDisk(MB)=48769
```

Die Parameter sind selbsterklärend und werden an dieser Stelle nicht näher erläutert.

4.3.4. Wie kann ich Anwendungen für PlanetLab entwickeln?

Um in PlanetLab eigene Anwendungen in der Sprache C für z.B. Netzwerktests entwickeln zu können wird das Safe-Raw-Socket API verwendet. Mit Hilfe von Safe-Raw-Sockets kann auf Netzwerkdaten wie zum Beispiel IP, ICMP, UDP oder TCP Header zugegriffen werden. Um PlanetLab-fähige Anwendungen zu entwickeln, muss das Safe-Raw-Socket API verwendet werden. Zu beachten ist, dass jeweils nur der Traffic vom eigenen Slice analysiert werden kann. Es kann nur von Ports gesendet oder empfangen werden, die dem Slice zugeteilt sind. Zur Verwendung von Safe-Raw-Sockets muss die Header-Datei “planetlab.h“ eingebunden werden.

Das Konzept der Safe-Raw-Sockets ist sehr eng an die Socketprogrammierung in C unter Linux gebunden. D.h. die Vorgangsweise bei der Entwicklung von Anwendungen entspricht zum größten Teil der unter Linux. Auf die Socketprogrammierung unter Linux wird in dieser Arbeit nicht näher eingegangen; es wird auf die zahlreichen Dokumentationen verwiesen.

Die wichtigsten Schritte zur Erstellung von Safe-Raw-Sockets werden nun näher vorstellt. Für eine detaillierte Dokumentation der Safe-Raw-Sockets sei auf http://www.planet-lab.org/raw_sockets/index-one.html verwiesen.

1. Safe-Raw-Socket erstellen

Zum Erstellen eines Safe-Raw-Sockets wird der Standard-socket()-Aufruf verwendet. Jedoch gibt es ein paar Besonderheiten zu beachten:

- Der erste Parameter (domain) muss auf PF_INET gesetzt werden
- Der zweite Parameter (Sockettyp) muss auf SOCK_RAW gesetzt werden
- Der dritte Parameter (Protokoll) muss eine der folgenden Konstanten sein:

Konstante bzw. Protokollname	**Bedeutung**
IPPROTO_TCP	Es werden TCP-Pakete gesendet/empfangen.
IPPROTO_UDP	Es werden UDP-Pakete gesendet/empfangen.
IPPROTO_ICMP	Es werden ICMP-Echo-Requests gesendet und ICMP-Echo-Replies empfangen (wird für PING benötigt).
IPPROTO_ICMP_TCP	Socket der für einen bestimmten TCP-Port ICMP-Nachrichten empfängt.
IPPROTO_ICMP_UDP	Socket der für einen bestimmten UDP-Port ICMP-Nachrichten empfängt.

Der folgende Beispielcode erzeugt einen TCP-Socket:

```
sock = socket(PF_INET, SOCK_RAW, IPPROTO_TCP);
```

2. Safe-Raw-Socket binden

Zum Binden des Sockets an einen bestimmten Port oder im Falle eines ICMP-Sockets an einen bestimmten ICMP-Identifier, wird der Standard-bind()-Aufruf verwendet.

Der folgende Beispielcode bindet den TCP-Socket vom obigen Beispiel an den Port 9090 zum Senden und Empfangen von TCP-Paketen:

```
struct sockaddr_in sin;

memset(& sin, 0, sizeof(sin));
sin.sin_port = htons(9090);

bind(sock, (struct sockaddr *)& sin, sizeof(sin));
```

3. Daten versenden/empfangen

Es können die üblichen Systemfunktionen wie *sendto*, *sendmsg*, *recv*, *recvfrom*, *recvmsg* oder *select* verwendet werden. Zu beachten ist, dass die Funktion *send* nicht verwendet werden darf.

Um Pakete zu senden/empfangen, die neben dem jeweiligen Protokollheader (TCP, UDP, ICMP) auch den IP-Header beinhalten, muss dies mit der Konstante IP_HDRINCL gesetzt werden (nach erfolgreichem bind()-Aufruf).

Der folgende Beispielcode zeigt das Setzen von IP_HDRINCL mit Hilfe des setsockopt()-Aufrufs:

```
int tmp = 1;

setsockopt(sock, 0, IP_HDRINCL, & tmp, sizeof(tmp));
```

Um einen Safe-Raw-Socket zu schliessen wird der Standard-close()-Aufruf verwendet.

4. Näheres zur Verwendung von ICMP-Sockets

Durch die Angabe des ICMP-Identifiers im bind()-Aufruf ist es möglich, nur ICMP-Nachrichten zu senden/empfangen, die den gleichen ICMP-Identifier haben.

Der folgende Beispielcode zeigt das Binden eines ICMP-Sockets an den ICMP-Identifier mit der Nummer 23456. Nur Nachrichten mit dem Identifier 23456 können von diesem Socket gesendet/empfangen werden.

```
struct sockaddr_in sin;

sock = socket(PF_INET, SOCK_RAW, IPPROTO_ICMP);

memset(& sin, 0, sizeof(sin));
sin.sin_port = htons(23456);

bind(sock, (struct sockaddr *)& sin, sizeof(sin);
```

Es ist nicht erlaubt, dass zwei verschiedene Slices einen ICMP-Socket an den gleichen TCP/UDP-Port oder ICMP-Identifier binden. Für Details zum Thema Port-Management sei auf http://www.planet-lab.org/raw_sockets/ports.html verwiesen.

5. Sniffer-Sockets

Ein Slice hat die Möglichkeit einen sogennanten “Sniffer-Socket“ an einem Port zu erstellen um IP-Pakete an diesem Port zu überwachen. Sniffer-Sockets sind read-only, d.h. es erfolgt ausschliesslich lesender Zugriff auf Daten.

Um einen Sniffer-Socket zu erstellen muss mit dem setsockopt()-Aufruf SO_RAW_SNIFF vor dem bind()-Aufruf gesetzt werden.

Der folgende Beispielcode zeigt die Vorgangsweise beim Erstellen eines Sniffer-Sockets:

```
#include < planetlab.h >

int tmp, sock;
struct sockaddr_in sin;

sock = socket(PF_INET, SOCK_RAW, IPPROTO_TCP);

tmp = 1;
setsockopt(sock, 0, SO_RAW_SNIFF, & tmp, sizeof(tmp));

sin.sin_port = htons(1234);

bind(sock, (struct sockaddr *)& sin, sizeof(sin));
```

Für eine detaillierte Beschreibung von Sniffer-Sockets und den Einschränkungen im Zusammenhang mit Sniffer-Sockets sei auf http://www.planet-lab.org/raw_sockets/api_sniffer.html sowie auf http://www.planet-lab.org/raw_sockets/api_restrictions.html verwiesen.

6. Realisierung von Ping mit Safe-Raw-Sockets

Das unten folgende Beispiel zeigt eine Implementierung von BSD/Unix Ping unter Verwendung der Safe-Raw-Sockets. Das Beispiel wurde von http://www.cs.huji.ac.il/labs/danss/planetlab/SafeRawSockets.pdf übernommen. Die rot markierten Codeausschnitte markieren die Änderungen die gemacht wurden, damit das Programm unter PlanetLab verwendet werden kann.

Die folgende Aufzählung erläutert die wichtigsten Punkte:

- Einbinden von “planetlab.h“
- Socket muss an einen lokalen Port gebunden werden
- IP_HDRINCL setzen, zur Inkludierung des IP-Headers
- Den IP-Header explizit definieren
- Verwendung des structs icmphdr anstelle des structs icmp

Mögliche Implementierung von Ping unter PlanetLab in der Sprache C:

```
#include "planetlab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/types.h>
#include <netinet/ip_icmp.h>

#define BUFFER_SIZE 1024

u_short in_cksum(const u_short *addr, register int len, u_short csum)
{
        register int nleft = len;
        const u_short *w = addr;
        register u_short answer;
        register int sum = csum;

        /**  Our algorithm is simple, using a 32 bit accumulator (sum),
```

```
        *  we add sequential 16 bit words to it, and at the end, fold
        *  back all the carry bits from the top 16 bits into the lower
        *  16 bits.*/

        while (nleft > 1) {
                sum += *w++;
                nleft -= 2;
        }

        /* mop up an odd byte, if necessary */
        if (nleft == 1)
                sum += htons(*(u_char *)w << 8);

        /*add back carry outs from top 16 bits to low 16 bits*/
        sum = (sum >> 16) + (sum & 0xffff);
        /* add hi 16 to low 16 */
        sum += (sum >> 16);             /* add carry */
        answer = ~sum;                  /* truncate to 16 bits */
        return (answer);

}

int main(int argc, char * argv[])
{
    int sock;
    struct sockaddr_in sin;
    unsigned short local_port;
    unsigned char protocol;
    char * buffer, * buffer2, * dnsdata;
    struct iphdr * ip_header;
    struct icmphdr * icmp_header;
    char * remote_ip_str;
    unsigned short buffer_size, buffer_size2;
    int tmp, len;
    short randomseq;

    if (argc != 3) {
        fprintf(stderr, "USAGE: %s icmp-id destination\n", argv[0]);
        return 1;
    }

    protocol = IPPROTO_ICMP;

    local_port = atoi(argv[1]);
    remote_ip_str = argv[2];

    if ((sock = socket(PF_INET, SOCK_RAW, protocol)) < 0) {
        perror("socket");
        exit(1);
    }
    memset(& sin, 0, sizeof(sin));
    sin.sin_port = htons(local_port);

    if ((bind(sock, (struct sockaddr *)& sin, sizeof(sin))) < 0) {
        perror("bind");
        exit(1);
    }

    tmp = 1;
    setsockopt(sock, 0, IP_HDRINCL, & tmp, sizeof(tmp));

    buffer_size = sizeof(struct iphdr) + sizeof(struct icmphdr)
                  + sizeof(struct timeval);
```

```
    buffer_size2 = BUFFER_SIZE;
    buffer = (char *) malloc(buffer_size);
    buffer2 = (char *) malloc(buffer_size2);

    memset(buffer, 0, sizeof(buffer));

    memset(& sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = inet_addr(remote_ip_str);

    ip_header = (struct iphdr *) buffer;
    ip_header->ihl = 5;
    ip_header->version = 4;
    ip_header->tos = 0;
    ip_header->tot_len = htons(buffer_size);
    ip_header->id = rand();
    ip_header->ttl = 64;
    ip_header->frag_off = 0x40;
    ip_header->protocol = protocol;
    ip_header->check = 0; /* This will be done in the kernel */
    ip_header->daddr = inet_addr(remote_ip_str);
    /* Leave src IP address blank, kernel will fill it out. */
    ip_header->saddr = 0;

    icmp_header = (struct icmphdr *) (ip_header + 1);
    icmp_header->type = ICMP_ECHO;
    icmp_header->code = 0;
    icmp_header->un.echo.id = htons(local_port);
    icmp_header->un.echo.sequence = 0;

    struct timeval *tp = (struct timeval *)& buffer[28];
    gettimeofday(tp, NULL);

    icmp_header->checksum = in_cksum((const u_short *) icmp_header,
                sizeof(struct icmphdr) + sizeof(struct timeval), 0);

    if (sendto(sock, buffer, buffer_size, 0, (struct sockaddr *) &sin,
        sizeof(struct sockaddr_in)) < 0)
    {
        perror("sendto");
    }

    printf("sent query\n");

    len = sizeof(sin);
    if (recvfrom(sock, buffer2, buffer_size2, 0, (struct sockaddr *)
        &sin, &len) < 0)
    {
        perror("recvfrom");
        return 1;
    }
    printf("received response\n");

    close(sock);
    return 0;
}
```

4.3.5. Slice-Management

PlanetLab-Administratoren haben die Möglichkeit, mittels eines speziellen `site_admin` – Accounts typische Administratortätigkeiten (ähnlich wie `root`-Account) durchzuführen. Der `site_admin` – Account wird automatisch bei der Installation eines Nodes erstellt. Der Administrator hat die Möglichkeit mit `sudo` folgende administrative Funktionen auszuführen:

`/usr/local/planetlab/bin/pl-ps`	Liste der laufenden Prozesse mit dem zugehörigen VServer wird ausgegeben. (Normales 'ps' liefert falsche VServer-IDs!)
`/usr/local/planetlab/bin/pl-catlogs`	Log-Files in /var/log können mit Files konkateniert werden (können also vom Administrator eingesehen werden).
`/usr/local/planetlab/bin/pl-limitbw`	Bandwidth-Limits können für einen Node gesetzt werden. Der Aufruf des Kommandos mit dem Parameter "on" bewirkt eine Limitierung der Bandwidth auf 5 MBit/s.
`/usr/sbin/tcpdump`	`tcpdump`. Ausführen
`/sbin/shutdown`	Rechner (Node) ausschalten.

4.4. Arbeiten mit PlanetLab – Netzwerktests mit Scriptroute

4.4.1. Was ist Scriptroute?

Scriptroute ist eine verteilte Testplattform für Netzwerktests und wurde von der University of Washington entwickelt. Die offizielle Webseite von Scriptroute ist http://www.scriptroute.org. Das Prinzip von Scriptroute basiert auf Ruby-Skripten (die sogenannten "scriptroute-measurement-scripts") mit speziell entwickelten Klassen für Netzwerktests, die entweder lokal oder remote auf Scriptroute-Servern interpretiert werden können. Der Scriptroute-Service wird von PlanetLab mit ca. 200 Servern unterstützt. Für die effektive Verwendung von Scriptroute ist es sinnvoll, sich für die Scriptroute-Mailing-List anzumelden, welche auf https://mailman.cs.washington.edu/mailman/listinfo/scriptroute zu finden ist.

4.4.2. Welche Netzwerktests kann ich mit Scriptroute machen?

Im Zuge der Installation von Scriptroute werden einige Beispielskripte installiert. Darunter Implementierungen von Ping, Traceroute und SProbe. Mit Hilfe von SProbe kann die Bandbreite zwischen dem eigenem Rechner und einem Zielrechner unter Angabe der IP-Adresse abgeschätzt werden.

Mit Hilfe der Scriptroute-Paketklassen IP, ICMP, UDP und TCP können Pakete der jeweiligen Protokolle erzeugt werden und verschiedene Optionen und Flags der jeweiligen Protokollheader gesetzt werden. Somit ist es möglich, individuelle Netzwerktests zu programmieren. In dem weiter unten folgenden UML-Klassendiagramm sind die oben erwähnten Klassen dargestellt.

4.4.3. Wie funktioniert Scriptroute?

Scriptroute kann prinzipiell in drei Teile gegliedert werden: der Home-Server, der Client und die Scriptroute-Server. Im Folgenden werden diese näher erläutert:

1. Home-Server (DNS-Directory Server)

Der Home-Server verfügt über eine dynamisch aktualisierte DNS-Datenbank, in der immer die aktuell verfügbaren Scriptroute-Server eingetragen sind. Der Client verbindet sich nun per HTTP zum Home-Server und erhält eine Server-Liste mit den aktuell verfügbaren Scriptroute-Servern. Auf der Webseite http://www.scriptroute.org:3967/ können interaktiv die aktuellen Server abfragt werden. Dieser DNS-Directory Server ist auch der in der Konfigurationsdatei eingetragene Standard, ebenso der Port (3967). Der zu verwendende Home-Server und der Port werden in der Konfigurationsdatei scriptroute.conf konfiguriert.

2. Client

Der Rechner, der die Skripte ausführen möchte. Entweder auf dem lokalen Scriptroute-Server oder Remote per Http-POST auf einem Scriptroute-Server.

3. Scriptroute Server

Der Scriptroute-Server erhält über das Front-End (CGI-Interface) die Skripte vom Client und gibt sie an den Scriptroute-Interpreter (Ruby-Interpreter) weiter. Der Interpreter interagiert über das Send-train API mit dem Scriptroute-Daemon (in der Abbildung als Network Guardian bezeichnet) und dieser entscheidet welche Pakete wann bzw. wie gesendet werden.

Die folgende Abbildung zeigt die Funktionsweise von Scriptroute detailliert:

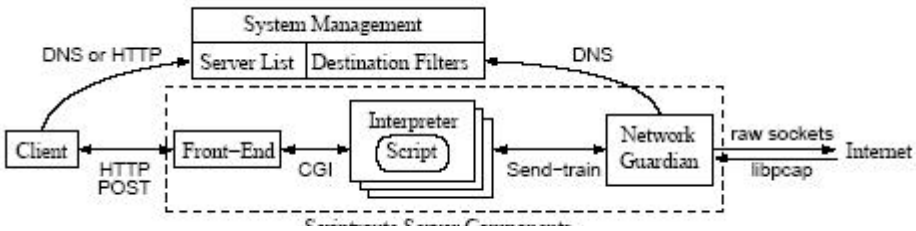

Abbildung 10: Funktionsweise von Scriptroute[11]

[11] Quelle: http://www.cs.washington.edu/research/networking/scriptroute/papers/installation.pdf

4.4.5. Installation von Scriptroute unter PlanetLab

Um Scriptroute unter PlanetLab installieren zu können, wird ein spezielles Skript mit dem Namen “planetlab-install“ bereitgestellt, dass auf http://www.scriptroute.org/planetlab/planetlab-install heruntergeladen werden kann. Mit dem folgenden Befehl wird das Skript heruntergeladen und ausgeführt:

```
wget -nc http://www.scriptroute.org/planetlab/planetlab-install
     && sh ./planetlab-install
```

4.4.6. Beispielskript - Ping

Das untenfolgende Beispielskript zeigt eine Implementierung des Ping-Befehls unter Verwendung von Ruby und den Scriptroute-Klassen. Das Beispiel wurde dem Scriptswriter’s Guide entnommen, welcher die offizielle Dokumentation zur Erstellung eigener Skripte darstellt.

```
#! /usr/bin/srinterpreter

probe = Scriptroute::Icmp.new(0) # anything else might not get a
                                 # response.
probe.ip_dst = ARGV[0]
probe.icmp_type = Scriptroute::Icmp::ICMP_ECHO
probe.icmp_code = 0
probe.icmp_seq = 1

last = Time.now - 1.0;

( 1..10 ).each { |rep|
  probe.icmp_seq = rep
  delay = 1 - (Time.now-last)

  packets = Scriptroute::send_train([ Struct::DelayedPacket.new(1 -
                                    (Time.now-last),probe) ])

  if(packets[0].response) then
    response = packets[0].response.packet
    rtt = (response) ? ((packets[0].response.time -
                         packets[0].probe.time) * 1000.0) : '*'
    if(response.is_a ? (Scriptroute::Icmp)) then
      puts rep.to_s + ' ' + response.ip_src.to_s + ' %5.3f ms' % rtt
    end
    last = Time.at(packets[0].probe.time);
  else
    puts '' + rep.to_s + ' to ' + probe.ip_dst.to_s + ' timed out'
  end
  $stdout.flush
}
```

Es werden insgesamt 10 ICMP-Nachrichten gesendet. Dafür wird der send_train – Aufruf von Scriptroute verwendet. Der erste If-Block dient zur Unterscheidung, ob eine Antwort empfangen wurde oder nicht. Fall dies der Fall ist, wird in der zweiten If-Anweisung überprüft, ob es sich um eine ICMP-Nachricht handelt (manche Server antworten mit UDP). Falls auch dies der Fall ist, so wird die RTT berechnet und zusammen mit der IP und der Paketnummer ausgegeben. Falls keine Antwort gesendet wurde, wird "timed out" zusammen mit der IP und der Paketnummer ausgegeben.

4.4.7. Eigene Netzwerktests für Scriptroute entwickeln

Die Programmiersprache Ruby

Falls individuelle Skripte für Scriptroute entwickelt werden sollen, müssen grundlegende Ruby-Kenntnisse vorhanden sein. Ruby ist eine voll objektorientierte Programmiersprache. Auf die Sprache Ruby wird an dieser Stelle nicht näher eingegangen, sondern es sei auf die Ruby-Webseite http://www.ruby-lang.org/en/ und auf das sehr detaillierte E-Book http://www.rubycentral.com/book/index.html verwiesen, dass sich sowohl für den Einsteiger als auch für Fortgeschrittene eignet. Den Scriptwriter's Guide, welcher die offizielle Dokumentation zur Erstellung eigener Skripte darstellt, befindet sich auf der offiziellen Homepage http://www.scriptroute.org.

Scriptroute-Modul (top-level-environment)

Sämtliche Scriptroute-spezifischen Klassen sind in dem Modul (bzw. Namespace) Scriptroute definiert. Daher werden alle Klassen so referenziert: "Scriptroute::KlasseXYZ". Das Scriptroute-Modul implementiert ein paar Methoden zur Konfiguration des Scriptroute-Daemons und Versionsabfrage des Interpreters. Es handelt sich um die folgenden Methoden:

Name der Methode bzw. des Structs	**Beschreibung**
Scriptroute.DaemonVersion	Liefert eine Struktur mit Versionsinformationen über den installierten Scriptroute-Daemon.
Scriptroute.DaemonConfig	Liest das installierte Daemon-Configuration-File als Hashtable aus. Alle

	gelesenen Daten werden in den passenden Datentyp konvertiert und nicht alles als String interpretiert.
Scriptroute::InterpreterVersion	Ein konstantes Struct mit Haupt und Unterversionsnummer des installierten Interpreters (z.B. 0.2).

Scriptroute-spezifische Klassen

Um individuelle Skripte zu entwickeln, werden die zuvor erwähnten Paketklassen verwendet. Der Anwender hat die Möglichkeit IP-Pakete, ICMP-Pakete, UDP-Pakete und natürlich TCP-Pakete mit Hilfe der jeweiligen Klassen zu erstellen. Es können nun die Header der jeweiligen Protokolle individuell über die Member-Variablen der Paketklassen konfiguriert werden. Die Basisklasse sämtlicher Scriptroute-Klassen ist Object, d.h. jede Klasse erbt alle Methoden von Object (so wie in Java). Die Klasse Scriptroute::IP ist Basisklasse aller anderen Paketklassen. Siehe das Klassendiagramm auf der nächsten Seite.

Die folgende Abbildung zeigt die Klassenstruktur der Paketklassen als UML-Klassendiagramm mit den wichtigsten Elementen der Protokollheader als Attribute. Sämtliche Attribute besitzen Default-Werte, sodass gezielt einzelne Werte verändert werden können. Weiters werden Attribute, die nicht vom User gesetzt wurden automatisch vom Scriptroute-Daemon gesetzt.

Alle Werte, sowie die Konstanten für die Scriptroute::ICMP-Klasse finden sich in der Original-Dokumentation (im Scriptwriter's Guide) auf der Webseite http://www.cs.washington.edu/research/networking/scriptroute/papers/scriptwriting.pdf.

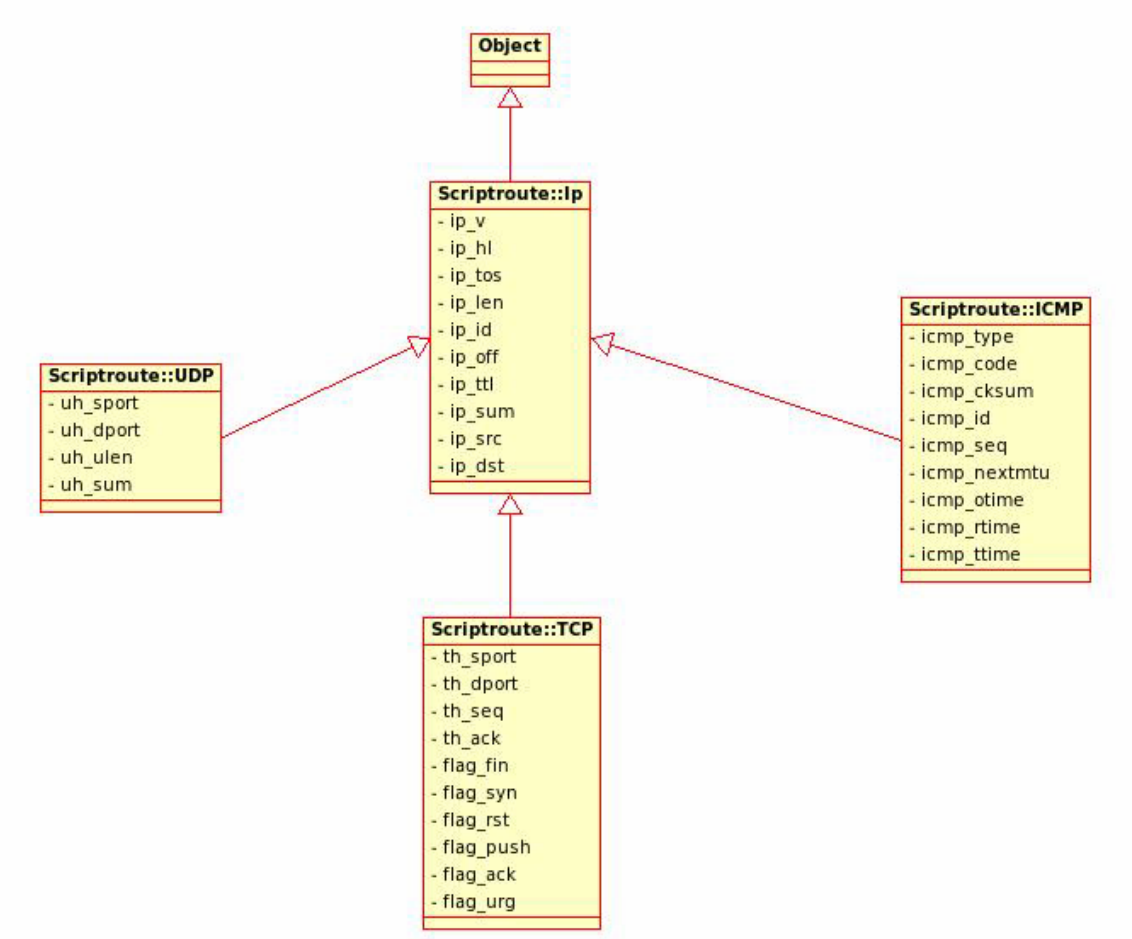

Abbildung 11: Klassenstruktur der Paketklassen

Kapitel 5

Abschlussbetrachtung

5.1. Resümee

In diesem Abschnitt werden abschliessend die Vor- und Nachteile von DummyNet erläutert. Im Fall von PlanetLab und Internet 2 können nur theoretische Aussagen gemacht werden, da weder Internet 2 noch PlanetLab praktisch ausprobiert werden konnte. Weiters soll in Kapitel 5.1.4 eine Entscheidungshilfe mit Hilfe einiger Entscheidungsvariablen gegeben werden, damit der Leser weiss, wann er welche Testumgebung verwenden kann.

5.1.1. Vor- und Nachteile von Dummy Net

Vorteile von DummyNet

Ein Vorteil von DummyNet ist die Kompaktheit der Testumgebung. DummyNet ist ja ein Feature der IPFW-Firewall und diese ist fixer Bestandteil eines jeden FreeBSD-Basissystems. Es braucht nur DummyNet in der Kernelkonfigurationsdatei aktiviert werden und der Kernel neu kompiliert und installiert werden. Weiters ist die Installation von FreeBSD für den Einsatz von DummyNet auch für Anfänger kein grosses Problem, da mit sysinstall ein graphisches Installationstool zur Verfügung steht. Ein weiterer Vorteil ist, dass mit PicoBSD ein FreeBSD-System mit DummyNet-Funktionalität zur

Verfügung steht, welches auf eine einzige bootfähige Floppy-Diskette passt. D.h. soll DummyNet vor einer vollständigen Installation zuerst einmal ausprobiert werden, so wird nur diese Diskette benötigt. Ein weiterer Vorteil ist, dass mit DummyNet relativ schnell Ergebnisse erzielt werden können. Sollen beispielsweise alle ausgehenden UDP-Pakete um einen fixen Wert verzögert werden, so kann dies mit zwei IPFW-Befehlen erfolgen: ein Befehl für die Erstellung der Pipe und ein Befehl für die Konfiguration der Pipe. Aufgrund der Tatsache, dass DummyNet fix im Kernel integriert ist, ergibt sich praktisch kein Overhead bei der Verwendung von DummyNet. Deshalb wird DummyNet häufig für Bandwidth-Management auf Routern verwendet. Es ist z.B. möglich mit wenigen Befehlen die Bandbreite auf einem Router für die jeweiligen Subnetze zu regulieren.

Nachteile von DummyNet

Obwohl die Installation von FreeBSD kein ernstes Problem darstellt, ist es doch ein Nachteil, dass DummyNet nur für FreeBSD verfügbar ist. Es wurde zwar im Internet nach Linux-kompatiblen Implementierungen von DummyNet gesucht; jedoch wurden keine gefunden. Der User muss sich nicht nur in die Testumgebung einarbeiten, sondern eben auch in ein neues Betriebssystem. Dies ist z.B. beim Einsatz von NistNet oder vom NS-Emulator nicht der Fall, da beide für Linux verfügbar sind. Ein weiterer Nachteil ist, dass im Zuge der Installation von DummyNet keinerlei Beispielskripte installiert werden (wie etwa beim NS-Emulator). Die Beispiele müssen aus der Manpage oder dem Internet stückweise zusammengesucht werden. Es gibt zwar recht umfangreiche Web-Seiten (siehe Literaturverzeichnis) mit Dokumentation zur prinzipiellen Funktionsweise von DummyNet, aber eigentlich keine komplexeren Beispiele.

5.1.2. Zusammenfassung zu Internet 2

Zuerst muss einmal klargestellt werden, dass Internet 2 keine klassische Netzwerktestumgebung ist, wie etwa NistNet oder Emulab. Die eben erwähnten Testumgebungen können “nur“ Netzwerktraffic in irgendeiner Weise simulieren bzw. manipulieren. Internet 2 ist aber eine Einrichtung, die sich zum Ziel gesetzt hat, an Netzwerktechnologien zu arbeiten, die den Nachfolger des heutigen Internet bestimmen werden. Die Arbeit von Internet 2 umfasst mit Working Groups für Voice Over IP, Multicast, IPv6, Digital Video usw. viele eigenständige Gebiete. Jedoch gibt es bei

Internet 2 schon eine Initiative, die sich mit Netzwerktests beschäftigt, nämlich die E2Epi (End-to-End-Performance-Initiative) mit Hilfe des E2E piPEs Frameworks. Diese Netzwerktests dienen aber nur der Verbesserung der Performance und dem Auffinden von Performanceproblemen innerhalb vom Abilene-Network für die Abilene-Nutzer. Sie sind also eher von administrativer Art. Es gibt in Internet 2 keine Möglichkeit, Netzwerktests im Stile von z.B. Emulab zu machen. Weiters ist noch wichtig, dass zum Zeitpunkt der Verfassung dieser Arbeit nur Institutionen innerhalb der USA Mitglied von Internet 2 werden können. Soweit festgestellt werden konnte, gibt es beispielsweise für Mitarbeiter einer österreichischen Universität keine Möglichkeit, sich aktiv (abgesehen von Mailing-Lists) an Projekten einer Working Group (Voice Over IP, IPv6 usw.) zu beteiligen.

5.1.3. Zusammenfassung zu PlanetLab

Gleich wie bei Internet 2 waren die Arbeiten zum Thema PlanetLab rein theoretischer Natur. Ein Beitritt zu PlanetLab war aufgrund der hohen Hardwareanforderungen (v.a. mind. 1 GByte RAM) für die PlanetLab Nodes derzeit nicht möglich. PlanetLab fordert, dass jedes Mitglied mindestens 2 Rechner zur Verfügung stellt, welche exklusiv für PlanetLab verwendet werden. Weiters müssen sich diese Rechner ausserhalb der hauseigenen Firewall befinden. Die detaillierten Anforderungen finden sich in Kapitel 4.1.3, “Wie werde ich Mitglied in PlanetLab?“. Aufgrund der ausführlichen Dokumentation war es aber möglich, trotz der mangelnden Mitgliedschaft, PlanetLab etwas genauer zu studieren. Lediglich der Zutritt zum Member-Bereich blieb klarerweise versagt. Es konnten keine Slices erstellt oder die Sensors ausprobiert werden. Es gibt zwar in Emulab die Möglichkeit, einen PlanetLab-Slice zu erstellen, jedoch stand diese Funktion aufgrund von API-Problemen (“Planetlab support is currently broken due to API incompatibilities introduced into PLC“) während der Arbeiten leider nicht zur Verfügung.

Wie in Kapitel 4.1.7, “Services, Anwendungen in PlanetLab“ beschrieben, gibt es ähnlich zu Internet 2 viele Einsatzgebiete von PlanetLab. PlanetLab ist keine reine Netzwerktestumgebung wie etwa Emulab. Im Bereich network-measurement ist Scriptroute die erste Wahl. Wie im Kapitel 4.4, “Arbeiten mit Planetlab – Netzwerktests mit Scriptroute“ erläutert, basiert Scriptroute auf Ruby-Skripten, wobei mit den Paketklassen von Scriptroute die Header von Netzwerkprotokollen (IP, TCP, UDP,

ICMP) manipuliert werden können. Hier muss aber noch erwähnt werden, dass nicht alle Header-Attribute gesetzt werden können. Die Checksummen werden beispielsweise stets vom Scriptroute-Daemon gesetzt. Jedoch besitzt der Entwickler insgesamt eine ähnliche Flexibilität wie bei der herkömmlichen Socketprogrammierung in C.

Es gibt enge Zusammenarbeiten zwischen Internet 2 und PlanetLab. Zahlreiche PlanetLab-Nodes sind Teil des Abilene-Networks; auf der anderen Seite bieten acht der Core-Router-Nodes vom Abilene-Network PlanetLab-Funktionalität.

5.1.4. Welche Testumgebung soll ich verwenden?

Diese Frage läßt sich nicht allgemein beantworten, da unterschieden werden muss, welche Ansprüche bzw. Forderungen an eine Testumgebung gestellt werden. Die Entscheidung, welche Testumgebung verwendet wird, liegt im Endeffekt beim User.

Aufgrund der Tatsache, dass nur DummyNet praktisch getestet werden konnte und DummyNet eigentlich ziemlich gut mit den Testumgebungen der ersten Arbeit vergleichbar ist, werden hauptsächlich die gleichen Entscheidungsvariablen verwendet; dadurch bekommt der Leser einen Vergleich zwischen den Testumgebungen der ersten Arbeit und DummyNet. Abbildung 12 zeigt die überarbeitete Tabelle mit den Entscheidungsvariablen.

	Programmierkenntnisse erforderlich (Tcl-Kenntnisse, IPFW-Firewall)	eigenes Netzwerk erforderlich	Installation erforderlich	Anmeldung erforderlich	Installation von alternativem Betriebssystem (FreeBSD)	Möglichkeit von virtuellen Knoten	grafische Visualisierung der Ergebnisse	Geringer Overhead durch Einsatz der Testumgebung	Mailingliste zur Hilfe vorhanden	Umfangreiche Dokumentation
emulab				X		X			X	X
ns-e	X	X	X			X	X		X	X
NistNet	X	X	X					X	X	
DummyNet	X	X	X		X			X	X	X

Abbildung 12: Zusammenfassung wichtiger Entscheidungsvariablen

Aus der Tabelle geht hervor, dass DummyNet eine interessante Alternative zu NistNet darstellt. Ein Vorteil von DummyNet zu NistNet liegt darin, dass DummyNet auch auf Rechnern mit nur einer Netzwerkkarte eingesetzt werden kann. Weiters ist eine umfangreichere Dokumentation vorhanden, was die Einarbeitungszeit in die Testumgebung verkürzt. Demgegenüber steht natürlich der grosse Nachteil, dass DummyNet nur für FreeBSD verfügbar ist, NistNet jedoch für Linux. Weiters gibt es auch keine graphische Oberfläche für DummyNet. Dies kann für manche User ein Nachteil sein und ist im Endeffekt vom User abhängig. Manche Anwender arbeiten lieber über eine Shell, andere bevorzugen graphische Oberflächen. Während der Arbeiten an DummyNet ist es jedoch nicht als unangenehme Tatsache aufgefallen, dass DummyNet Kommandozeilenorientiert arbeitet.

Weiters ist noch zu sagen, dass DummyNet nicht den Komfort von Emulab (z.B. schnelles Zusammenklicken von Szenarios mit graphischem Editor) bietet. Für einen Vergleich der Testumgebungen emulab, ns-e und NistNet sei auf die erste Arbeit zum Thema “Evaluierung von Netzwerktestumgebungen“ verwiesen.

5.2. Probleme und Besonderheiten

5.2.1. Dummy Net

Die erste Hürde bzw. Besonderheit im Zusammenhang mit DummyNet ist das Aktivieren der DummyNet-Funktionalität (genau beschrieben im Kapitel 2.2, “Installation“). DummyNet ist nach der Installation von FreeBSD standardmässig deaktiviert. Werden die Schritte jedoch wie beschrieben durchgeführt, sollte das kein Problem darstellen. Das Kompilieren und Installieren eines Kernels klingt zwar kompliziert und mag den einen oder anderen abschrecken, ist aber im Endeffekt kein grosses Problem (siehe Anweisungen in Kapitel 2.2.).

Die wichtigste Besonderheit im Zusammenhang mit der IPFW-Firewall ist der Umstand, dass standardmässig jeglicher Traffic blockiert wird; dies ist natürlich für die Arbeit mit DummyNet nicht wünschenswert. Der Anwender muss also, wie in Kapitel 2.1.2, “Arbeitsweise von DummyNet“ beschrieben, explizit mit einer IPFW-Regel den Traffic für das gewünschte Protokoll (ICMP, IP, TCP, UDP) freigeben. Weiters ist noch wichtig, dass in der Regel immer ein Full-Duplex-Link simuliert werden soll; also Download und Upload. Hierfür müssen in DummyNet immer zwei getrennte Pipes definiert werden: eine Pipe für den Download und eine Pipe für den Upload.

5.2.2. Internet 2

Das Hauptproblem bei Internet 2 bestand darin herauszufinden, ob Internet 2 überhaupt eine geeignete Netzwerktestumgebung darstellt. Es gibt zwar eine ausführliche Dokumentation über die verschiedenen Working Groups usw. auf der offiziellen Web-Site, jedoch finden sich nirgends konkrete Informationen die auf eine Verwendung als Netzwerktestumgebung schliessen lassen. Nach langem Suchen wurde festgestellt, dass Internet 2 sich nicht als Netzwerktestumgebung eignet. Falls dies doch der Fall sein sollte und übersehen wurde, so würde Internet 2 an der Universität Innsbruck nicht genützt werden können, da die Mitgliedschaft nur für Institutionen innerhalb der USA möglich ist.

5.2.3. PlanetLab

Im Gegensatz zu Internet 2 war bei PlanetLab gleich klar, dass es mit Hilfe von Scriptroute als Netzwerktestumgebung verwendet werden kann. Aufgrund der vorher erwähnten Gründe (Hardwareanforderungen) wurde PlanetLab nur theoretisch untersucht. Das größte Problem bestand in der dürftigen Dokumentation über die Verwendung von Scriptroute im Zusammenhang mit PlanetLab. Dokumentation über die Installation unter PlanetLab ist nicht vorhanden bzw. konnte nicht gefunden werden. Es gibt nur das Installationsskript zum Download (siehe Kapitel 4.4.5, “Installation von Scriptroute unter PlanetLab“). Falls Scriptroute konkret mit PlanetLab verwendet werden soll, treten aufgrund der dürftigen Dokumentation sicher Schwierigkeiten im Zuge der Installation auf. Die einzige Hilfe sind dann die Mailing Lists von PlanetLab (siehe Kapitel 4.1.5, “Kontakt zu PlanetLab“) oder die eigene Mailing List von Scriptroute.

Literaturverzeichnis

DummyNet

[1] Offizielle DummyNet-Homepage
http://info.iet.unipi.it/~luigi/ip_dummynet/

[2] Tutorial für DummyNet in Englisch
http://cs.ecs.baylor.edu/~donahoo/tools/dummy/tutorial.htm

[3] Deutschsprachige Einführung in DummyNet
http://homepage.univie.ac.at/l.ertl/talks/trafficshaping/img0.html

[4] Offizielle FreeBSD-Homepage
http://www.freebsd.org/

[5] FreeBSD-Download
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/

[6] Offizielles FreeBSD-Handbuch mit Installationsanleitung in Englisch
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html

[7] Offizielles FreeBSD-Handbuch mit Installationsanleitung in Deutsch
http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/index.html

Internet 2

[1] Offizielle Internet 2-Homepage
http://www.internet2.edu/

[2] Übersicht über die Mitglieder von Internet 2
http://members.internet2.edu/university/universities.cfm

[3] Internationale Kooperationen von Internet 2
http://international.internet2.edu/partners/

[4] Mailing Lists von Internet 2
https://mail.internet2.edu/wws/
[5] Event-Kalender von Internet 2
http://events.internet2.edu/
[6] Übersicht über alle Working Groups
http://www.internet2.edu/working-groups.html
[7] Offizielle Abilene-Homepage
http://abilene.internet2.edu/
[8] Abilene Peer Networks
http://abilene.internet2.edu/peernetworks/

PlanetLab

[1] Offizielle PlanetLab-Homepage
http://www.planet-lab.org/
[2] Übersicht über die Mitglieder von PlanetLab
http://www.planet-lab.org/php/institutions.php
[3] Hardwareanforderungen an PlanetLab-Nodes
http://www.planet-lab.org/consortium/hardware.php
[4] Relevante Dokumente für den PlanetLab-Beitritt
http://www.planet-lab.org/consortium/
[5] Übersicht über die PlanetLab Mailing Lists und Working Groups
http://www.planet-lab.org/php/workinggroups.php
[6] PlanetLab-Software Download
http://www.planet-lab.org/Software/download.php
[7] Configuration File für PlanetLab-Nodes
http://www.planet-lab.org/consortium/bootcdconfig.php
[8] User-Account Erstellung
https://www.planet-lab.org/db/accounts/register.php
[9] Übersicht über alle aktuellen Slices
https://www.planet-lab.org/php/slices.php
[10] Offizielle Safe-Raw-Sockets Dokumentation
http://www.planet-lab.org/raw_sockets/index-one.html

[11] Offizielle Scriptroute-Homepage
http://www.scriptroute.org
[12] Aktuell verfügbare Scriptroute-Server
http://www.scriptroute.org:3967/

Sonstiges

[1] DFN-Verein (Deutsches Forschungsnetz)
http://www.dfn.de/content/de/
[2] DANTE (Delivery of Advanced Network Technology to Europe)
http://www.dante.org.uk/
[3] ACOnet-Homepage
http://www.aco.net/
[4] GEANT Network
http://www.geant.net/
[5] YUM-Homepage
http://linux.duke.edu/projects/yum/
[6] Offizielle Ruby-Homepage
http://www.ruby-lang.org/en/
[7] Detailliertes Ruby E-Book
http://www.rubycentral.com/book/index.html