UNIVERSITÄT
KOBLENZ · LANDAU

Studienarbeit

Nagios unter VNUML

Abgabe: 26.1.2011

Bearbeiter: Torsten Römer

Matrikelnummer: 1009520026

Betreuung: Prof. Dr. Christoph Steigner
Dipl. Inf. Harald Dickel
Dipl. Inf. Frank Bohdanowicz

Inhaltsverzeichnis

Vorwort **4**

1 Einführung in Nagios **5**

1.1 Historie **5**

1.2 Systemarchitektur **6**

1.2.1 Plugins **7**

1.2.2 Weboberfläche **9**

1.2.3 Performancedaten **11**

1.2.4 Konfiguration **12**

1.2.5 Der Nagios-Kern **16**

1.2.5.1 Host-Tests und Service-Tests **16**

1.2.5.2 Harte und weiche Zustände **17**

1.2.5.3 Nutzung von Topologieinformationen **17**

1.2.6 Benachrichtigungen **18**

1.2.6.1 Nachrichtenfilter **19**

1.2.6.2 Beispiel **20**

2 Installation von Nagios **22**

2.1 Installation von Software unter VNUML **22**

2.2 Vorbereiten der Installation **25**

2.3 Installation **27**

2.4 Testen der Installation **30**

3 Die Nagios-Konfiguration **34**

3.1 VNUML-Szenario **34**

3.2 Hostobjekte **35**

3.3 Zeitfensterobjekte **39**

3.4 Kommandoobjekte **39**

3.5 Testen der Konfiguration **41**

3.6 Hostextinfo-Objekte **42**

3.7 Serviceobjekte **45**

4 Überwachung von Netzwerkdiensten **47**
4.1 Hypertext Transfer Protocol(HTTP) **47**
4.1.1 Installation/Konfiguration **48**
4.1.2 Überwachung **49**
4.2 Secure Shell Protocol(SSH) **51**
4.2.1 Installation/Konfiguration **51**
4.2.2 Überwachung **51**
4.2.3 Überwachung lokaler Ressourcen **52**
4.3 File Transfer Protocol(FTP) **56**
4.3.1 Installation/Überwachung **57**
4.3.2 Überwachung **59**
4.4 Domain Name System(DNS) **60**
4.4.1 Installation/Konfiguration **61**
4.4.2 Überwachung **66**
4.5 MySQL **67**
4.5.1 Installation/Konfiguration **67**
4.5.2 Überwachung **70**
4.6 Mailserver **71**
4.6.1 Installation/Konfiguration **72**
4.6.2 Überwachung **76**

Fazit **81**

Quellenangaben **82**

Vorwort

Die Virtualisierungssoftware *VNUML* wird an der Universität Koblenz-Landau eingesetzt, um Studierenden des Fachbereichs Informatik einen praxisorientierten Umgang mit verschiedenen Netzwerkprotokollen zu ermöglichen, was mit einem einzelnen Rechner sonst nur schwer zu bewerkstelligen ist. Gegenstand dieser Arbeit ist der Einsatz der Monitoringsoftware *Nagios* in einem mit VNUML erstellten virtuellen Netzwerks, ein Schwerpunkt liegt dabei auf der Überwachung von verschiedenen Netzwerkdiensten durch Nagios. Der Titel mag etwas irreführen: Da sich der Betrieb von Nagios im virtuellen Netz nicht wirklich von dem im realen Netz unterscheidet, ist im Endergebnis eher eine allgemeingültige Nagios-Anleitung (für Nagios-Einsteiger) herausgekommen.

Zu den einzelnen Kapiteln: In Kapitel 1 wird ein allgemeiner Überblick über den Aufbau eines Nagios-Systems und seiner einzelnen Komponenten geboten. Kapitel 2 ist im wesentlichen eine Installationsanleitung. Im dritten Kapitel wird die Konfiguration von Nagios beschrieben; im vierten und letzten Teil folgt schließlich eine Betrachtung der für die verschiedenen Netzwerkdienste zuständigen Bestandteile des Systems, wobei auch einige Hinweise auf Installation und Inbetriebnahme dieser Dienste im VNUML-Netz gegeben werden.

Auf der beiliegenden DVD befinden sich zwei UML-Imagedateien. Die eine enthält alle in dieser Arbeit vorgestellten Installationen einschließlich der zugehörigen Konfigurationsdateien und Nagios-Objektdefinitionen, eine Anleitung zur Inbetriebnahme der installierten Programme ist auf der DVD enthalten. Auf dem zweiten Image sind lediglich das Nagios-Hauptprogramm und die zugehörigen Plugins installiert. Der zu den Images passende UML-Kernel ist ebenfalls vorhanden.

1. Einführung in Nagios

1.1 Historie

Nagios ist ein Open-Source-Werkzeug zur Überwachung von IT-Systemen und Netzwerken. Als Betriebssystem wird ein Unix-Derivat benötigt, die Systeme im zu überwachenden Netz sind hingegen beliebig. Nagios wurde ab 1999 von Ethan Galstad, wohnhaft in Saint Paul, Minnesota, zunächst unter dem Namen *NetSaint* entwickelt. Aus rechtlichen Gründen wurde das Projekt 2001 in Nagios umbenannt. Der Name Nagios ist eines der in der IT-Welt beliebten rekursiven Akronyme und steht für *Nagios ain´t gonna insist on sainthood*, zu deutsch sinngemäß „Nagios besteht nicht auf seiner Heiligkeit“, eine Anspielung auf einen möglichen Rechtsstreit, dem Galstad mit der freiwilligen Namensänderung aus dem Weg gegangen ist. Zugleich ist es eine Komposition aus den Worten Network und Hagios, dabei ist letzteres die griechische Übersetzung des englischen Saint.

Motivation für die Entwicklung von Nagios bzw. NetSaint war die Absicht Galstads, Netzwerkmonitoring-Dienste für kleinere Unternehmen anzubieten, wobei er kommerziell vertriebene Produkte als zu teuer und damals frei verfügbare als für seine Zwecke unzureichend erachtete. Einer seiner Hauptkritikpunkte war dabei die fehlende oder zumindest unbefriedigende Weboberfläche derartiger Werkzeuge.[1]

Durch die (ursprünglich nicht geplante) Veröffentlichung seiner Arbeit als Open-Source erfuhr Nagios eine rasche Verbreitung - die zunehmende Nachfrage an Support veranlasste Galstad Ende 2007 zur Gründung der *Nagios Enterprises LLC*[2], welche diesen als kommerzielle Dienstleistung anbietet.

1 vgl. (Galstad 2005)
2 `http://www.nagios.com/`

1.2 Systemarchitektur

Die Aufgabe eines Netzwerkmonitoring-Tools ist es, Informationen über ein IT-Netzwerk und dessen zugehörige Komponenten zu sammeln, um diese einerseits in einer für den Anwender übersichtlichen Form zusammenzustellen, als andererseits aktiv über Fehlerzustände zu informieren. Die bei der Verfolgung dieser Ziele beteiligen Komponenten des Nagios-Systems zeigt Abb.1.

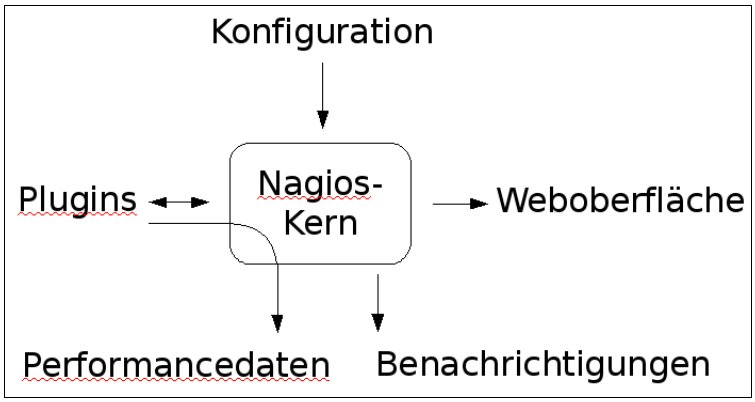

Abb.1: Nagios-Systemkomponenten

Ein Vorteil von Nagios gegenüber einigen konkurrierenden Produkten besteht in seinem modularen Aufbau, der zu einer größeren Flexibilität führt, was mögliche Einsatzbereiche betrifft: Über Konfigurationsdateien steuert der Benutzer das Verhalten des Systems. Gemäß der Einstellungen dort erfolgt das Sammeln der Informationen, welches bei Nagios die Aufgabe der sogenannten *Plugins* ist. Diese Informationen werden dem Benutzer in einer *Weboberfläche* ansprechend präsentiert, über wichtige Zustandsänderungen wird der Benutzer zudem vom *Nagios-Benachrichtigungssystem* unterrichtet, z.B. per E-Mail. *Performancedaten* schließlich sind detailliertere Informationen, die für die (optionale) Verarbeitung durch externe Programme bestimmt sind.[3]

Die Funktionen der einzelnen Bestandteile des Nagios-Systems sollen in diesem Abschnitt skizziert und anhand eines einfachen Beispiels verdeutlicht werden.

3 Performancedaten sind also weniger ein Teil des Nagios-Systems, vielmehr handelt es sich um reine Ausgaben.

1.2.1 Plugins

Das Sammeln der Informationen, die eigentlichen Tests (z.B. Erreichbarkeit eines Rechners, Verfügbarkeit eines Dienstes etc.) werden von kleinen Hilfsprogrammen, die als Plugins bezeichnet werden, durchgeführt. Der Nagios-Kern ruft im Bedarfsfall die einzelnen Plugins auf, die ihre Ergebnisse (über eine für alle Plugins identische Schnittstelle) dem Nagios-Kern zur Verfügung stellen. Dadurch bleibt das Hauptprogramm von den Details einzelner Tests abgeschirmt. Nagios verwendet dabei für die Beschreibung der Zustände zu überwachender Rechner, Dienste etc. ein sogenanntes *Ampelsystem*, d.h. der Zustand wird in der Antwort des Plugins durch einen der Werte *rot, gelb, grün* bzw. im „Nagios-Dialekt“ *CRITICAL, WARNING, OK* beschrieben.[4]

Zur Verdeutlichung des oben gesagten ein Beispiel: Der Rechner mit der Adresse `10.0.3.1` soll auf Erreichbarkeit geprüft werden. Dazu soll das Plugin `check_icmp` verwendet werden, welches ICMP-Pakete[5] generiert und an den zu überprüfenden Rechner sendet. Der gewünschte Rückgabewert des Plugins soll sein:

- *CRITICAL(rot)*, wenn der Rechner nicht erreichbar ist[6]
- *WARNING(gelb)*, wenn der Rechner zwar erreichbar ist, aber die Antwortzeit 200ms überschreitet oder mehr als 20% der Pakete nicht beantwortet werden
- *OK(grün)*, sonst

Der entsprechende Aufruf von `check_icmp` sieht dann so aus:[7]

```
check_icmp -H 10.0.3.1 -w 201,21% -c 10000,100%
```

Über die Option `-H (--hostname)` erhält `check_icmp` die Adresse des zu überprüfenden Rechners, über `-w (--warning)` und `-c (--critical)` die

4 An dieser Stelle sei noch darauf hingewiesen, dass Nagios tatsächlich noch einen vierten Zustand, "`UNKNOWN`"(orange), kennt. Dieser zeigt meist eine unsachgemäße Bedienung eines Plugins (z.B. Verwendung einer nicht unterstützten Option, Angabe unzulässiger Werte für einzelne Parameter) an.

5 `check_icmp` versendet ICMP Echo Request-Pakete, wie das Programm `ping`

6 Als Timeout werden hier zehn Sekunden festgesetzt

7 Die Nagios-Plugins sind eigenständige Programme und können einzeln über die Kommandozeile aufgerufen und getestet werden.

Schwellwerte, ab denen es eine Warnung bzw. einen kritischen Zustand anzeigen soll. Durch die Übergabe von Parametern an die Plugins wird also jeweils das gewünschte Verhalten erreicht. Nagios modifiziert im laufenden Betrieb diese Werte übrigens nicht, sie werden vielmehr unverändert aus vom Administrator angelegten Definitionen[8] in den Plugin-Aufruf kopiert. Ist der angesprochene Rechner gemäß der oben angegebenen Vereinbarungen uneingeschränkt erreichbar, könnte eine mögliche Ausgabezeile zum obigen Befehl wie folgt aussehen:

```
OK – 10.0.3.1: rta 3.340ms, lost 0%      →[9]
|rta=3.340ms;201.000;10000.000;0; pl=0%;21;100;;
```

Die Informationen, welche auf das „|“ folgen, sind vom Plugin bereitgestellte Performancedaten(siehe 1.2.3). Die Informationen ab dem „OK“ bis zum „|“ werden als *Serviceoutput* bezeichnet. Sie werden von Nagios in der Weboberfläche wiedergegeben, um dem Benutzer das Eingrenzen etwaiger Fehlerquellen schneller zu ermöglichen. Für das weitere Verhalten des Nagios-Systems ist allein der zurückgegebene Zustand, hier `OK`, maßgeblich.[10] Nagios reagiert dabei auf Zustandsänderungen mit Nachrichten an den Netzwerkadministrator und/oder, wenn ein ernster Fehler zu befürchten ist, weitere zuständige Personen.[11] Solche Nachrichten werden in der Praxis meist E-Mails sein, aber auch Benachrichtigungen per SMS oder das Ansteuern von Warnleuchten etc. sind unter Zuhilfename externer Programme (zumindest theoretisch) möglich.

Pakete mit einer ganzen Reihe von *Standard-Plugins* sind frei erhältlich (z.B. auf der Nagios-Hauptseite `http://www.nagios.org`) Diese umfassen Erreichbarkeitstests, Tests für das Überprüfen lokaler Ressourcen wie Festplattenplatz, Systemlast, Systemzeit bis hin zu Informationen über einzelne Dateien oder Hardware-Checks mit LM[12]-Sensoren etc. Weiterhin Port-Tests und Tests für Netzwerkdienste wie DNS, FTP, SMTP etc. und vieles

8 In den Nagios-Konfigurationsdateien, siehe Abschnitt 1.2.3

9 Durch einen abschließenden Pfeil soll hier und im folgenden ein Zeilenumbruch kenntlich gemacht werden.

10 Nagios extrahiert den Zustand nicht aus der vom Plugin generierten Textausgabe, sondern orientiert sich an zurückgegebenen Fehlercodes. (Diese sind 0(`OK`), 1(`WARNING`), 2(`CRITICAL`) und 3(`UNKNOWN`).)

11 Präzisere Angaben sind hier nicht möglich, da das Verhalten über die Konfiguration verändert werden kann.

12 linux-monitoring

andere.[13] Damit sind die Einsatzmöglichkeiten von Nagios jedoch noch nicht erschöpft. So gibt es Austauschplattformen für Nagios-Plugins im Internet[14] sowie die Möglichkeit, eigene Plugins zu schreiben. Die Einfachheit der Schnittstellen zum Nagios-Kern (Aufruf über die Kommandozeile, einfache Zeichenkette als Rückgabewert) bringen es mit sich, dass man ein Nagios-Plugin in jeder beliebigen Programmiersprache erstellen kann. Durch diese Flexibilität kann alles, was sich elektronisch messen lässt, mit Nagios überwacht werden, wenn es nur sinnvoll möglich ist, die gemessenen Werte in das Ampelsystem von Nagios umzusetzen. Beispiele sind die Überwachung des Serverraumes mit Temperatur-, Feuchtigkeits-, Rauch- oder Bewegungssensoren, schließlich sind aber auch Anwendungen, die in keiner direkten Beziehung zum IT-Bereich stehen, denkbar.

In dieser Arbeit sind ausschließlich die oben genannten Standard-Plugins von Bedeutung. Wie diese in das Nagios-System eingebunden werden, wird in den Kapiteln 3 und 4 anhand mehrerer Beispiele detailliert beschrieben.

1.2.2 Weboberfläche

Um dem Benutzer die Informationen über den Zustand des Netzwerkes in übersichtlicher Form anbieten zu können, kann auf eine grafische Oberfläche nicht verzichtet werden. Damit aber die Informationen von verschiedenen Standorten aus zugänglich sind, erfolgt die Darstellung der Informationen bei Nagios über eine Weboberfläche. An der jeweiligen Arbeitsstation wird so lediglich ein Browser benötigt, d.h. es muss dort keine zusätzliche Software installiert werden.

Die Weboberfläche bietet verschiedene Sichten auf die von den Plugins gesammelten Daten. Sie zeigen unterschiedliche Ausschnitte der Informationen in entsprechend unterschiedlicher Detailliertheit. Aufgrund ihrer Vielzahl kann eine Besprechung der einzelnen Punkte im Rahmen dieser Arbeit nicht vorgenommen werden, was wegen der einfachen und intuitiven Bedienung der Weboberfläche jedoch nicht stört. Stattdessen sollen hin und

13 Bei einigen Diensten hat der Benutzer sogar die Auswahl zwischen mehreren verfügbaren Plugins.
14 z.B. `http://exchange.nagios.org/`

wieder Beispiele einen Eindruck vermitteln.

Abb.2 zeigt oben einen Screenshot der Nagios-Weboberfläche, am linken Bildrand sind unter dem Stichwort Monitoring verschiedene Sichten auf die von den Plugins gesammelten Daten zu erreichen, im Beispiel ist der Punkt *Host Detail* ausgewählt. Da gegenwärtig alle Rechner des Netzes erreichbar sind, erscheinen sämtliche Zeilen rechts im Bild dem Zustand OK entsprechend der zugehörigen Ampelfarbe grün unterlegt. Am unteren Ende der Abbildung befindet sich eine vergrößerte Kopie der fünften Zeile, welche dem oben betrachteten Erreichbarkeitstest entspricht[15], am rechten Ende der Zeile erkennt man die Ausgabe des Serviceoutputs des benutzten Plugins `check_icmp`. Wie ein Blick auf die Adressleiste des Browsers verrät, erreicht man die Nagios-Weboberfläche unter der Adresse des Nagios-Servers, im Beispiel `10.0.5.1`, gefolgt von `/nagios`.

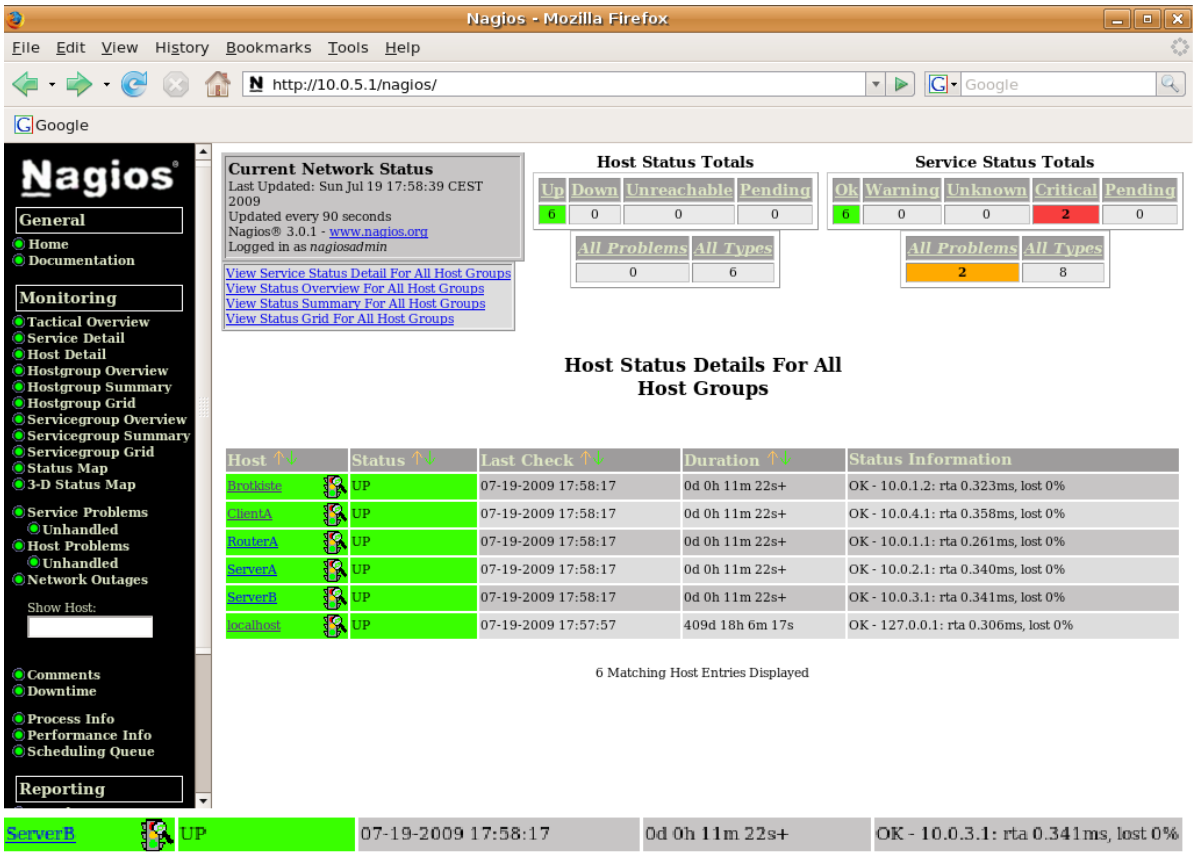

Abb.2 Weboberfläche von Nagios

15 vgl Abschnitt 3.1, Abb.8: `ServerB` ist im betrachteten Beispielnetz die Adresse `10.0.3.1` zugewiesen.

1.2.3 Performancedaten

Die Ausgabe eines Großteils der verfügbaren Plugins enthält neben Ampelzustand und Serviceoutput (Daten zur Ausgabe in der Weboberfläche) sogenannte Performancedaten, die vom Nagios-Kern ignoriert werden und zur Weiterverarbeitung durch externe Programme bestimmt sind. Das System kann dazu so konfiguriert werden, dass die Performancedaten in einer Datei abgelegt werden, die dann einem anderen Programm als Eingabe dient. Dabei kann der Administrator dafür Sorge tragen, dass Nagios die Daten mit zusätzlichen Informationen versieht (Zeitstempel, betroffener Host etc.), um eine sinnvolle Weiterverarbeitung zu ermöglichen. Das Ziel ist hierbei meist eine grafische Darstellung dieser Daten. Dazu werden diese, nach Host und Dienst/Service getrennt und nach Zeitstempeln sortiert, durch das externe Programm zunächst in Round-Robin-Datenbanken gespeichert und anschließend über eine Erweiterung der Nagios-Weboberfläche grafisch dargestellt. Programme für die Verarbeitung der Performancedaten und andere Nagios-Erweiterungen können von der Nagios-Hauptseite[16] unter der Rubrik *Add-Ons* heruntergeladen werden, gegenwärtig sind dort bereits über zweihundert Programme frei verfügbar.

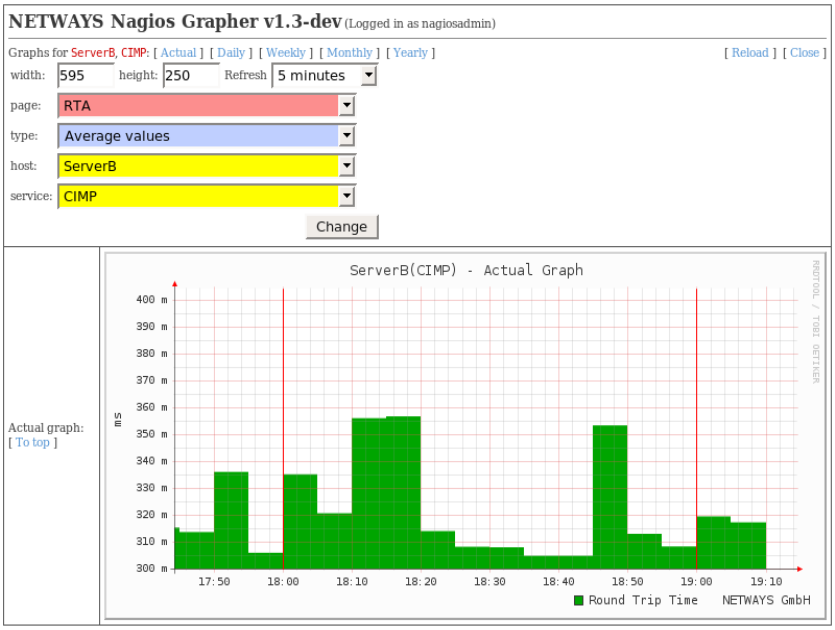

Abb.3 Beispiel für ein mit dem Nagios-Add-On *Nagiosgrapher* erstelltes Diagramm:
Round Trip Time der von check_icmp gesendeten Pakete an ServerB (IP 10.0.3.1) im Zeitraum der letzten 90 Minuten.

16 http://www.nagios.org/

1.2.4 Konfiguration

Über Einstellungen in den Konfigurationsdateien werden dem System Informationen bezüglich der Topologie des Netzwerks bekanntgemacht, und über die Konfiguration wird gesteuert, welche Netzwerkgeräte und welche Dienste auf denselben überwacht werden sollen. Auch wann, wie und wie oft diese Überprüfungen jeweils stattfinden, wird hier festgelegt. Die Konfiguration ist vom Benutzer selbst zu erstellen, Veränderungen des Netzwerkes müssen eingepflegt werden.

Die Konfiguration von Nagios erfolgt durch vom Administrator in der Konfigurationsdatei `nagios.cfg` angelegte Objektdefinitionen und vorgenommene Einstellungen. Nagios unterscheidet dabei folgende Typen von Objektdefinitionen (mit jeweils vereinfachter Funktionsbeschreibung):

- *host*: Definiert einen zu überwachenden Netzwerkknoten und den entsprechenden Erreichbarkeitstest
- *service*: Definiert einen Dienst auf einem Host und den entsprechenden Service-Test
- *contact*: Definiert eine Kontaktperson, die vom System Nachrichten erhält
- *hostgroup, servicegroup, contactgroup*: Fassen jeweils mehrere hosts, services bzw. contacts zu Gruppen zusammen
- *timeperiod*: Definiert ein Zeitfenster, z.B. Arbeitszeiten oder Wochentage etc.
- *command*: Definiert einen Puginaufruf über die Kommandozeile, etwa einen Erreichbarkeitstest
- *hostextinfo, serviceextinfo*: Externe Informationen über Hosts/Services, meist zur Darstellung in der Weboberfläche, z.B. Icons, Links oder Grafiken

Darüberhinaus gibt es noch folgende optionale, in dieser Arbeit nicht benutzten Objekttypen:

- *servicedependency, hostdependency*: Abhängigkeitsinformationen ermöglichen dem System, Fehlerquellen automatisch einzugrenzen

- *serviceescalation, hostescalation*: Für eine differenziertere Problembehandlung; basierend auf der Dauer, die ein Problem besteht, werden unterschiedliche Kontakte benachrichtigt

Der besseren Übersicht wegen ist es ratsam, die Objektdefinitionen auf mehrere Dateien zu verteilen (z.B. nach Typen getrennt), diese können dem System über entsprechende Einträge in `nagios.cfg` bekanntgemacht werden. So könnte man beispielsweise die Zeitfensterobjektdefinitionen in eine Datei `timeperiods.cfg` auslagern und diese durch Angabe eines `cfg_file`-Befehls in der Datei `nagios.cfg` wieder in die Konfiguration einbinden:

```
cfg_file=/etc/nagios/timeperiods.cfg
```
[17]

Da einige der genannten Typen eine große Anzahl an Parametern umfassen, die für viele Instanzen auf gleiche Werte gesetzt werden, gibt es zusätzlich die Möglichkeit, *Templates* zu definieren und deren Eigenschaften an einzelne Instanzen zu vererben, was ebenfalls zur Übersichtlichkeit der Konfiguration beiträgt.

Neben den Definitionen in `nagios.cfg` etc. ist es möglich, Änderungen an der Konfiguration über die Weboberfläche vorzunehmen. Diese werden jedoch nicht in den Konfigurationsdateien gespeichert, sind also nach einem Neustart von Nagios verloren. Zwar hat man die Möglichkeit, mit dem Setzen des Flags `retain_state_information` in `nagios.cfg` die vorgenommen Änderungen persistent zu machen, doch da diese an anderer Stelle gespeichert werden[18], stimmt die tatsächliche Konfiguration (auch nach einem Neustart) nicht mehr mit den Definitionen in `nagios.cfg` überein.

Neben `retain_state_information` und `cfg_file` gibt es weitere Parameter bzw. Direktiven, über die in `nagios.cfg` Einfluss auf das Verhalten des Systems genommen werden kann, diese sind jedoch zu zahlreich, um sie hier vollständig wiedergeben zu können.[19]

Nicht alle benötigten Definitionen müssen bei einer Nagios-Installation manuell erstellt werden, da eine rudimentäre Grundkonfiguration

17 Im Verzeichnis /etc/nagios befinden sich stets die Nagios-Konfigurationsdateien.

18 Diese Informationen werden in einer Extra-Datei gespeichert, deren Name durch den Parameter state_retention_file in der Konfiguration festgelegt wird.

19 Eine entsprechende Auflistung findet man in (Barth 2009, S436ff.)

„mitgeliefert“ wird, die bereits eine stattliche Zahl von Kommando- und Zeitfensterobjekten etc. enthält und meist ohne größere Änderungen übernommen werden kann; man ergänzt sie einfach um die eigenen Definitionen. Auch Host-Definition und Service-Definitionen für den Nagios-Rechner[20] sind bereits vorhanden, sodass ein erster Testlauf stattfinden kann, bevor man sich selbst an das Editieren der Konfigurationsdateien heranwagt und ohne dass Netzwerkkomponenten beteiligt sind.[21] Da das Erstellen einer vollständigen Konfiguration - insbesondere für größere Netze - trotzdem immer noch sehr aufwändig bleibt, gibt es mittlerweile eine Reihe verschiedener Werkzeuge, die das Anlegen und Pflegen einer Nagios-Konfiguration erleichtern sollen. Auch solche Hilfsprogramme kann man als Add-Ons von der Nagios-Hauptseite beziehen. Die im Rahmen der vorliegenden Arbeit an der von Nagios mitgebrachten Grundkonfiguration vorgenommenen Änderungen bzw. Erweiterungen erfolgten jedoch ausschließlich in Handarbeit.

Ist die Bedienung der Weboberfläche auch ohne weitere Kenntnisse ausführbar und kann auf eine Verwendung der Performancedaten grundsätzlich verzichtet werden, so kommt der Einsteiger bei der Konfiguration des Systems und der Verwendung der Plugins nicht ohne eine ausführlichere Anleitung aus. Diese Anleitung zum Erstellen einer Nagios-Konfiguration folgt in Kap. 3, indem dort die mitgelieferte Grundkonfiguration sukzessive erweitert wird, um sie an ein virtuelles VNUML-Beispielnetz anzupassen. Um dennoch einen ersten Eindruck zu vermitteln, sollen an dieser Stelle abschließend diejenigen Erweiterungen betrachtet werden, die benötigt werden, um dem System eine neue Netzwerkkomponente „bekanntzumachen“ und es zu veranlassen, regelmäßig einen Erreichbarkeitstest durchzuführen. Betrachtet wird wieder das Beispiel des Rechners mit der Adresse 10.0.3.1 und dem in 1.2.1 vorgestellten Erreichbarkeitstest mittels `check_icmp`. Den Rechner definiert man über ein Hostobjekt:

20 Gemeint ist der lokale Rechner, auf dem man die Nagios-Installation vornimmt.

21 Man kann Nagios also auch ohne Vorhandensein eines Netzwerkes einsetzen. In diesem Fall ist man auf die Überwachung lokaler Ressourcen und Dienste beschränkt.

```
define host{
     use                  linux-server
     host_name            ServerB
     alias                ServerB
     address              10.0.3.1
     check_command        CIMP!201,21%!10000,100%
     parents              RouterA
     }
```

Durch die Verwendung von `use` „erbt“ das neue Hostobjekt die Eigenschaften des Templates `linux-server`, welches bereits in der Grundkonfiguration enthalten ist und in Kapitel 3 näher beschrieben wird, `ServerB` ist der Name des Rechners. Jedes Hostobjekt benötigt einen Erreichbarkeitstest, der über das Feld `check_command` angegeben wird. Dabei ist `CIMP` der Name eines zugehörigen Kommandoobjektes(siehe unten), die übrigen Werte sind Parameter, die an dieses Kommandoobjekt übergeben werden sollen, das Ausrufungszeichen fungiert als Trennzeichen. Über das Attribut `parents` erhält das System Topologieinformationen. Hier werden diejenigen Netzwerknachbarn angegeben, über welche der jeweilige Host vom Nagios-System aus zu erreichen ist, im Beispiel ist `RouterA` quasi der „next hop“ von `ServerB` in Richtung des Nagios-Servers. Das Kommandoobjekt `CIMP:`

```
define command{
     command_name   CIMP
     command_line   $USER1$/check_icmp -H $HOSTADDRESS$
                    -w $ARG1$ -c $ARG2$
     }
```

Das Kommandoobjekt assoziiert also den im Hostobjekt angegebenen Kommandonamen mit dem tatsächlich aufgerufenen Befehl. Die in Dollarzeichen eingeschlossenen Bezeichner sind Makros: `$HOSTADDRESS$` wird zum Inhalt des Feldes `address` des jeweiligen Hosts, hier also zu 10.0.3.1 expandiert, `$ARG1$` und `$ARG2$` zu den im Aufruf des Kommandoobjektes angegebenen, dort durch Ausrufungszeichen getrennten Parametern und `$USER1$` schließlich enthält den absoluten Pfad zu den Nagios-Plugins. Wie man sieht, ist das Kommandoobjekt nicht vom Hostobjekt abhängig; es kann den Erreichbarkeitstest für beliebig viele weitere Hosts definieren.

1.2.5 Der Nagios-Kern

Das Hauptprogramm steuert die Aufrufe der Plugins, speichert die gelieferten Informationen und hält diese für die Darstellung durch die Weboberflächen-Routinen bereit. Ggf. arbeitet es Performancedaten auf, speichert diese in Dateien und es übernimmt die Steuerung des Benachrichtigungssystems. Im Folgenden sollen einige Details bzgl. der Arbeitsweise des Nagios-Kerns gegeben werden.

1.2.5.1 Host-Tests und Service-Tests

Wie bereits erwähnt, werden die eigentlichen Tests durch externe Programme, die Plugins durchgeführt. Dabei unterscheidet Nagios zwischen sogenannten Host-Tests und Service-Tests, die durch den Nagios-Kern entsprechend der Angaben in Host- und Service-Objekten in der Konfiguration angestoßen werden. Beim Host-Test wird die Erreichbarkeit eines Rechners/Servers im Netz überprüft, bei einem Service-Test wird die Verfügbarkeit eines Dienstes auf einem bestimmten Server getestet. Es wird also nicht etwa global geprüft, ob eine Anfrage an einen Dienst irgendwo im Netz verarbeitet bzw. beantwortet wird, sondern der Dienst ist hierbei strikt an eine feste Netzwerkkomponente gebunden. Werden beispielsweise nach einem Ausfall des DNS auf dem hauseigenen Nameserver die Anfragen von einem sekundären Nameserver bedient, so wird der Benutzer den Ausfall nicht bemerken. Nagios hingegen stellt den Fehler fest, da seine diesbezügliche Anfrage fest an die Adresse des Nameservers gerichtet ist.

Host- und Servicetest unterscheiden sich vor allem in der systeminternen Beschreibung ihres Zustandes. Der systeminterne Zustand von Services entspricht dem Rückgabewert des für seine Überwachung eingesetzten Plugins, ist also einer der Werte *OK, WARNING, CRITICAL* bzw. *UNKNOWN*. Bei den Hosts hingegen wird der Zustand durch einen der Werte *UP, DOWN* und *UNREACHABLE* beschrieben.[22]

22 Siehe 1.2.5.3

1.2.5.2 Harte und weiche Zustände

Bei der Definition jedes einzelnen Tests kann vom Administrator festgelegt werden, wie oft bzw. über welchen Zeitraum Fehlerzustände toleriert werden, bis eine Benachrichtigung erfolgt. Dazu unterscheidet Nagios sogenannte *soft states* und *hard states*. Tritt ein Fehler (`CRITICAL`, `WARNING` oder `UNKNOWN`**)** erstmalig auf, wechselt der Zustand des betroffenen Dienstes zunächst in einen soft state. Über den Parameter `max_check_attempts` des Serviceobjektes wird angegeben, wie oft der Test fehlschlagen muss, damit dieser Zustand in einen hard state übergeht - erst dann sendet Nagios eine Benachrichtigung. Wird dagegen innerhalb der durch `max_check_attempts` angegeben Anzahl der Versuche der Test einmal erfolgreich durchgeführt, bleibt eine Benachrichtigung aus, der Zustand wechselt ebenfalls wieder in einen hard state(`OK`).[23] Um einen - vielleicht nur kurzfristigen - Fehlerzustand schneller wieder korrigieren zu können, kann man zudem über das Attribut `retry_check_interval` für die Dauer eines soft states (im Gegensatz zum `normal_check_interval` für hard states) eine höhere Taktung der Tests erzwingen.[24]

1.2.5.3 Nutzung von Topologieinformationen

Über den Parameter `parents` der Hostobjekte erhält das System Informationen über die Netzwerktopologie. Hier gibt der Administrator für jede Netzwerkkomponente jeweils diejenigen Netzwerkknoten an, die „auf dem Wege zur Nagios-Installation benachbart“ liegen. Durch diese Informationen ist Nagios in der Lage, Fehlerquellen besser einzugrenzen:

Der interne Zustand eines Netzwerkgerätes ist einer der Werte `UP`, `DOWN` und `UNREACHABLE`. Er wird aus dem Ergebnis des Erreichbarkeitstest folgendermaßen abgeleitet: `UP` sind alle Hosts, bei denen der

23 Der Zustand `OK` ist immer ein hard state. Da es im Beispiel keinen abweichenden hard state gegeben hat, erfolgt auch keine Benachrichtigung. Bei dieser Vorgehensweise ergibt sich das Problem, dass es bei ständig wechselnden Zuständen zu keiner Benachrichtigung kommt. Als Abhilfe verfügt Nagios über eine flap detection, die in nagios.cfg aktiviert werden kann

24 retry_check_interval und normal_check_interval sind Attribute des zugehörigen Service-Objekts, können also für verschiedene Services unterschiedlich sein

Erreichbarkeitstest erfolgreich ist. Als DOWN werden alle Hosts bezeichnet, die eine direkte Verbindung zu einem Host im Zustand UP besitzen, aber nicht antworten. Als UNREACHABLE werden alle diejenigen nicht antwortenden Hosts gekennzeichnet, die über keine direkte Verbindung zu einem Host im Zustand UP verfügen, da hier keine Aussage darüber gemacht werden kann, ob der betreffende Host läuft oder nicht.

Schlägt ein Host-Check fehl, so wird Nagios zunächst überprüfen, ob ein sich in der durch die parents-Informationen festgelegten Hierarchie näher am Nagiosserver befindenden Hosts ebenfalls nicht erreichbar ist. In diesem Fall wird nur dieser als DOWN gekennzeichnet, alle abhängigen Hosts als UNREACHABLE[25]. Durch eine entsprechende Konfiguration des Benachrichtigungssystems kann dann z.B. erreicht werden, dass man nur über den Ausfall dieses Hosts unterrichtet wird und unzählige Benachrichtigungen über ausgefallene Hosts und Dienste, die aufgrund dieses Ausfalls nicht mehr funktionieren können, ausbleiben.

1.2.6 Benachrichtigungen

Sobald bei einem Host oder Dienst ein Wechsel des hard states vorliegt oder sich ein Fehlerzustand fortsetzt, wird das Nagios-Benachrichtigungssystem aktiv. Verschiedene Filter ermöglichen, Nachrichten zu unterdrücken, damit der Administrator nicht von Fehlermeldungen bombardiert wird, die Nachrichten können zudem auf unterschiedliche Personen entsprechend ihrer jeweiligen Zustandsbereiche verteilt werden. Beim Versenden der Nachrichten bedient sich Nagios wieder externen Programmen, in dieser Arbeit wird sich auf das Programm mail beschränkt, das auf Linux-Systemen grundsätzlich zur Verfügung steht und mit dem E-Mails über die Kommandozeile versendet werden können. Mail wird auch in den entsprechenden Kommandoobjekten der Nagios-Grundkonfiguration verwendet, sodass man auf diese Weise hier keine eigenen Definitionen anlegen muss.

25 Natürlich nur, wenn keine alternative Route existiert

1.2.6.1 Nachrichtenfilter

Das Benachrichtigungssystem kann systemweit mit dem Schalter `enable_notifications` in der `nagios.cfg` aus- bzw. eingeschaltet werden.[26]

In jeder Host- bzw. Serviceobjektdefinition gibt es die Filter `notifications_enabled`, `notification_options`, `notification_interval` und `notification_period`. Mit `notifications_enabled=0` lässt sich das Benachrichtigungssystem für diesen Host bzw. Dienst gänzlich deaktivieren. Bei `notification_options` gibt man eine kommaseparierte Liste der Zustände an, bei welchen eine Benachrichtigung erfolgen soll. Die zulässigen Optionen sind bei Hosts `u(unreachable)`, `d(DOWN)`, `r(RECOVERED)`, `f(FLAPPING)`, `n(NONE)`[27] sowie bei Diensten `c(CRITICAL)`, `w(WARNING)`, `r(RECOVERED)`, `u(UNKNOWN)`, `n(NONE)`, `f(FLAPPING)`. Den Zeitraum, in dem Nachrichten generiert werden sollen, bestimmt `notification_period`[28], die Wiederholrate der Nachrichten wird über `notification_interval` in Minuten angegeben.[29]

Über den Parameter `contact_groups` sind diejenigen Personengruppen anzugeben, die eine Nachricht erhalten sollen. Ein `contact_group`-Objekt enthält eine kommaseparierte Liste von `contact`-Objekten, welche ihrerseits jeweils Informationen wie Name, Benutzername, Emailadresse etc. einer Person enthalten. In den `contact`-Objekten gibt es außerdem eine letzte Stufe von Nachrichtenfiltern mit den dort zur Verfügung stehenden Parametern `host_notification_period`, `host_notification_options`, `service_notification_period` und `service_notification_options`, die analog den oben bereits erwähnten `notification_period` und `notification_options` in Host- bzw. Serviceobjekt zu verwenden sind. Ist diese letzte Hürde der Filterung genommen, erfolgt der Aufruf eines Kommandoobjekts, dass über den Parameter `host_notification_commands` bzw. `service_notification_commands` angegeben wird.

---

26 Voreinstellung ist `enable_notifications=1`(Benachrichtigungssystem eingeschaltet)

27 Durch die Verwendung der Option `n(none)` werden alle Nachrichten unterdrückt. Es ergibt sich also derselbe Effekt wie beim Ausschalten durch `notifications_enabled=0`.

28 Mit Hilfe eines Zeitfensterobjektes, siehe Beispiel in 1.2.6.2

29 Wiederholt werden die Nachrichten nur im Fehlerfall.

1.2.6.2 Beispiel

Betrachtet wird wieder der Rechner `ServerB` mit der Adresse `10.0.3.1`. In der in 1.2.4 angegebenen Hostobjektdefinition von `ServerB` wird das Template `linux-server` verwendet; in dessen Definition[30] sind die Filter `notifications_enabled`, `notification_options`, `notification_interval` und `notification_period` bereits eingestellt. Als nächstes werden ein `contact`- und ein `contact_group`-Objekt benötigt:

```
define contact{
     contact_name                    Anna
     use                             generic-contact
     host_notification_commands      notify-host-by-email
     email                           anna@ServerB
     }

define contactgroup{
     contactgroup_name               admins
     alias                           Nagios Administrators
     members                         anna
     }
```

Im hier verwendeten Template `generic-contact` aus der Nagios-Grundkonfiguration sind die Nachrichtenfilter so eingestellt, dass alle Nachrichten passieren können. `ServerB` ist der Name des Mailservers.[31] Das zugehörige Kommandoobjekt `host-notify-by-email` ist ebenfalls der Nagios-Grundkonfiguration entnommen:

```
define command{
     command_name   notify-host-by-email
     command_line   /usr/bin/printf "%b" "***** Nagios *****\n
                    Notification Type $NOTIFICATIONTYPE$\n
                    Host: $HOSTNAME$\nState: $HOSTSTATE$\n
                    Address: $HOSTADDRESS$\n
                    Info: $HOSTOUTPUT$\n\n
                    Date/Time $LONGDATETIME$\n"
                    |/usr/bin/mail -s " ** $NOTIFICATIONTYPE$
                    Host Alert: $HOSTNAME$ is $HOSTSTATE$ **"
                    $CONTACTEMAIL$
     }
```

30 Siehe Abschnitt 3.2

31 In Abschnitt 4.6 wird die Installation und Inbetriebnahme eines entsprechenden Mailservers beschrieben. Die Übereinstimmung von Mailserver und Beispielrechner ist rein zufällig und ohne weitere Bedeutung.

Die auf den ersten Blick unübersichtilich wirkende `command_line` ist ein Aufruf des Programms `mail`; Mailtext, Betreff und Empfängeradresse werden dabei mit Hilfe zahlreicher Nagios-Makros zusammengestellt. Provoziert man einen Fehler beim Beispiel des Host-Checks von `ServerB`, produziert der obige Befehl folgende Nachricht an `Anna@ServerB`.

(Betreff: `** PROBLEM Host Alert: ServerB is DOWN **`)

```
***** Nagios *****
Notification Type: PROBLEM
Host: ServerB
State: DOWN
Address: 10.0.3.1
Info: CRITICAL - 10.0.3.1: rta nan, lost 100%

Date/Time: Thu Jul 23 18:41:07 CEST 2009
```

2 Installation von Nagios

Meist besteht die Möglichkeit, von der verwendeten Linux-Distribution bereitgestellte Nagios-Pakete über den Paketmanager zu installieren. Hier soll jedoch die Installation von Nagios 3.0.1 aus den Quellcodes beschrieben werden, wie sie für diese Arbeit durchgeführt wurde: Da die Entwicklung von Nagios z.Z. rege vorangetrieben wird, können die in den Distributionen angebotenen Pakete der jeweils aktuellen Version ein wenig „hinterherhinken“. Im ersten Abschnitt soll zunächst auf die Besonderheiten der Installation von Software allgemein unter VNUML eingegangen werden, die eigentliche Installation wird in den Abschnitten 2.2 und 2.3 beschrieben. Diese Trennung erfolgte, da sich die Installation von Nagios in einer VNUML-Umgebung nicht von der auf einem normalen Linux-System unterscheidet.

2.1 VNUML - Installation von Software

Unter der Bezeichnung User Mode Linux (UML) versteht man den Betrieb eines Linux-Kernels als Anwendungsprozess. Virtual Network UML ist eine Erweiterung von UML, bei der mehrere solcher Prozesse durch Softwareschnittstellen miteinander vernetzt werden. Das Resultat ist ein virtuelles Netzwerk, in welchem z.B. verschiedene Netzwerk- und Routingprotokolle getestet bzw. deren Funktionalität demonstriert werden können. Grundlage für ein solches VNUML-Netz (auch als VNUML-Szenario bezeichnet) ist dabei stets eine XML-Datei, in der vom Benutzer Namen, Anzahl, Netzwerkadressen etc. der virtuellen Maschinen festgelegt werden.[32] Das VNUML-System übernimmt dann den Start der virtuellen Maschinen und stellt die vom Benutzer gewünschten Verbindungen her.

Insbesondere kann eine virtuelle Verbindung zum Hostrechner hergestellt werden, über dessen Internetverbindung den virtuellen Maschinen der Zugriff auf das Internet ermöglicht werden kann, was die Installation von Software in das Dateisystem der virtuellen Maschinen beträchtlich erleichtert.

Der Zugriff auf die einzelnen Maschinen kann z.B. über eine in einem

32 Einzelheiten zum Erstellen einer solchen XML-Datei, zur Installation und Handhabung des VNUML-Systems findet man auf der VNUML-Heimseite `http://www.dit.upm.es/vnumlwiki/index.php/Main_Page`

Konsolenfenster hergestellte ssh-Verbindung erfolgen.

Um (Festplatten-)Speicherplatz zu sparen, verwenden die UML-Maschinen ein gemeinsames Image, das den größten Teil des verwendeten Dateisystems enthält. Änderungen am Dateisystem, die sich im laufenden Betrieb ergeben und also üblicherweise nicht alle Maschinen betreffen, werden in sogenannten COW-Dateien[33], welche wesentlich kleiner sind, festgehalten; dabei existiert für jede virtuelle Maschine eine eigene solche Datei.

In diesem Kapitel soll die Installation von Nagios ins Dateisystem der UML-Maschinen beschrieben werden, in diesem ersten Abschnitt finden dabei nur die das VNUML-System bzw. das COW-Dateisystem betreffenden Besonderheiten Beachtung, Grundlagen im Umgang mit dem VNUML-System werden dabei vorausgesetzt. Da der eigentliche Installationsvorgang in einer „gewöhnlichen“ Linux-Umgebung derselbe ist, wird dieser getrennt in den Abschnitten 2.2 und 2.3 behandelt.

Im folgenden sei ein VNUML-Szenario mit dem Namen *Nagios*, definiert in einer Datei `nagios.xml`[34], vorausgesetzt. In `nagios.xml` sei eine Maschine mit dem Namen `NagiosServer` definiert, auf welcher die Installation erfolgen soll, und es sei eine virtuelle Schnittstelle zum Hostrechner vorhanden, der seinerseits über eine Internetanbindung verfügt[35]. Zunächst stoppt man das Szenario, falls sich dieses noch in Ausführung befindet.

```
Host:#[36] vnumlparser.pl -d nagios.xml -u root
```

Da Nagios aus den Quellen übersetzt werden soll, kopiert man diese zunächst manuell in die Imagedatei, die man vorher mit Hilfe des mount-Befehls in das Dateisystem des Hosts ein- und hinterher wieder aushängt[37].

33 COW steht für Copy-On-Write

34 Das Listing der Datei `nagios.cfg` findet man im Anhang.

35 Die Topologie des Netzes wird im nächsten Kapitel genauer beschrieben, vorläufig genügt es, sich auf einen der virtuellen Rechner zu beschränken.

36 Um Missverständnisse zu vermeiden, ist bei der Wiedergabe von Konsolenbefehlen stets der Name des betreffenden Rechners mit angegeben, Host steht für den physikalischen Rechner, auf dem das VNUML-Szenario ausgeführt wird. Auf die Angabe des Arbeitsverzeichnisses wird dagegen aus Platzgründen verzichtet.

37 Meist sind keine manuellen Veränderungen am Image nötig, alle übrigen Programme wurden ausnahmslos mithilfe eines Paketmanagers installiert. Alternativ zur hier gewählten Vorgehensweise könnten die Nagios-Quellen bei laufendem Szenario z.B. mittels scp in das Dateisystem kopiert werden. Das direkte Beziehen der Quellen aus dem Internet via ftp auf die virtuelle Maschine stellt eine weitere Möglichkeit dar.

```
Host:# mount -o loop imagefile /mnt
Host:# cp nagios-3.0.1.tar.gz[38] /mnt/usr/local/src
Host:# cp nagios-plugins-1.4.8.tar.gz /mnt/usr/local/src
Host:# umount /mnt
```

Durch die Manipulation der Imagedatei sind evtl. vorhandene COW-Dateien ungültig geworden. Sie sind vor dem erneuten Start der Simulation zu löschen, es werden beim anschließenden Neustart des VNUML-Szenarios automatisch wieder neue angelegt.

```
Host:# rm -r ~/.vnuml/simulations/nagios/vms/*/root_cow_fs
Host:# vnumlparser.pl -t nagios.xml -u root -vB
```

Damit von `NagiosServer` aus über die Schnittstelle zum Host auf das Internet zugegriffen werden kann, richtet man eine Adressumsetzung mittels NAT ein und aktiviert IP-Forwarding:

```
Host:# iptables -t nat -o eth0 -j MASQUERADE      →
          -A POSTROUTING[39]
Host:# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Anschließend kann die Installation gemäß den Ausführungen in den Abschnitten 2.2 bzw. 2.3 auf `NagiosServer` durchgeführt werden (siehe dort). Da die Speicherkapazität der Image-Datei recht knapp bemessen ist, löscht man nach erfolgreicher Installation am besten den entstandenen, unnötigen „Ballast“ wieder:

```
NagiosServer:# cd /usr/local/src
NagiosServer:# rm -r nagios-plugins*
NagiosServer:# rm -r nagios-3.0.*
```

Man stoppt das Szenario und kann abschließend die Änderungen, die bisher nur die neue COW-Datei betreffen, mithilfe des Befehls `uml_moo` in das Image übernehmen. Dadurch werden die COW-Dateien erneut ungültig, weshalb sie wieder gelöscht werden müssen.

38 Die Quellen bekommt man auf der Nagios-Hauptseite www.nagios.org

39 Dabei ist eth0 das Interface, über das der Hostrechner mit dem Internet verbunden ist.

```
Host:# vnumlparser.pl -d nagios.xml -u root
Host:# uml_moo -d cow-file[40]
Host:# rm -r ~/.vnuml/simulations/nagios/vms/*
```

Da, wie bereits erwähnt, mehrere bzw. alle virtuellen Maschinen das oben manipulierte Image gemeinsam verwenden, steht die erstellte Nagios-Installation auch auf allen hiervon betroffenen Maschinen zur Verfügung. Ebenso sind die neu erstellten Benutzerkonten, Verzeichnisse etc.[41] jeweils vorhanden. Da aber nicht auf allen diesen Maschinen Nagios ausgeführt werden soll, sondern nur auf der dafür vorgesehenen Maschine `NagiosServer`, ist ein automatisches Starten von Nagios beim Systemstart wie in (Barth 2009, S.41) beschrieben, nur hinderlich. Stattdessen muss Nagios nach jedem Start des Szenarios auf dem Nagios-Server manuell gestartet werden[42].

2.2 Vorbereiten der Installation

Der Nachteil der Übersetzung aus den Quellpaketen ist, dass zunächst eine ganze Reihe anderer Pakete installiert werden muss (dieses aber mit Hilfe des Paketmanagers).

Nachfolgend eine Liste der Pakete, die für eine anschließende reibungslose Übersetzung[43] mittels APT[44] installiert wurden.

```
autoconf          libssl-dev          libmysqlclient15off
automake1.9       fping               libmysqlclient15-dev
```

40 Hierdurch werden COW- und Image-Datei miteinander verschmolzen und das Image durch das Resultat überschrieben. Man sollte also ggf. (wenn man z.B. nicht sicher ist, dass die Installation erfolgreich war) vor Ausführung von `uml-moo` eine Sicherungskopie der Image-Datei erstellen.

41 Siehe Abschnitt 2.2

42 Zu diesem Zweck wird in `nagios.xml` ein entsprechendes exec-tag eingefügt, siehe Abschnitt 4.1.1. Ebenso kann man auch beim Start der verschiedenen Netzwerkdienste verfahren, wenn der Dienst nur auf einer Maschine ausgeführt werden soll.

43 Nicht jedes der aufgelisteten Pakete ist für den Betrieb von Nagios unbedingt erforderlich, es ergeben sich aber ggf. Probleme bei der Übersetzung des ein oder anderen Plugins. Um von vornherein möglichst flexibel zu bleiben, wurde hier darauf geachtet, später möglichst viele Plugins zur Verfügung zu haben.

44 Advanced Packaging Tool, Paketverwaltungsystem von Debian-Linux, dass insbesondere auf den virtuellen Maschinen der VNUML-Umgebung zur Verfügung steht.

```
libgd-tools          libc6-dev           libltdl3-dev
libgd2-xpm-dev       libldap-2.4-2       libnet-snmp-perl
libldap2-dev         openssl             snmp
postgresql-client    build-essential     smbclient
libpq-dev            libapache2-mod-php5[45]
```

Außerdem wird ein Benutzer `nagios`[46] benötigt, mit dessen Rechten Nagios ausgeführt werden wird, eine zugehörige Benutzergruppe mit gleichem Namen sowie eine Benutzergruppe `nagcmd`[47], der neben `nagios` auch derjenige Benutzer angehören soll, unter dem der Webserver ausgeführt wird[48] und den man am einfachsten aus der Webserver-Konfigurationsdatei `httpd.conf`[49] ermittelt:

```
NagiosServer:# groupadd -g 9000 nagios
NagiosServer:# groupadd -g 9001 nagcmd
NagiosServer:# useradd -u 9000 -g nagios -G nagcmd     →
               -d /usr/local/nagios -c "Nagios Admin" nagios
NagiosServer:# locate apache2.conf
etc/apache2/apache2.conf
NagiosServer:# grep "^User" /etc/apache2/apache2.conf
User  ${APACHE_RUN_USER}
NagiosServer:# grep "APACHE_RUN_USER" /etc/apache2/envvars
export APACHE_RUN_USER=www-data
NagiosServer:# usermod -G nagcmd www-data
```

Weiterhin benötigt man das Homeverzeichnis `/usr/local/nagios` des Benutzers `nagios`, ein Verzeichnis `/etc/nagios` für die Ablage der Konfigurationsdateien sowie eines für veränderliche Daten, die während des Betriebs von Nagios anfallen (`/var/nagios`). Besitzer ist jeweils der Benutzer `nagios`:

45 Bei anderen Linux-Distributionen können die Bezeichner für die einzelnen Pakete abweichen.

46 Die in diesem Abschnitt gewählten Bezeichner für Benutzer, Gruppen und Verzeichnisse sind zum Teil von Nagios vorgeschrieben, ansonsten wird den Empfehlungen in (Barth 2009, S.36ff) gefolgt.

47 Die sogenannte Nagios Command Group.

48 Sonst kann die Weboberfläche wegen Berechtigungskonflikten nicht uneingeschränkt verwendet werden.

49 Je nach Webserver-Version bzw. Linux-Distribution findet man die gesuchten Angaben auch in einer Datei `apache.conf` bzw. `httpd.conf`, vgl. (Barth 2009, S.37).

```
NagiosServer:# mkdir /usr/local/nagios /etc/nagios     →
               /var/nagios
NagiosServer:# chown nagios.nagios /usr/local/nagios   →
               /etc/nagios /var/nagios
```

2.3 Installation

Von der Website `http://www.nagios.org/` kann man sich den Nagios-Quellcode und den der Plugins (als tarballs: `nagios-3.2.3.tar.gz` bzw. `nagios-plugins-1.4.15.tar.gz`[50]) herunterladen. Das Paket `nagios-3.0.1.tar.gz` entpackt man in ein Verzeichnis `/usr/local/src`[51] und wechselt anschließend in das dadurch neu angelegte Verzeichnis:

```
NagiosServer:# mkdir /usr/local/src
NagiosServer:# mv nagios-3.2.3.tar.gz /usr/local/src
NagiosServer:# cd /usr/local/src
NagiosServer:# tar xvzf nagios-3.2.3.tar.gz
NagiosServer:# cd nagios-3.2.3
```

Durch den folgenden `configure-Aufruf` werden die Quellen für die Übersetzung vorbereitet, die in 2.1 angelegten Verzeichnisse und Gruppen übergibt man über entsprechende Parameter:

```
NagiosServer:# ./configure -–sysconfdir=/etc/nagios    →
               --localstatedir=/var/nagios              →
               --with-command-group=nagcmd
```

Die detaillierten Ausgaben zum `configure`-Aufruf sollte man sich genauer ansehen, um ggf. fehlende Pakete vor der eigentlichen Übersetzung noch rechtzeitig nachinstallieren zu können. Dann wiederholt man diese Prozedur, bis man mit der Ausgabe zufrieden ist und fährt erst anschließend mit der Installation fort:

50 Die Plugins unterliegen einer eigenen Versionierung

51 Die Bezeichnerwahl für Verzeichnisse, Benutzer-/Gruppennamen, Benutzer-/Gruppen-IDs etc. orientiert sich an den Empfehlungen in (Barth 2009, S.26ff).

```
NagiosServer:# make all          (Übersetzung der Nagios-Quellen)
NagiosServer:# make install      (Installation des Hauptprogramms[52]
                                  und der CGI-Skripte[53])
NagiosServer:# make install-init   (Anlegen eines Init-Skriptes[54])
NagiosServer:# make install-commandmode
NagiosServer:# make install-config       (Anlegen der Nagios-
                                          Beispielkonfiguration[55])
```

Nun sind noch die Nagios-Plugins zu installieren. Man entpackt das Plugin-Paket wiederum im Verzeichnis `/usr/local/src`, gibt wie oben beim `configure`-Aufruf[56] die gewünschten Verzeichnisse an und startet die abschließende Übersetzung mit `make`:

```
NagiosServer:# mv nagios-plugins-1.4.15.tar.gz /usr/local/src
NagiosServer:# cd /usr/local/src
NagiosServer:# tar xvzf  nagios-plugins-1.4.15.tar.gz
NagiosServer:# cd  nagios-plugins-1.4.15
NagiosServer:# ./configure –sysconfdir=/etc/nagios     →
               --localstatedir=/var/nagios             →
               --with-nagios-user=nagios               →
               --with-nagios-group=nagios
NagiosServer:# make
NagiosServer:# make install
```

Wenn alles wunschgemäß funktioniert hat, befinden sich die übersetzten Plugins anschließend im Verzeichnis `/usr/local/nagios/libexec`.

Um die Nagios-Weboberfläche erreichbar zu machen, muss man den Webserver entsprechend einstellen. Im Wesentlichen muss lediglich eine Umsetzung der in der Adressleiste des Browsers eingegebenen (bzw. durch CGI-Skripte der Weboberfläche von Nagios selbst erzeugten) Adressen auf den tatsächlichen „Ort“ der CGI-Skripte im Dateisystem durch den Webserver

52 nach /usr/local/nagios/bin/

53 nach /usr/local/nagios/sbin/. Nagios kommuniziert mit dem Webserver über das Common Gateway Interface. Die CGI-Skripte sind die für die Erzeugung der Weboberfläche zuständigen Module.

54 in /etc/init.d/

55 in /etc/nagios/

56 Wiederum ist es zweckmäßig, die Ausgabe gründlich zu studieren und ggf. benötigte Pakete nachzuinstallieren.

vorgenommen werden, hierzu bedient man sich der ScriptAlias-Direktive in der Apache-Konfiguration. Musste man früher selbst Hand an diese Dateien anlegen, so kann man seit Nagios Version 3 eine entsprechende Konfigurationsdatei automatisch erzeugen lassen:

```
NagiosServer:# cd /usr/local/src/nagios-3.2.3
NagiosServer:# make install-webconf
```

Dadurch wird eine Datei `nagios.conf` in `/etc/apache2/conf.d/` angelegt, welche die nötigen Anweisungen enthält; sie werden bei einem Neustart des Webservers wirksam[57].

```
NagiosServer:# /etc/init.d/apache2 restart
```

Da mit der Weboberfläche nicht nur Ausgaben von Nagios dargestellt, sondern über sie auch das Verhalten des Systems gesteuert werden kann, verhindert man unberechtigten Zugriff üblicherweise durch den Einsatz von passwortgeschützten Accounts. Einen solchen kann man z.B. anlegen mit:

```
NagiosServer:# htpasswd -c /etc/nagios/htpasswd.users  →
               nagiosadmin
New password: xxxx
Re-type new password: xxxx
```

Dabei ist auf die Datei `/etc/nagios/htpasswd.users`, die das Benutzer-Passwort-Paar aufnimmt, in der oben automatisch erzeugten Datei `/etc/apache2/conf.d/nagios.conf` verwiesen, ein anderer Dateiname ist also nicht zulässig bzw. führt nicht zum gewünschten Verhalten, solange man die entsprechenden Einstellungen in `nagios.conf` nicht entsprechend korrigiert. Ebenso empfiehlt es sich, den angegebenen Benutzernamen, `nagiosadmin`, zu verwenden. Nur `nagiosadmin` ist nämlich berechtigt, sämtliche Funktionen, die die Weboberfläche bietet, zu nutzen - bedingt durch entsprechende Voreinstellungen in der Konfigurationsdatei für die CGI-Skripte `/etc/nagios/cgi.cfg`.

57 Ggf. ist zu überprüfen, ob in der Hauptkonfigurationsdatei des Webservers eine entsprechende Include-Anweisung auch tatsächlich vorhanden ist.

2.4 Testen der Installation

Es empfiehlt sich an dieser Stelle zu überprüfen, ob die Installation erfolgreich war. Dazu startet man Nagios über das Init-Skript in `/etc/init.d/`,außerdem muss auf `NagiosServer` auch der Webserver Apache laufen, der analog gestartet wird:

```
NagiosServer:# /etc/init.d/nagios start
Starting nagios: done.
NagiosServer:# /etc/init.d/apache2 start
Starting web server: apache2apache2: Could not reliably
determine the server's fully qualified domain name, using
10.0.5.1 for ServerName
```

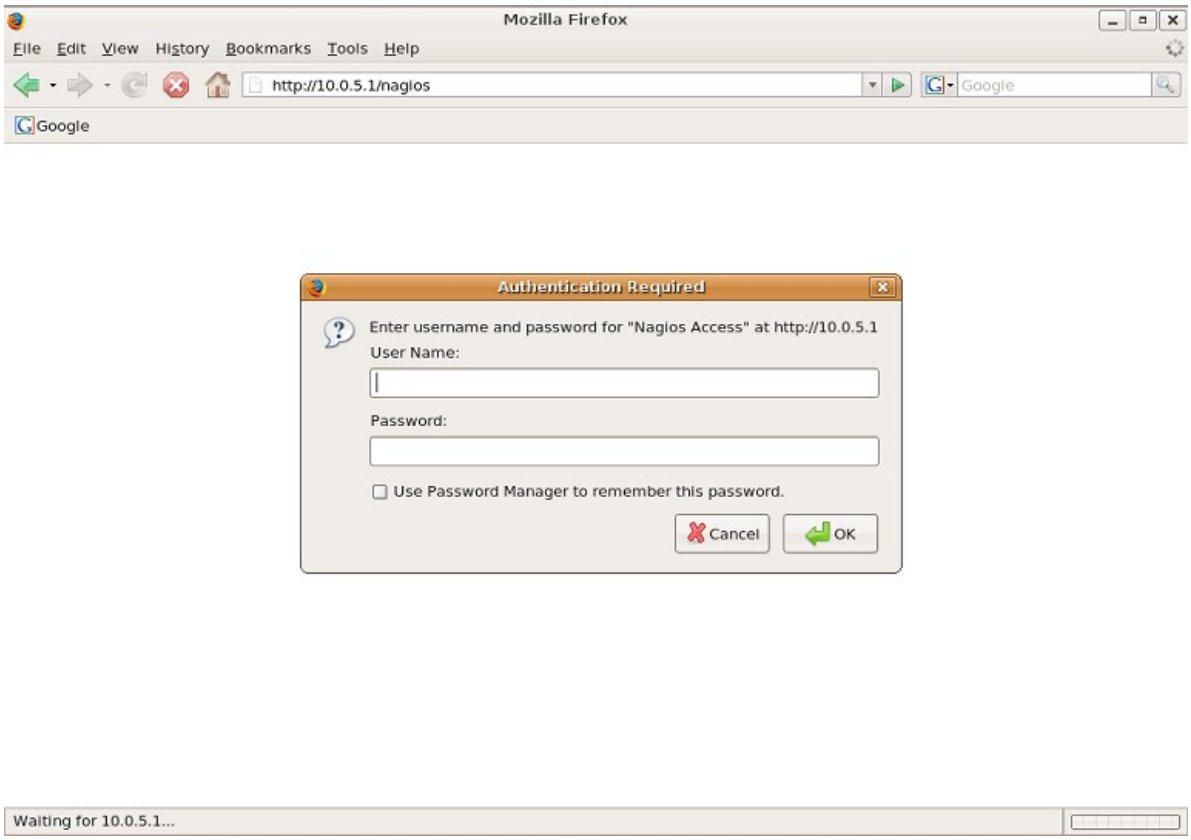

Abb.4 Nagios-Login-Panel

Anschließend sollte auf die Weboberfläche des Systems zugegriffen werden können, in dem man `http://`*`nagiosserver-address`*`/nagios`[58] in der Adressleiste eines auf dem Hostrechner geöffneten Browserfensters eingibt. Man gelangt zunächst zu einem Login-Panel (Abb.4), nach der Eingabe des Benutzernamens (`nagiosadmin`) und des Passworts (`xxxx`) erreicht man den in Abb.5 gezeigten Begrüßungsbildschirm. Darunter die Anzeige der *Status Map* (Abb.6), welche die Topologie des überwachten Netzwerkes auf dem Bildschirm darstellt, gegenwärtig ist nur der lokale Rechner mit der auf ihm laufenden Nagios-Installation zu sehen.In Abb.7, die eine Liste aller überwachten Services zeigt (Schaltfläche *Service Detail*), erkennt man anhand des zurückgegebenen Status `OK` (grün unterlegt), dass die einzelnen von der Beispielkonfiguration verwendeten Plugins ebenfalls arbeiten. Eine Ausnahme bildet der Service *Root Partition*, der rot unterlegt ist, da er sich im Zustand `CRITICAL` befindet: Hiermit wird der freie Speicherplatz auf der Systempartition überwacht und dieser ist, wie schon angesprochen, bei den virtuellen Maschinen oft knapp bemessen.

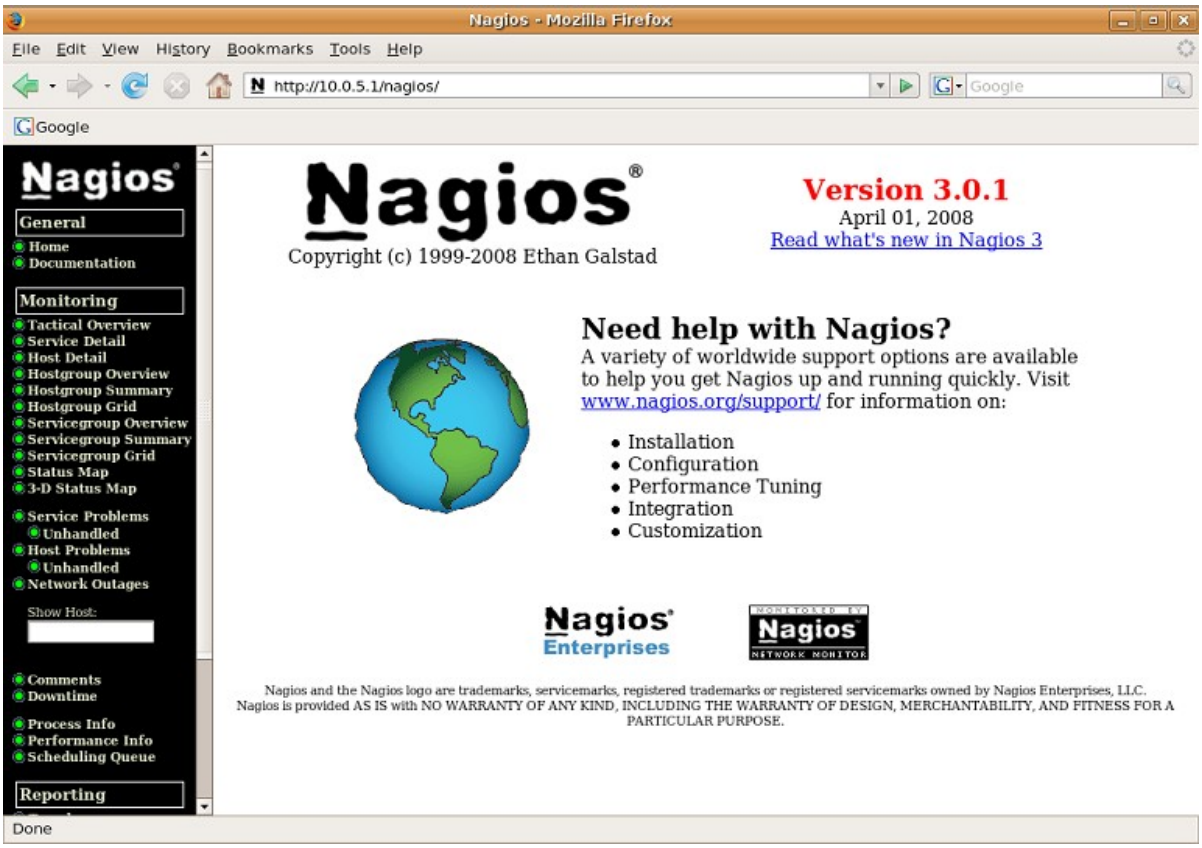

Abb.5 Begrüßungsbildschirm

58 Dabei ist *`nagiosserver-address`* natürlich durch die entsprechende Adresse zu ersetzen.

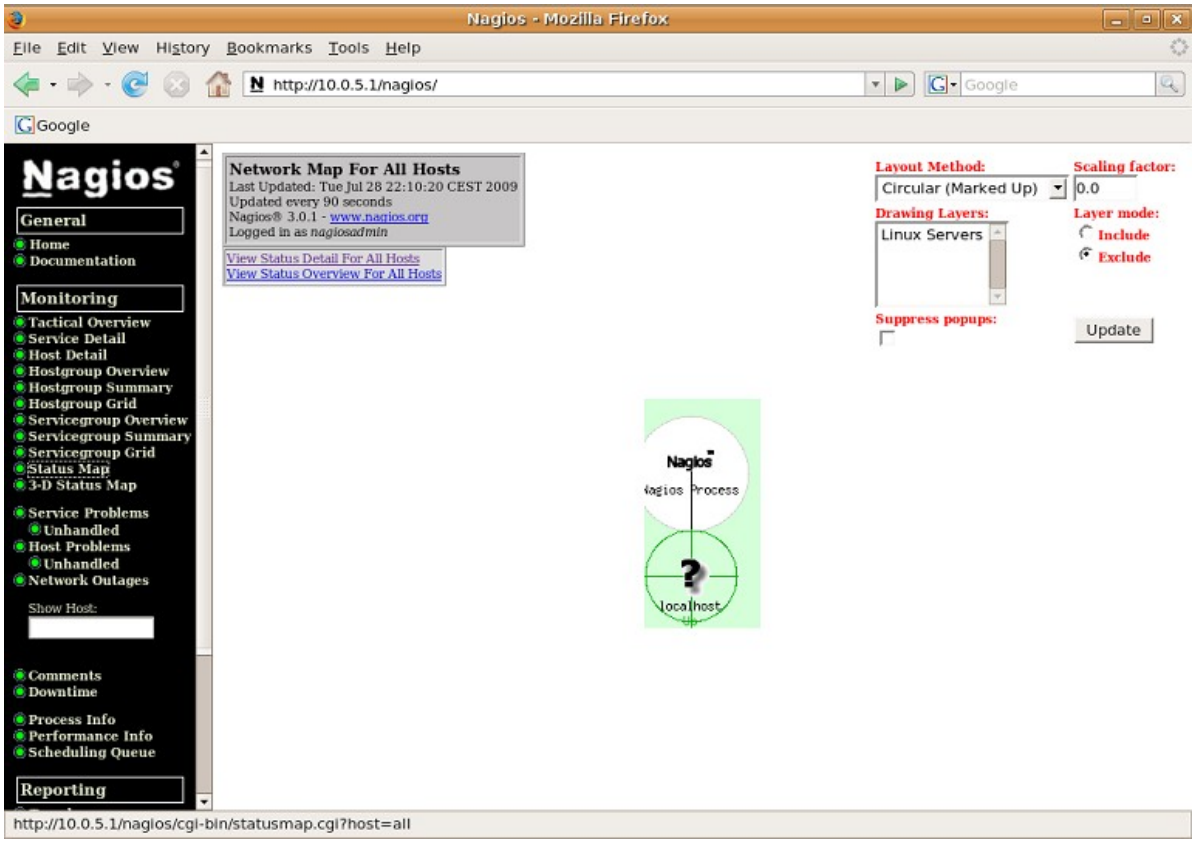

Abb.6 Status Map

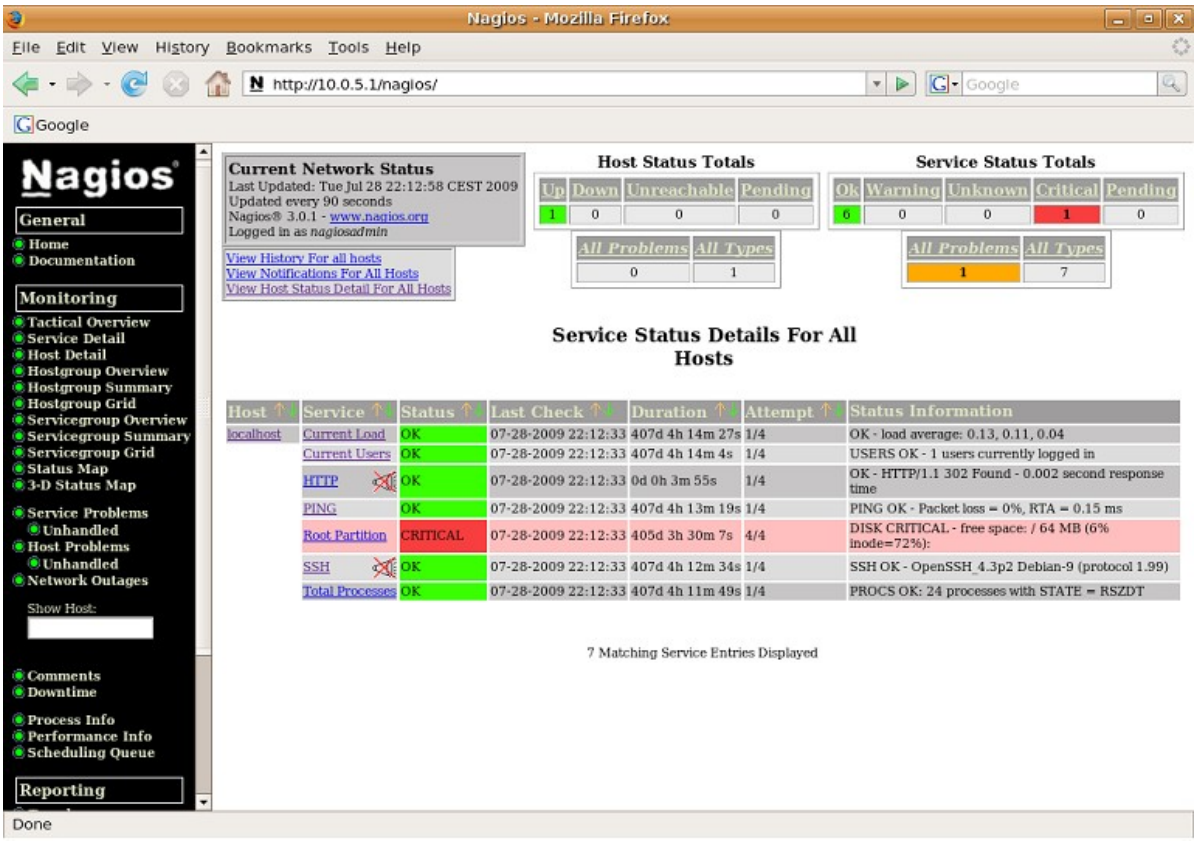

Abb.7 Service Detail

Die übrigen Dienste sind *Current Load* (Auslastung der CPU), *Current Users* (Anzahl der eingeloggten Benutzer), *PING* (ICMP-Request-Check, ähnlich dem Host-Check mit `check_icmp`), *HTTP*, *SSH* (ausführliche Besprechung in Kapitel 4) und *Total Processes* (aktuelle Anzahl aller Prozesse).

Da die Plugins eigenständige Programme sind, können diese manuell und einzeln getestet werden. Ein Beispiel für einen Plugin-Aufruf wurde bereits in Kapitel 1 für das Plugin `check_icmp` vorgestellt. Um diesen zu testen, wechselt man in das Verzeichnis, in welches die Plugins installiert wurden und startet es dort (mit vorangestelltem „./“):

```
NagiosServer:# cd /usr/local/nagios/libexec       →
NagiosServer:# ./check_icmp -H 10.0.3.1 -w 201,21%
               -c 10000,100%
OK – 10.0.3.1: rta 3.340ms, lost 0%               →
|rta=3.340ms;201.000;10000.000;0; pl=0%;21;100;;
```

3 . Die Nagios-Konfiguration

3.1VNUML-Szenario

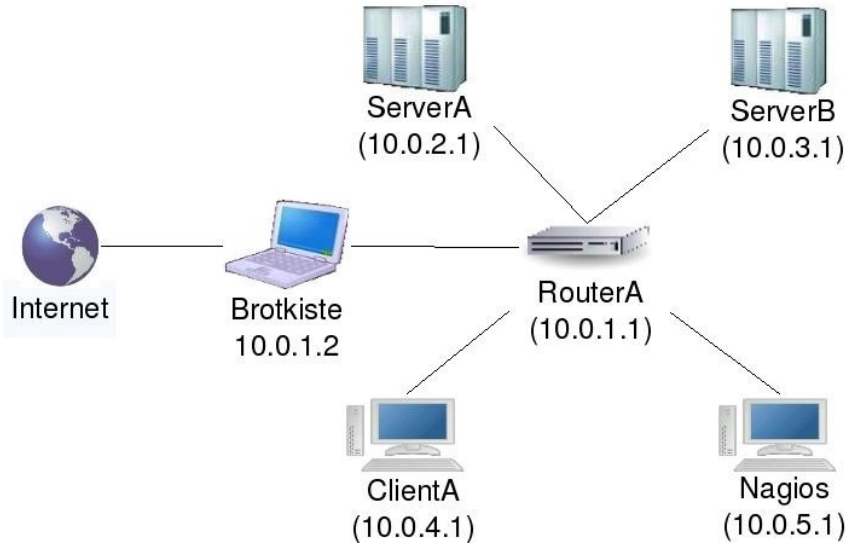

Abb.8: VNUML-Szenario „nagios“

Abb.8 zeigt das in dieser Arbeit verwendete VNUML-Szenario. Das Netz mit insgesamt fünf virtuellen Maschinen soll mit dem auf dem Rechner `Nagios` laufenden Nagios-System überwacht werden. Der Host `Nagios`[59] ist mit der `RouterA` genannten Maschine direkt verbunden, diese hat wiederum jeweils eine direkte Verbindung zu den übrigen Maschinen `HostA`, `Brotkiste`, `ServerA` und `ServerB`. `HostA` ist stellvertretend für in einem Netzwerk befindliche Arbeitsstationen, auf `ServerA` und `ServerB` sollen im nächsten Kapitel verschiedene Netzwerkdienste installiert werden, gleichzeitig wird die Nagios-Konfiguration sukzessive ausgebaut. `Brotkiste` ist der physikalische Rechner, auf dem das VNUML-Szenario ausgeführt wird und der eine Internetanbindung bereitstellt.

Einen besonderen Grund für die gewählte Netztopologie und die Anzahl der Maschinen gibt es dabei nicht[60]; grundsätzlich sind die folgenden

59 Der Host Nagios entspricht der im vorigen Kapitel mit Nagios-Server bezeichneten virtuelle Maschine.

60 Die verschiedenen Netzwerkdienste wurden ursprünglich auf zwei Server verteilt, um ggf. verteilte Dienste simulieren zu können. Insbesondere sollte die Möglichkeit bestehen, auf dem zweiten Server einen sekundären Nameserver einzurichten, es wurde der Einfachheit wegen dann aber doch hierauf verzichtet.

Betrachtungen auf beliebige Netze übertragbar.

3.2 Hostobjekte

Zunächst sollen die sogenannten Hostobjekt-Definitionen beschrieben werden, über welche dem Nagios-System die einzelnen Komponenten des Netzwerks bekanntgemacht werden. Für den Rechner `Nagios` ist eine solche bereits in `/etc/nagios/localhost.cfg` vorhanden, für die Hostobjekt-Definitionen der virtuellen Maschinen des VNUML-Szenarios wird eine eigene Datei angelegt:

```
Nagios:# touch /etc/nagios/vnumlhosts.cfg
```

Damit diese von Nagios auch berücksichtigt wird, ist sie dem System in `nagios.cfg` mittels einer Zeile `cfg_file=/etc/nagios/vnumlhosts.cfg` zugänglich zu machen. In die Datei `vnumlhosts.cfg` können dann die Hostobjekt-Definitionen für die fünf noch fehlenden Maschinen geschrieben werden, deren Aussehen und Inhalt am Beispiel des Eintrags für `RouterA` näher erläutert werden sollen:

```
define host{
     use                     linux-server
     host_name               RouterA
     alias                   RouterA
     address                 10.0.1.1
     parents                 localhost
     }
```

Dabei bedeuten im Einzelnen:

- *use*: Über das Attribut use können Eigenschaften zuvor definierter Objekte, sogenannter Templates an das Hostobjekt vererbt werden. Eine Betrachtung des an dieser Stelle verwendeten Templates `linux-server` folgt weiter unten in diesem Abschnitt.
- *host_name*: ordnet dem Gerät einen eindeutigen Namen zu, über den der Komponente Service- und Kommandoobjekte etc. zugeordnet werden können.

- *alias*: Zusätzliche Informationen, etwa zur Ausgabe in der Weboberfläche. Für die Maschinen in diesem Beispiel sind die Werte für host_name und alias stets identisch (die Namen der Hosts sind ja bereits in gewissem Sinne selbsterklärend gewählt)[61].

- *address*: Eine Netzwerkadresse, unter der das Gerät erreichbar ist. Für die UML-Maschinen ist das eine der Adressen der benutzen Interfaces aus der XML-Datei. Für den Nagios-Server `Nagios` verwendet man am einfachsten die Adresse des Loopback-Interfaces (`127.0.0.1`). An diese Adressen richten sich jeweils der Erreichbarkeitstest sowie Service-Tests, die mit dem jeweiligen Gerät assoziiert sind.

- *parents*: Über das Attribut `parents` erhält Nagios Informationen über die Topologie des Netzwerks: Unter `parents` sind die Namen[62] derjenigen direkt verbundenen Komponenten anzugeben, die sich „auf dem Wege zur Nagios-Installation“ befinden. Für die vier Maschinen `HostA`, `Brotkiste`, `ServerA` und `Server B` ist hier also jeweils `RouterA` anzugeben, für `RouterA` ergibt sich der Eintrag `localhost` (=`Nagios`) und beim Rechner `Nagios` selbst bleibt das Feld leer.

Es folgen die Definitionen der Templates `linux-server` und `generic-host`[63], dessen sich in der Definition von `linux-server` bedient wird. Beide Templates wurden aus der mittels `make install-config`[64] erstellten Beispielkonfiguration übernommen. Anschließend werden die für die weiteren Betrachtungen relevanten Punkte nochmal erläutert.

```
define host{
        name                            linux-server
        use                             generic-host
```

61 Die unter `host_name` und `alias` angegebenen Bezeichner sowie die in der XML-Datei angegebenen Namen der Geräte können durchaus voneinander abweichen. Um Verwirrung zu vermeiden, wurden die Geräte einheitlich bezeichnet.

62 Unter Namen sind hier die jeweils unter für `host_name` angegebenen Werte zu verstehen.

63 Die Definitionen der Templates findet man in der Datei `/etc/nagios/objects/templates.cfg`

64 siehe Abschnitt 2.2

```
        check_period                    24x7
        check_interval                  5
        retry_interval                  1
        max_check_attempts              10
        check_command                   check-host-alive
        notification_period             workhours
        notification_interval           120
        notification_options            d,u,r
        contact_groups                  admins
        register                        0
        }

define host{
        name                            generic-host
        notifications_enabled           1
        event_handler_enabled           1
        flap_detection_enabled          1
        failure_prediction_enabled      1
        process_perf_data               1
        retain_status_information       1
        retain_nonstatus_information    1
        notification_period             24x7
        register                        0
        }
```

- *check_period*: Hier gibt man den Zeitraum an, in welchem das Gerät überwacht werden soll. Das etwas kryptisch anmutende `24x7` ist ein Bezeichner für ein Zeitfenster(`timeperiod`)-Objekt, und ist als eine Abkürzung für „24 Stunden am Tag, 7 Tage die Woche“ zu verstehen. Etwas weiter unten bezeichnet `workhours` ebenfalls ein solches Zeitfensterobjekt. Es überdeckt den Zeitraum von 9 Uhr morgens bis 5 Uhr nachmittags und das von Montag bis einschließlich Freitag.

- *check_interval*: Die Zeitspanne in Minuten, bis der nächste Hostcheck durchgeführt wird, wenn der letzte Hostcheck erfolgreich war

- *retry_interval*: Die Zeitspanne bis zum nächsten Hostcheck, falls der letzte Test *nicht* erfolgreich war

- *max_check_attempts*: Gibt an, wie oft der Erreichbarkeitstest fehlschlagen darf, bis das Nagios-Benachrichtigungssystem

zuständige Personen(-gruppen) informiert, also bis zu einer Änderung des hard state.

- *check_command*: Definiert ein Kommandoobjekt, das für den Erreichbarkeitstest aufgerufen wird.

- *notification_period, notification_interval, notification_options, contact_groups, notifications_enabled*: Diese Angaben dienen der Steuerung des Nagios-Benachrichtigungssystems und ihre Bedeutung wurde bereits in Abschnitt 1.2.6 beschrieben.

- *register*: Durch die Angabe von 0 teilt man dem System mit, dass es sich nur um ein Template handelt, das nicht wie ein realer Rechner im Netz zu behandeln ist.

In der Definition von `generic-host` schließlich sind verschiedene Flags gesetzt, die jedoch allesamt optional sind. Ihnen wird im folgenden nur an denjenigen Stellen Beachtung geschenkt, an denen eine bestimmte Einstellung erforderlich ist. Es fällt auf, dass das Attribut `notification_period` in beiden Templates definiert ist, dabei „überschreibt“ die Angabe in der Definition von `linux-server` diejenige aus der Definition von `generic-host`. Die Templates können also auch verwendet werden, wenn man mit den Werten für einzelne Attribute nicht einverstanden ist - diese kann man dann in den neuen Definitionen einfach überschreiben.

Für Testzwecke sind die Voreinstellungen für die einzelnen Timer ziemlich lang. Auch ist es nicht sinnvoll, Benachrichtigungen auszufiltern. Es bietet sich daher an, das Template `linux_server` z.B. wie folgt zu modifizieren, um die Reaktionszeit des Systems zu verkürzen und das Filtern von Benachrichtigungen zu unterbinden (in Klammern rechts daneben jeweils die alten, voreingestellten Werte):

```
define host{
     name                            linux-server
     use                             generic-host
     check_period                    24x7
     check_interval                  0.5             ;(5)
     retry_interval                  0.1             ;(1)
```

```
max_check_attempts              5               ;(10)
check_command                   check-host-alive
notification_period             24x7            ;(workhours)
notification_interval           5               ;(120)
notification_options            d,u,r,f         ;(d,u,r)
contact_groups                  admins
register                        0
}
```

3.3 Zeitfensterobjekte

Die oben bereits erwähnten Zeitfenster- oder `timeperiod`-objekte `24x7` und `workhours` sind wie die Hostobjekt-Templates `linux-server` und `generic-host` bereits Teil der mitgelieferten Basiskonfiguration.[65] Zeitfensterobjekte beziehen sich stets auf eine Kalenderwoche, und bestehen aus einer Anzahl kleinerer Zeitfenster, die man durch Auflistung von Uhrzeiten an den einzelnen Wochentagen definiert.[66] Im Rahmen dieser Arbeit wurde darauf verzichtet, neue Objekte zu kreieren. Um dennoch einen Eindruck zu vermitteln, sei hier die Definition des einzig verwendeten Zeitfensterobjektes `24x7` wiedergegeben:

```
define timeperiod{
     timeperiod_name     24x7
     alias               24 Hours A Day, 7 Days A Week
     sunday              00:00-24:00
     monday              00:00-24:00
     tuesday             00:00-24:00
     wednesday           00:00-24:00
     thursday            00:00-24:00
     friday              00:00-24:00
     saturday            00:00-24:00
     }
```

3.4 Kommandoobjekte

Der Aufruf der Nagios-Plugins erfolgt über sogenannte Kommandoobjekte[67], mit ihrer Hilfe wird die Syntax des auszuführenden

65 Sie befinden sich in der Datei /etc/nagios/objects/timeperiods.cfg

66 Dies hat sich mit dem Erscheinen von Nagios 3.0 geändert, das wesentlich flexiblere Zeitfensterobjekte erlaubt. Ausführliche Informationen hierzu findet man in der Dokumentation unter `http://www.nagios.org/docs/`.

67 Auch alle Aufrufe externer Programme wie z.B. `mail` oder das Ablegen von Performancedaten werden über die

Befehls festgelegt. In Hostobjekten z.B. wird über das Attribut `check_command` auf ein solches Kommandoobjekt referenziert, im Beispielnetz ist es für alle vorhandenen Maschinen (ausgenommen `ServerB`) das gleiche Objekt `check-host-alive`:[68]

```
define command{
     command_name   check-host-alive
     command_line   $USER1$/check_ping -H $HOSTADDRESS$  →
                    -w 3000.0,80% -c 5000.0,100% -p 5
     }
```

Ein Kommandoobjekt besitzt immer genau die beiden Attribute `command_name` und `command_line`.[69] Über `command_name` definiert man den Bezeichner für das Kommandoobjekt. Über dessen Angabe im `check_command`-Teil von Hostobjekt (bzw. Serviceobjekt) führt das System den unter `command_line` definierten Befehl aus.[70]

`-H`, `-w`, `-c`, `-p` sind Optionen des verwendeten Plugins `check_ping`. Die Verwendung der Optionen `-w` und `-c` ist den meisten Plugins gemein, über sie übergibt man Schwellwerte, die einem Warnzustand bzw. kritischen Zustand der zu überprüfenden Ressource entsprechen. Im obigen Beispiel bedeuten die übergebenen Argumente, dass das Plugin mit einer Warnung reagieren soll, wenn die Antwortzeit 3ms[71] überschreitet oder wenigstens 80% der gesendeten ICMP-Pakete unbeantwortet bleiben, der Zustand soll hingegen als kritisch eingestuft werden, wenn die Antwortzeit 5ms überschreitet oder sämtliche ICMP-Pakete nicht beantwortet werden. Mit `-H` gibt man wahlweise eine Hostadresse oder einen Hostnamen an, mit `-p` wird eingestellt, wie viele ICMP-Pakete das Plugin `check_ping` versenden soll.

Die in Dollarzeichen eingeschlossen Bezeichner sind Makros, die das Editieren der Kommandoobjekt-Definitionen erleichtern. `$HOSTADDRESS$` wird

Kommandoobjekte realisiert.

68 Die Definition von `check-host-alive` stammt wiederum aus der Nagios-Beispielkonfiguration. Das Attribut `check_command` in der Hostobjekt-Definition von `ServerB` wurde im Beispiel in Kapitel 1 abgeändert. Der Host-Check erfolgt hier über das dort neu definierte Kommandoobjekt `CIMP`, das nicht das Plugin `check_ping`, sondern das eng verwandte `check_icmp` verwendet.

69 Es gibt noch zwei weitere Attribute, `template` und `name`, diese sind optional und stellen Mechanismen für die Vererbung zwischen Kommandoobjekten dar. Ein Beispiel für die Anwendung dieser Möglichkeit ist jedoch weder in der Beispielkonfiguration noch den verwendeten Quellen zu finden.

70 In der Nagios-Beispielkonfiguration und in den Beispielen in dieser Arbeit stimmen der Bezeichner des Kommandobjektes und der des verwendeten Plugins oftmals überein, was aber nicht vorgeschrieben ist.

71 `check_ping` erwartet die Zeitangaben in μs, im Gegesatz zu `check_icmp`, das mit ms arbeitet.

stets zur Netzwerkadresse derjenigen Komponente, für die das Kommando ausgeführt werden soll, expandiert. `$USER1$` wurde bei der Installation von Nagios automatisch gesetzt und enthält den Pfad zum Verzeichnis mit den Plugins. Eine wichtige Möglichkeit für die Verwendung von Makros ist die Übergabe von Argumenten an die Kommandoobjekte. Hierfür stehen die Makros `$ARG1$`, `$ARG2$`, ... zur Verfügung. Beispielsweise ist es denkbar, dass an einzelne Komponenten im Netz unterschiedliche Anforderungen hinsichtlich der Erreichbarkeit gestellt werden. Damit nicht für jede Komponente ein eigenes Kommandoobjekt erstellt werden muss, nutzt man den Mechanismus der Parameterübergabe. Auf diese Weise wird nur ein neues Kommandoobjekt benötigt, welches die Schwellwerte als Argumente erwartet:

```
define command{
     command_name   check-host-alive-2
     command_line   $USER1$/check_ping -H $HOSTADDRESS$ →
                    -w $ARG1$ -c $ARG2$ -p 5
     }
```

Soll z.B. für `RouterA` als zentralen Punkt im Beispielnetz kein Ausfall und keine Antwortzeit über 1ms toleriert werden, ändert sich der `check_command`-Teil im Hostobjekt wie folgt:

```
(alt) check_command  check-host-alive
(neu) check_command  check-host-alive-2!500.0,0%!1000.0,0%!
```

Die Ausrufungszeichen fungieren dabei wiederum als Trennzeichen zwischen den einzelnen Argumenten (vgl. Beispiel in 1.2.4), wobei `$ARG1$` stets den Wert zwischen dem ersten und zweiten Ausrufungszeichen annimmt, `$ARG2$` den zwischen zweitem und drittem usw.

3.5 Testen der Konfiguration

Findet sich ein Fehler in der Konfiguration, so lässt sich das System nicht mehr starten. Dann auftretende Fehlermeldungen können sich als schwer verständlich erweisen. Es empfiehlt sich daher, im Laufe des Aufbaus einer Konfiguration diese immer wieder zu testen, um so Fehler schneller

lokalisieren und korrigieren zu können. Hierfür bietet das Nagios-Hauptprogramm den Schalter -v (--verify):

```
Nagios:# /usr/local/nagios/bin/nagios -v /etc/nagios/nagios.cfg
```

Ist alles in Ordnung, kann man die neue Konfiguration laden:

```
Nagios:# /etc/init.d/nagios reload
```

3.6 Hostextinfo-Objekte

Die Auswirkungen des Anlegens der neuen, in Abschnitt 3.2 erstellten Hostobjekte auf die *Status Map* der Weboberfläche zeigt Abb.9.

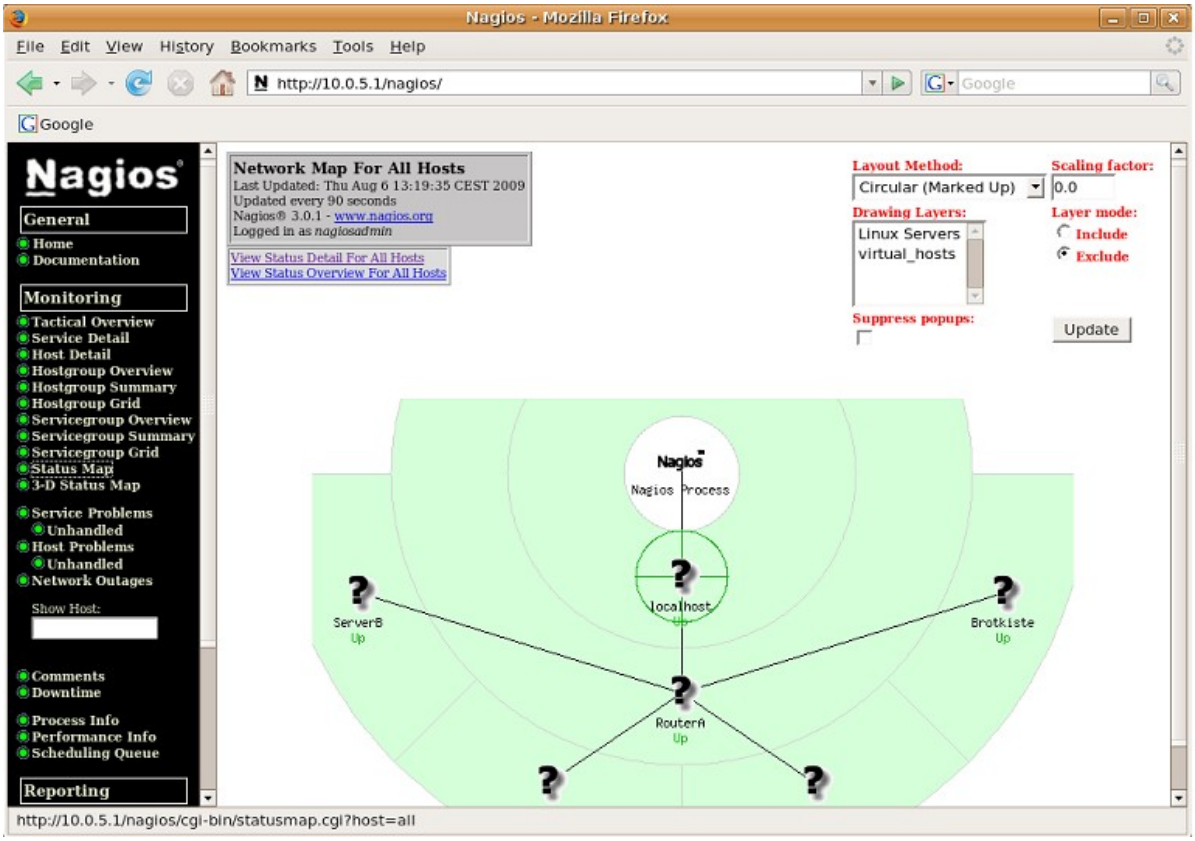

Abb.9: Nagios Status Map zum VNUML-Szenario

Sämtliche Geräte des Netzes sind mit einem Fragezeichen-Icon versehen. Außerdem sind die Geräte nicht logisch, sondern nach Ihrem „Abstand in hops“ zum Nagios-Server gruppiert, diese Gruppen befinden sich dann in konzentrischen Kreisen um den Rechner mit der Nagios-Installation.

Offensichtlich entspricht diese Darstellung des Netzes nur eingeschränkt der Vorstellung der Topologie, die man beim Betrachten von Abb.8 gewinnt. Um die Darstellung durch die Weboberfläche den Vorstellungen des Benutzers anpassen zu können, gibt es die sogenannten `hostextinfo`[72]-Objekte. Für jedes Hostobjekt kann ein solches angelegt werden, das Attribut `statusmap_image` definiert ein Icon zur Wiedergabe in der topologischen Karte und mit dem Attribut `2d_coords` legt man seine Koordinaten in der Karte fest. Als Beispiel das `hostextinfo`-Objekt von `ClientA`:

```
define hostextinfo{
      host_name            ClientA
      statusmap_image      client.gd2[73]
      2d_coords            100,150 [74]
      }
```

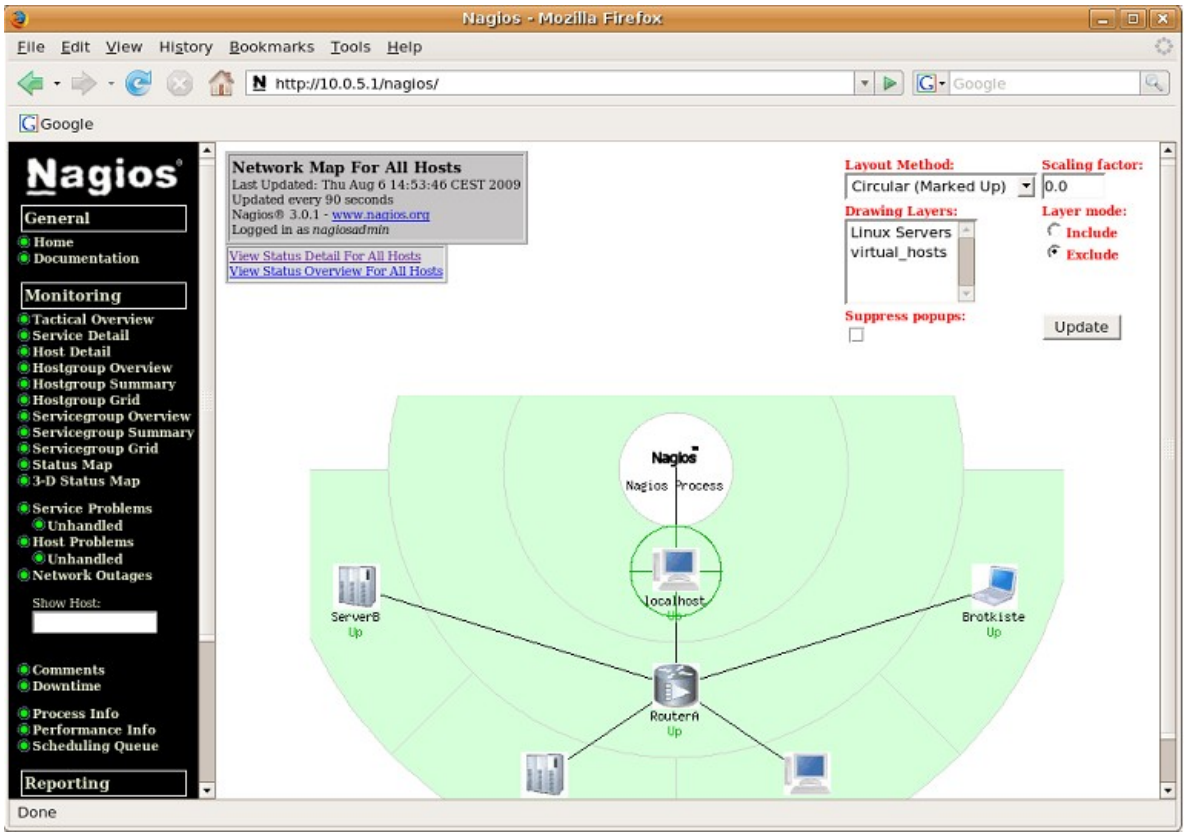

Abb.10: Nagios Status Map mit eigenen Icons

72 `hostextinfo` ist ein Kürzel für Extended Host Information. In `hostextinfo`-Objekten stehen neben den hier beschriebenen noch andere Attribute zur Verfügung, über welche den Hosts weitere Informationen zugeordnet werden können, vgl. (Barth 2009, S372ff).

73 Das Icon `client.gd2` wird im Verzeichnis `/usr/local/nagios/share/images/logos` erwartet. Ist die Datei nicht vorhanden, so wird weiterhin ein Fragezeichen angezeigt.

74 (100,150) bedeutet 100 Pixel rechts neben und 150 Pixel unterhalb der linken, oberen Bildecke.

Hat man für alle Hosts entsprechende Definitionen angelegt und ruft man (nach erneutem Laden der Nagios-Konfiguration) anschließend die *Status Map* auf, so sind zwar die Fragezeichen durch die neuen Icons ersetzt worden, das Layout der Karte ist aber noch immer das gleiche (Abb.10). Damit die Icons auch an ihren vorgesehen Koordinaten dargestellt werden, wählt man im Push-Down-Menü *Layout Method* am rechten oberen Bildende den Punkt *User supplied coords* (Abb.11). Um diesen Umweg nicht jedesmal gehen zu müssen, kann die Voreinstellung des Push-Down-Menüs in der Konfigurationsdatei für die Weboberfläche `/etc/nagios/cgi.cfg` angepasst werden. In dieser findet sich die Einstellung `default_statusmap_layout`. Voreingestellt ist 5 (entspricht `Circular (Marked Up)`), die richtige neue Einstellung ist 0 (entspricht `User supplied coords`).

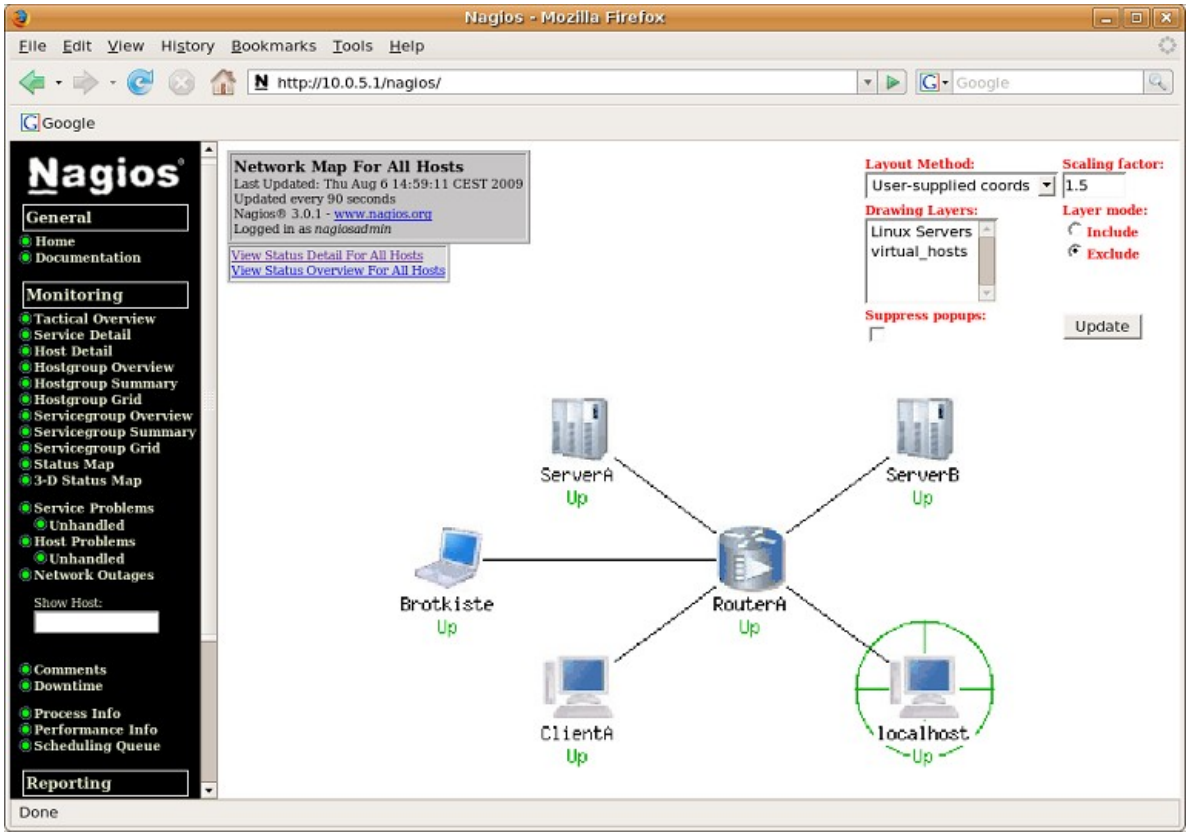

Abb.11: Nagios Status Map mit eigenen Icons und Koordinaten

3.7 Serviceobjekte

Die letzte wichtige Gruppe von Objekten in der Nagios-Konfiguration bilden die sogenannten Serviceobjekte. Die einzelnen Attribute von Serviceobjekt-Definitionen sind weitgehend die gleichen wie die von Hostobjekt-Definitionen. Die Nagios-Basiskonfiguration enthält wieder entsprechende Templates, `generic-service` und das davon abgeleitete `local-service`. Die Serviceobjekte aller Dienste in dieser Arbeit bedienen sich der Definition von `local-service` über eine entsprechende `use`-Anweisung. Die Timer (für eine schnellere Reaktion von Nagios auf Veränderungen bzgl. der Verfügbarkeit von Diensten) und die Nachrichtenfilter können analog zu den Ausführungen zu Hostobjekt-Definitionen in Abschnitt 3.2 angepasst werden. Im Folgenden soll sich daher auf die Beschreibung der wichtigsten Unterschiede zu Hostobjekten beschränkt werden. Dies geschieht am einfachsten anhand eines Beispiels:[75]

Überwacht werden soll der freie Speicherplatz auf der Root-Partition des Nagios-Servers `Nagios`. Für diesen Zweck stellt Nagios das Plugin `check_disk` zur Verfügung und in der Grundkonfiguration sind bereits ein entsprechendes Kommandoobjekt und Serviceobjekt definiert:

```
define command{
     command_name        check_local_disk
     command_line        $USER1$/check_disk -w $ARG1$   →
                         -c $ARG2$ -p $ARG3$
     }

define service{
     use                 local-service
     host_name           localhost
     service_description Root Partition
     check_command       check_local_disk!20%!10%!/
     }
```

Der `host_name` assoziiert das Serviceobjekt fest mit einem Host, hier `localhost`, also dem Nagios-Server selbst. Der Service besitzt, im Gegensatz zum Host, keinen eigenen (und eindeutigen) Namen, sondern lediglich ein Attribut `service_description`, die hier eingetragenen Informationen werden

75 Zahlreiche weitere Beispiele für Serviceobjekte folgen in Kapitel 4

in der Weboberfläche angezeigt und mit Ihnen werden ggf. Ausgaben in log-files und/oder Performancedaten gekennzeichnet. Auch das Attribut `parents` gibt es bei Serviceobjekten nicht. Die Interaktion mit dem unter `check_command` angegebenen Kommandoobjekt ist die gleiche wie bei den Hostobjekten.

Das Plugin `check_disk` erhält über die Parameter `-w` und `-c` Prozentangaben: Unterschreitet der freie Speicherplatz auf der über `-p` angegebenen Partition („/“, also die Root-Partition, hier als `$ARG3$` übergeben) diese Werte, reagiert es entsprechend mit Warn- bzw. Fehlermeldung.

`check_disk` und weitere in der Basiskonfiguration verwendete Plugins erhalten ihre Informationen über den Aufruf von Systembefehlen auf dem lokalen Rechner und können daher ausschließlich Überwachungsaufgaben auf dem Nagios-Server übernehmen. Wie mit Nagios entsprechende Ressourcen auch entfernter Geräte überwacht werden können, wird in Abschnitt 4.2.3 beschrieben.

4. Überwachung von Netzwerkdiensten

Im folgenden sollen verschiedene Netzwerkdienste auf den virtuellen Maschinen `ServerA` und `ServerB` des in Abschnitt 3.1 vorgestellten VNUML-Szenarios überwacht werden. Einige dieser Dienste, http, ssh und dns, laufen bereits standardmäßig auf den virtuellen Rechnern[76]. Die übrigen, mysql, ftp und ein mailserver, müssen erst noch installiert werden. Die für den Betrieb dieser Dienste benutzten Programme sind im Rahmen dieser Arbeit sämtlich über den Paketmanager installiert worden; Besonderheiten bei der Installation von Software auf die virtuellen Maschinen eines VNUML-Szenarios wurden in Abschnitt 2.1 ausführlich beschrieben und sind daher in den folgenden Abschnitten nicht mehr für einzelne Pakete gesondert dargelegt.

In den einzelnen Abschnitten wird jeweils einleitend kurz der Einsatzbereich des betreffenden Protokolls/Dienstes beschrieben. Falls notwendig, folgen Hinweise zu Installation und Konfiguration. Nach einer Beschreibung der von Nagios für den jeweiligen Dienst bereitgestellten Plugins bilden Beispiele für entsprechende Erweiterungen der Nagios-Konfiguration aus Kapitel 2 den Abschluss der einzelnen Abschnitte.

4.1 Hypertext Transfer Protocol (HTTP)

Das HTTP ist ein Protokoll der Anwendungsschicht. Die betreffenden Anwendungen sind zum einen *Browser*, die mittels HTTP Internetseiten im (X)HTML-Format[77] aus dem World Wide Web laden, auf Serverseite kommt dabei ein sogenannter *Webserver* zum Einsatz. Hiervon abweichende Verwendungen sind zwar denkbar, aber unüblich. Grundsätzlich ist es aber möglich, beliebige Daten mit dem HTTP zu übertragen.

76 Dies muss natürlich nicht zwingend bei allen Versionen der für VNUML verfügbaren Dateisystem-Images zutreffen.

77 (X)HTML = (Extensible) Hypertext Markup Language

4.1.1 Installation/Konfiguration

Apache[78] ist ein weit verbreiteter, für alle gängigen Betriebsysteme frei verfügbarer Webserver, der auf den virtuellen Maschinen bereits vorinstalliert ist. Eine Änderung der Konfiguration ist ebenfalls nicht notwendig. Um Apache auf der Maschine `ServerA` vom Host komfortabel starten zu können, kann man z.B. die Definition der Maschine `ServerA` in `nagios.xml` um ein sogenanntes *exec-Tag* erweitern:

```
<exec seq="starthttp" type="verbatim">
     /etc/init.d/apache2 start
</exec>
```

Der Befehl in der mittleren Zeile wird auf dem Rechner `ServerA` mittels folgendem Aufruf des VNUML-Parsers auf dem Host ausgeführt:

```
Host:# vnumlparser.pl -x starthttp@nagios.xml -vB
```

Um zu überprüfen, ob der Webserver wunschgemäß arbeitet, soll lediglich ein einfaches HTML-Beispieldokument angelegt werden.[79]

```
<html>
  <head>
    <title>Hallo Welt HTML</title>
  </head>
  <body>
    <h1>Hallo Welt!</h1>
    <p>Bin nur ein Beispieldokument</p>
    <p>ohne Informationsgehalt.</p>
  </body>
</html>
```

Damit der Webserver auf `ServerA` diese Seite standardmäßig darstellt, speichert man diese dort unter `/var/www/index.html` ab. Sie sollte anschließend bei Eingabe der IP-Adresse von `ServerA` in der Adressleiste des Browsers angezeigt werden.

78 Apache Software Foundation, `http://www.apache.org`

79 Das ist nicht unbedingt nötig: Normalerweise bringt eine Installation von Apache bereits ein Beispieldokument mit sich.

4.1.2 Überwachung

Die Überwachung des Webservers erfolgt mit Hilfe des Plugins `check_http`, welches sehr viele Optionen besitzt, mit denen sich seine Funktionalität für verschiedenste Zwecke anpassen lässt. Vielen Nagios-Plugins sind die folgenden *Standardoptionen* gemein, die hier daher gesondert aufgeführt werden sollen:

-h, --help	Anzeige von Hilfsinformationen
-v, --verbose	Ausgabe zusätzlicher Informationen
-V, --version	Anzeige der Plugin-Version
-H, --hostname	zur Angabe der IP-Adresse oder des Namens des zu überwachenden Zielrechners
-t, --timeout	zur Angabe eines Timeouts in Sekunden
-w, ––warning	zur Angabe der Warnschwelle
-c, --critical	zur Angabe der kritischen Schwelle
-4, --use-ipv4	zwingend IPv4 benutzen
-6, --use-ipv6	zwingend IPv6 benutzen

Dabei bedeuten im Falle von `check_http` die angebenen Schwellwerte die Antwortzeit des Servers in Sekunden. Nun zu den speziellen Optionen von `check_http`**:**

-I, --IP-address	wie -H, nur wird bei der Verwendung von `-H` zusätzlich der Hostname im HTTP-Header übertragen
-u, --url	URL oder relativer Pfad der gewünschten Seite
-p, --port	zur Angabe eines alternativen Ports
-L, --link-url	zur Ausgabe eines Links auf die mittels `-u` spezifizierte Seite in der Nagios-Weboberfläche
-a, --authorization	zur Angabe eines User-Passwort-Paares, falls der Zugriff eine Authentifikation erfordert
-f, --ondirect	zur Steuerung des Verhaltens des Plugins im Falle eines Redirects durch den Server

-e, --expect	Suchstring, der in der ersten Statuszeile der zurückgelieferten Seite erwartet wird
-s, --string	Suchstring, der in der zurückgelieferten Seite erwartet wird
-r, --regex	wie vor, aber auf einen regulären Ausdruck
-R, --regexp	wie vor, jedoch ohne Unterscheidung von Groß- und Kleinschreibung
-P, --post	zur Datenübermittlung mittels POST-Befehl an den Webserver
-m, --pagesize	zum Überprüfen der Größe der zurückgelieferten Seite
-N, --no-body	Beschränkung der Auswertungen auf die Header-Daten der zurückgelieferten Seite
-M, --max-age	Kontrolle des Alters der zurückgelieferten Seite
-A, --useragent	zur expliziten Angabe eines User-Agents im HTTP-Header
-k, --header	zur Angabe von HTTP-Header-Tags
-S, --ssl	Verwendung einer SSL-Verbindung
-C, --certificate	Überprüfung des Zertifikats der zurückgelieferten Seite

Wirkt die Vielzahl der Möglichkeiten auch verwirrend, so wird man in der Praxis meist nur einige der Optionen in einer einzelnen Definition kombinieren. Es folgen Beispiele für ein Service- und ein Kommandoobjekt, mit deren Aufnahme in die Nagios-Konfiguration eine Überwachung des Webservers `Apache2` auf `ServerA` erreicht wird, wobei nur wenige der oben angegebenen Optionen tatsächlich benötigt werden:

```
define service{
     use                  local-service
     host_name            ServerA
     service_description HTTP
     check_command        check_http!/index.html!3!1!hallo
     }

define command{
```

```
command_name   check_http
command_line   $USER1$/check_http -H $HOSTADDRESS$  →
               -c $ARG2$ -w $ARG3$ -u $ARG1$ -R $ARG4$
}
```

Der hier verwendete Aufruf von `check_http` erzeugt eine Warnung, sobald die Antwortzeit eine Sekunde überschreitet bzw. einen kritischen Rückgabewert, wenn mehr als drei Sekunden benötigt werden. Um sicherzustellen, dass es sich auch um die erwartete Seite handelt, wird mit der Option `-R` zusätzlich ein Textvergleich bewirkt.

4.2 Secure Shell-Protokoll (SSH)

Das Secure Shell-Protokoll ermöglicht den Aufbau sicherer Verbindungen zwischen Rechnern in Computernetzen. SSH ist heute weit verbreitet, es gibt Implementationen für alle gängigen Betriebssysteme und es hat weniger sichere, vormals populäre Programme wie z.B. `rlogin`, `telnet`, `rcp` etc. aus der Praxis (zumindest in öffentlichen Netzen) weitgehend verdrängt. Die gegenüber diesen Programmen erhöhte Sicherheit erklärt sich durch die Verwendung einer RSA-Verschlüsselung bei der Authentifizierung.

4.2.1 Installation/Konfiguration

Auf den Rechnern des VNUML-Szenarios ist bereits eine SSH-Suite[80] installiert, und auch auf dem Hostrechner selbst sollte eine vorhanden sein.[81] Eine Konfiguration ist ebenfalls nicht nötig - der Server-Daemon `sshd` ist nach dem Start des Szenarios auf allen Maschinen aktiv, der Verbindungsaufbau zwischen zwei virtuellen Maschinen, wie er ja getestet werden soll, ist also bereits möglich.

4.2.2 Überwachung

Für die Überwachung von SSH bietet Nagios das Plugin `check_ssh` an,

80 OpenSSH, www.openssh.com/de

81 Denn diese ist schon zum Betrieb von VNUML notwendig.

welches nur den SSH-Handshake auswertet, d.h. es wird keine Authentisierung und auch kein Benutzerkonto `nagios` auf dem Server benötigt. Das Plugin kann also verwendet werden, auch wenn dem Benutzer `nagios` kein Zugriff auf den Server gestattet ist. (Im Beispielnetz ist das Benutzerkonto natürlich dennoch vorhanden.) Neben den Standardoptionen `-H`,`-4`,`-6` und `-t` kennt `check_ssh` noch die folgenden:

`-p, --port`	zur Angabe eines alternativen Ports
`-r, --remote-version`	zum Überprüfen der SSH-Version[82]

In die Nagios-Konfiguration werden folgende einfache Kommando- bzw. Servicedefinitionen aufgenommen, um den SSH-Service auf `ServerA`[83] zu überwachen:

```
define command{
     command_name        check_ssh
     command_line        $USER1$/check_ssh -H $HOSTADDRESS$
     }

define service{
     use                 local-service
     host_name           ServerA
     service_description SSH
     check_command       check_ssh
     }
```

4.2.3 Überwachung lokaler Ressourcen

Ein Anwendungsbeispiel für das SSH-Protokoll im Zusammenhang mit der Überwachung von Netzwerken durch Nagios ist die Möglichkeit, lokale Ressourcen wie Festplattenplatz, CPU-Auslastung usw. auf entfernten Rechnern im Netz zu überprüfen, indem dort über eine SSH-Verbindung gestartete Plugins zur Ausführung gebracht werden. Hierfür steht das Plugin `check_by_ssh` zur Verfügung, welches die Verbindung aufbaut, um anschließend ein Kommando an den entfernten Rechner zu übergeben,

82 Bei Verwendung dieser Option erhält man von Nagios eine Warnung, falls der angegebene Wert nicht mit der Version auf dem Server übereinstimmt. Die aktuelle Version auf dem Server kann man dort z.B. mittels `ssh -V` abfragen.

83 `ServerA` wurde hier beliebig gewählt, man hätte genauso jeden anderen virtuellen Rechner verwenden können.

welches dort ausgeführt wird. Dazu muss aber der entsprechende Befehl auf diesem Rechner erst einmal vorhanden sein - im Falle des VNUML-Beispiels stellt dies keinen zusätzlichen Aufwand dar, da die Nagios-Plugins auf allen Maschinen zur Verfügung stehen. Lediglich auf dem Hostrechner selbst müsste im Bedarfsfall noch nachgebessert werden, am einfachsten indem man die gewünschten (fertig kompilierten) Plugins hierher kopiert. Hier sollen aber nur lokale Ressourcen der Maschine `ClientA` überwacht werden.

Damit das Plugin `check_by_ssh` eine SSH-Verbindung zum virtuellen Rechner aufbauen kann, ohne dass eine Passwortabfrage erfolgt, wird eine *Public-Key-Authentifizierung* verwendet. Dabei wird auf Serverseite der öffentliche Schlüssel und auf Clientseite der zugehörige geheime Schlüssel benötigt. Das Schlüsselpaar erzeugt man mithilfe des Programms `ssh-keygen`, das in OpenSSH enthalten ist und speichert die Schlüssel im Unterverzeichnis `.ssh` des Homeverzeichnisses ab[84],der öffentliche Schlüssel gehört dabei in die Datei `authorized_keys`, der geheime Schlüssel erhält von ssh-keygen bereits den endgültigen Dateinamen `id_rsa`:[85]

```
nagios@Nagios:> ssh-keygen -t rsa -N ""
nagios@Nagios:> su
Nagios:# scp /usr/local/nagios/.ssh/id_rsa_pub    →
         10.0.4.1:/usr/local/nagios/.ssh/authorized_keys[86]
```

Das Plugin `check_by_ssh` verfügt über die üblichen Standardoptionen, wobei die mittels `-w` bzw. `-c` übergegebenen Werte Antwortzeiten in Sekunden bedeuten. Daneben existiert eine ganze Reihe weiterer Optionen, in den untenstehenden Beispielen wird nur noch folgende verwendet:

`-C, --command` zur Angabe eines auf dem entfernten Rechner auszuführenden Plugin-Aufrufs

Als Beispiele für lokale Ressourcen auf den VNMUL-Maschinen sollen

84 `ssh-keygen` speichert die Schlüssel automatisch hier ab, das Verzeichnis `.ssh` selbst legt es, falls noch nicht vorhanden, ebenfalls an.

85 vgl. (Franken 2010)

86 Anstelle des Kopierens sollte man die Datei einfach umbenennen, wenn man nach den Änderungen `uml_moo` ausführen möchte.

der Festplattenspeicher sowie die Systemlast auf `ClientA` überwacht werden, hierfür stehen die Plugins `check_disk` und `check_load` zur Verfügung. Beide sind bereits in der Beispielkonfiguration aktiv, wo sie für die Überprüfung entsprechender Ressourcen des Hostrechners eingesetzt werden. Eine Warnung soll erfolgen, wenn der freie Festplattenplatz auf `ClientA` 250MB unterschreitet bzw. die durchnittliche Anzahl gleichzeitig aktiver Prozesse in der letzten Minute (den letzten 5, den letzten 15 Minuten) auf ClientB größer als 3 (4,4) war; die kritischen Schwellwerte seien 50MB bzw. 5 (8,8) gleichzeitig aktive Prozesse. Die entsprechenden Aufrufe (und mögliche Antworten) der beiden Plugins lauten dann:

```
nagios@ClientA:> cd /usr/local/nagios/libexec
nagios@ClientA:> ./check_disk -w 250 -c 50 -p /[87]
DISK WARNING - free space: / 216 MB (30% inode=70%);|  →
/=484MB;488;688;0;738

nagios@ClientB:> ./check_load -w 3.0,4.0,4.0 -c 5.0,8.0,8.0
OK - load average: 0.00, 0.00, 0.00|                  →
load1=0.000;3.000;5.000;0;                            →
load5=0.000;4.000;8.000;0;                            →
load15=0.000;4.000;8.000;0;
```

Beide Aufrufe sollten mithilfe des Plugins `check_by_ssh` auch vom Nagios-Server gestartet werden können:

```
nagios@Nagios:> cd /usr/local/nagios/libexec
nagios@Nagios:> ./check_by_ssh -H 10.0.4.1            →
                -C "~/libexec/check_disk -w 250 -c 50 -p /"
DISK WARNING - free space: / 216 MB (30% inode=70%);|  →
/=484MB;488;688;0;738


nagios@Nagios:> ./check_by_ssh -H 10.0.4.1            →
                -C "~/libexec/check_load              →
                -w 3.0,4.0,4.0 -c 5.0,8.0,8.0"
```

87 Über -p wird der Pfad des zu prüfenden Gerätes angegeben, im Beispiel wird also der Speicherplatz auf der Root-Partition gemessen.

```
OK - load average: 0.00, 0.00, 0.00|                    →
load1=0.000;3.000;5.000;0;                              →
load5=0.000;4.000;8.000;0;                              →
load15=0.000;4.000;8.000;0;
```

Die Ausgaben sind die gleichen wie oben, d.h. `check_by_ssh` liefert die Ausgabe des entfernt aufgerufenen Plugins unverändert zurück (sofern dieser Aufruf erfolgreich war). Sind die oben beschriebenen Installationen und Tests erfolgreich abgeschlossen, können nach dem bereits bekannten Schema Service- und Kommandoobjekt-Definitionen erstellt werden:

```
# Kommandoobjekt für "check_disk per SSH"
define command{
     command_name   check_ssh_disk
     command_line   $USER1$/check_by_ssh -1              →
                    -H $HOSTADDRESS$                     →
                    -C "$USER1$/check_disk               →
                    -w $ARG1$ -c $ARG2$ -p /"
     }

# Kommandoobjekt für "check_load per SSH"
define command{
     command_name   check_ssh_load
     command_line   $USER1$/check_by_ssh -1              →
                    -H $HOSTADDRESS$                     →
                    -C "$USER1$/check_load               →
                    -w $ARG1$ -c $ARG2$"
     }

# Serviceobjekt Speicherplatz Root-Partition auf ClientA
define service{
     use                 local-service
     host_name           ClientA
     service_description Root Partition
     check_command       check_ssh_disk!250!50
     }

# Serviceobjekt Systemauslastung auf ClientA
define service{
     use                 local-service
     host_name           ClientA
     service_description Local Load
     check_command       check_ssh_load!3.0,4.0,4.0!     →
                         5.0,8.0,8.0
     }
```

Neben der Verwendung des Plugins `check_by_ssh` gibt es noch weitere Möglichkeiten, mit Nagios Informationen über entfernte Rechner zu gewinnen: Mit dem Einsatz des Nagios Remote Plugin Executor (NRPE), der zusätzlich zu den verwendeten Plugins auf dem zu überwachenden Rechner installiert werden muss, kann man die Möglichkeiten des Users `nagios` auf diesem einschränken[88]. Der Nagios Service Check Acceptor (NSCA) läuft auf dem Nagios-Server und wartet passiv auf eingehende Ergebnisse extern ausgeführter Tests. Da auf dem entfernten Rechner also ebenfalls eine Nagios-Instanz vorhanden sein muss, wird der NSCA normalerweise nur beim verteilten Monitoring eingesetzt, außerdem können auf diese Weise keine Informationen über Rechner ohne eigene Nagios-Instanz gewonnen werden. Schließlich gibt es Plugins, die über das Simple Network Management Protocol (SNMP) direkt Informationen über entfernte Rechner ermitteln können, sofern auf diesen ein SNMP-Daemon läuft. Durch die Verbreitung von SNMP ergibt sich der Vorteil, auf diese Weise auch Zugriff auf Ressourcen von "Nicht-Linux-Rechnern" erhalten zu können.[89]

4.3 File Transfer Protocol (FTP)

Das FTP ist ein Protokoll der Anwendungsschicht und ist, wie der Name sagt, für die Übertragung von Dateien[90] konzipiert. Ferner können Verzeichnisse angelegt und gelöscht werden, auch das Umbenennen von Dateien/Verzeichnissen oder das Ändern der Zugriffsberechtigungen ist möglich, kurz: Durch die Verwendung eines ftp-client-Programms unterscheidet sich der Umgang mit Dateien auf einem entfernten Server für den Benutzer äußerlich wenig vom Umgang mit Dateien im lokalen, eigenen Dateisystem[91].

Der überwiegende Teil der im Internet vorgenommenen Übertragungen erfolgt dabei über das sogenannte Anonymous-FTP (anonymes FTP), hier wird auf eine Benutzerauthentifizierung verzichtet, die entsprechenden Dateien

88 Denn über die SSH-Verbindung kann dieser beliebige Befehle auf dem entfernten Rechner ausführen.
89 Das geht natürlich auch ohne SNMP, wenn man Plugins für andere Betriebssysteme selbst erstellt.
90 Meist erfolgt der Datenaustausch zwischen einem Server im Internet und dem lokalen Rechner.
91 Dabei kann man eine Verbindung über eine Konsole herstellen oder auch einen FTP-Client mit grafischer Benutzeroberfläche verwenden.

stehen also jedermann zum Herunterladen zur Verfügung. Soll den Benutzern auch das Hochladen von Dateien erlaubt oder soll der Zugriff der Benutzer auf gewisse, z.B. private Dateien eingeschränkt werden, arbeitet man meist mit Zugangsberechtigungen; der Benutzer muss sich dann zu Beginn der Sitzung mit Benutzername und Passwort beim Server authentisieren.

4.3.1 Installation/Konfiguration

Um auf `ServerA` einen FTP-Server aufzusetzen, installiert man das Paket `proftpd`[92]:

```
ServerA:# apt-get install proftpd
```

Nach der Installation befindet sich der Pro-FTP-daemon bereits in Ausführung, und auch nach dem Neustart des Szenarios wird er wieder automatisch gestartet.[93] Damit steht einer Überwachung durch Nagios eigentlich nichts mehr im Wege, trotzdem soll kurz auf die Inbetriebnahme des `proftpd` eingegangen werden.

Während des Installationsvorgangs erfolgt ggf. die Abfrage, ob ein Zugriff zum Server durch anonyme Benutzer erlaubt werden soll, hier sei dies der Einfachheit wegen erwünscht.[94] Dieser Zugriff beschränkt sich auf Dateien im Verzeichnis `/home/ftp`. Daten, die man über Anonymous-FTP allen Nutzern zur Verfügung stellen möchte, gehören also hierher. Für Testzwecke genügt eine leere Datei:

```
ServerA:# touch /home/ftp/test.txt
```

Um sich zu vergewissern, dass alles ordnungsgemäß funktioniert, kann man die Datei z.B. von `ClientA` aus via ftp „herunterladen“.[95] Eine

92 http://www.proftpd.org, Pro-FTP ist ein Open Source-Projekt und heute weit verbreitet. Es wurde als Verbesserung des vormals oft eingesetzten WU-FTP (WU=Washington University) entwickelt und bietet im Vergleich zum „Vorgänger“ vor allem flexiblere Konfigurationsmöglichkeiten.

93 Der Start von `proftpd` schlägt fehl, wenn das System die eigene Netzwerkadresse nicht ermitteln kann, man erhält dann eine Fehlermeldung der Art „`proftpd – warning: unable to determine IP-address...`“. Am einfachsten beseitigt man das Problem durch einen entsprechenden Eintrag in der Datei `/etc/hosts`: „`10.0.2.1 ServerA`“. Ist ein DNS-Server (wie in 4.4 beschrieben) in Betrieb, tritt dieses Problem nicht auf.

94 Bleibt diese Abfrage aus, so kann man durch Einkommentieren des Abschnitts `<anonymous>` in der Konfigurationsdatei `/etc/proftpd/proftpd.conf` diese Funktionalität hinzufügen.

95 Unter Umständen ist der Zugriff von Außen durch Einstellungen in der Datei `/etc/hosts.deny` noch blockiert.

entsprechende ftp-Sitzung könnte in etwa folgendermaßen ablaufen (Benutzereingaben sind blau hervorgehoben):

```
ClientA:~# ftp 10.0.2.1
Connected to 10.0.2.1.
220 ProFTPD 1.3.3a Server (Debian) [10.0.2.1]
Name (10.0.2.1:root): anonymous
331 Anonymous login ok, send your complete email address as
your password
Password: some-email-address
230-Welcome, archive user anonymous@10.0.4.1 !
230-
230-The local time is: Fri Jan 21 14:46:42 2011
230-
230-This is an experimental FTP server.  If you have any
unusual problems,
230-please report them via e-mail to <root@ServerA>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      ftp             0 Jan 21 12:26
test.txt
-rw-r--r--   1 ftp      ftp           170 Nov  3 14:26
welcome.msg
226 Transfer complete
ftp> get test.txt
local: test.txt remote: test.txt
200 PORT command successful
150 Opening BINARY mode data connection for test.txt
226 Transfer complete
ftp> bye
221 Goodbye.
ClientA:~# ls test.*
test.txt
```

Die Datei kann auch einfacher mithilfe eines Browsers heruntergeladen werden. Dazu gibt man auf dem Hostrechner in die Adressleiste des Browsers `ftp://10.0.2.1` ein (siehe Abb.12). Eine genaue Beschreibung der Konfigurationsmöglichkeiten des `proftpd`, wie etwa der Verwaltung von Lese- und Schreibrechten oder des Speicherplatzes auf dem Server etc., findet man z.B. in (Lowes 2001).

Dies behebt man leicht durch die Aufnahme einer Zeile „`ftpd: ALL`“ in die Datei `/etc/hosts.allow`.

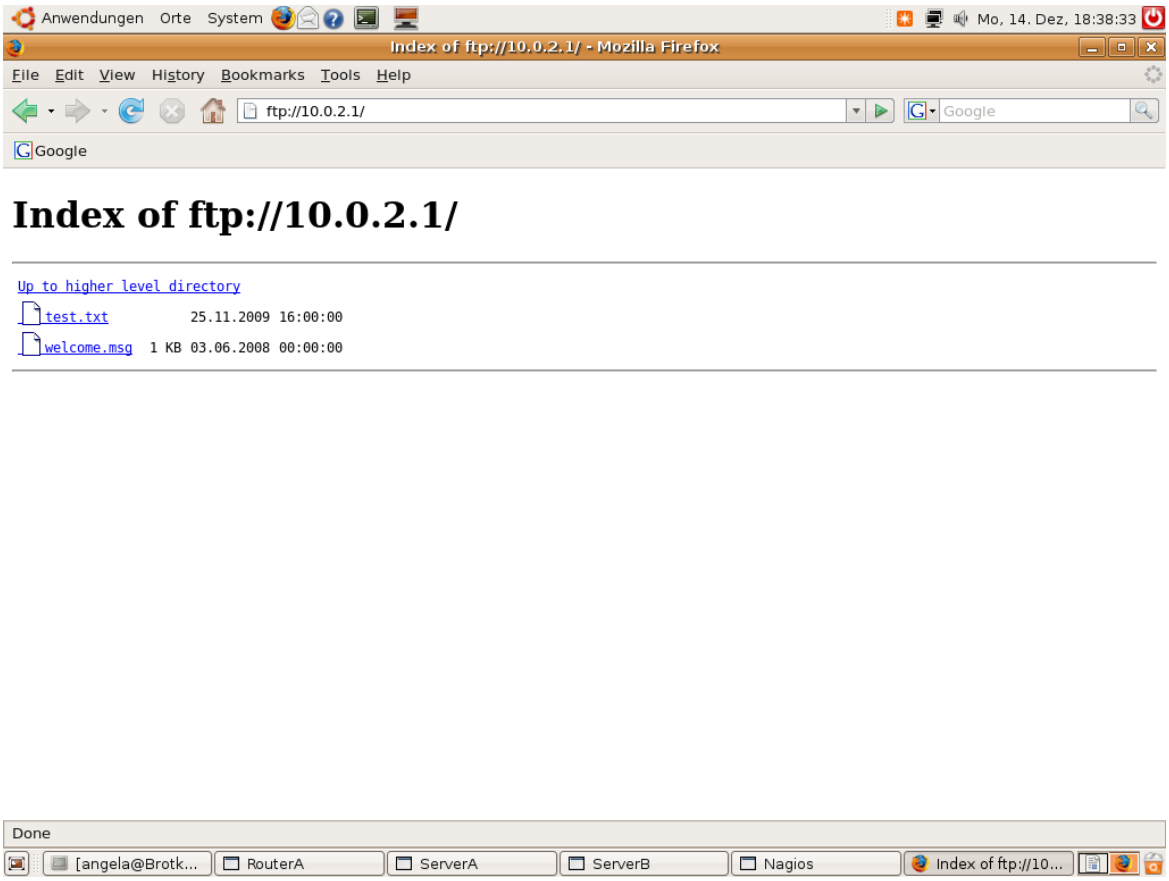

Abb.12 Zugriff auf die ftp-Testdatei im Browserfenster

4.3.2 Überwachung

Nagios bietet hierfür das Plugin `check_ftp`. Es findet dabei kein Login zum Server statt, vielmehr ähnelt die Funktionalität von `check_ftp` einem einfachen Portscan. Es handelt sich nicht um ein eigenständiges Plugin, sondern es wird das generische Plugin `check_tcp`[96] verwendet, wobei durch den Aufruf über `check_ftp` folgende Parameter automatisch gesetzt werden:

`-p, --port`	wird auf den ftp-Standardport 21 gesetzt
`-e, --expect`	wird auf „`220`“ gesetzt[97]
`-q, --quit`	wird auf „`QUIT\r\n`“ gesetzt

Den mit `-e` angegebenen Suchstring erwartet das Plugin in der Antwort

96 `check_tcp` und `check_udp` sind generische Netzwerkplugins, die immer dann eingesetzt werden können, wenn für einen bestimmten Dienst kein dediziertes Plugin zur Verfügung steht. Eine komplette Liste der verfügbaren Optionen beider Plugins findet man in (Barth 2009, S133ff).

97 Die 220 ist der ftp-Antwortcode für „Service ready for new user“

des Servers, mit dem über `-q` definierten Befehl wird die Verbindung ordnungsgemäß geschlossen. Die Optionen `-w` und `-c` erwarten bei `check_ftp` wieder Antwortzeiten in Sekunden, über `-H` gibt man wieder die Netzwerkadresse des Servers an. Es ergeben sich folgende Nagios-Objektdefinitionen für die Überwachung des FTP-Dienstes auf `ServerA`:

```
define command{
     command_name   check_ftp
     command_line   $USER1$/check_ftp -H $HOSTADDRESS$ →
                    -w $ARG1$ -c $ARG2$
     }

define service{
     use                        local-service
     host_name                  ServerA
     service_description        FTP
     check_command              check_ftp$0.5$1
     }
```

4.4 Domain Name System (DNS)

Als Domain Name System bezeichnet man eine verteilte Datenbank im Internet, durch die sogenannte Domänennamen (z.B. `www.google.de`) auf die entsprechenden Internetadressen (z.B. `74.125.39.99`) abgebildet werden. Das DNS ist hauptsächlich als eine Erleichterung für den Menschen beim Umgang mit dem Internet gedacht, da man sich Namen wie `www.google.de` wesentlich leichter merken kann als eine Fülle von Internetadressen.[98]

Wird ein Name an den Server gesendet, der (hoffentlich) die zugehörige Adresse zurückliefert, spricht man von einem *forward lookup*, im umgekehrten Fall von einem *reverse lookup*, welcher eine eher untergeordnete Rolle spielt. Ein typisches Beispiel für einen forward lookup ergibt sich, sobald man eine Internetseite anhand ihres Domänennamens über die Adressleiste des Browsers aufruft. Dadurch wird ein DNS-Client-Programm gestartet. Dieses schickt eine DNS-Anfrage an den Nameserver[99], und die gewünschte Seite wird im Browser mithilfe der vom Nameserver zurückgelieferten Internetadresse geöffnet. Möchte man hingegen in

98 vgl. (Baader 2003)

99 Ein DNS-Server wird allgemein kurz als Nameserver bezeichnet.

Erfahrung bringen, was sich hinter einer bestimmten IP-Adresse verbirgt, hilft ein reverse lookup.

Das DNS im Internet ist hierarchisch organisiert: Hat der angesprochene Nameserver die gewünschte Information nicht selbst griffbereit, leitet er die Anfrage an einen anderen, zuständigen Server weiter.[100] Der Namensraum ist dabei in sogenannte *Domänen* (engl. domains) oder auch *Zonen* unterteilt.

4.4.1 Installation/Konfiguration

Auf den UML-Maschinen ist der DNS-Server BIND[101] installiert und dieser wird auch nach dem Starten eines Szenarios (auf allen Maschinen) bereits ausgeführt. Als Resolver-Werkzeuge[102] stehen die Programme `nslookup` und `dig`[103] zur Verfügung, mit ihnen kann überprüft werden, ob das DNS wunschgemäß funktioniert. Eine Installation zusätzlicher Software ist also nicht nötig.

Der BIND-Daemon `named` fungiert zunächst aber als reiner DNS-Caching-Server (auch als Caching-Only Nameserver bezeichnet), d.h. es sind auf den virtuellen Maschinen zunächst keinerlei DNS-Informationen vorhanden, und BIND sammelt lediglich Adress/Namenpaare, die durch DNS-Anfragen an Nameserver im Internet bekannt werden. BIND soll gemäß folgender Funktionalität umkonfiguriert werden:

- An die Stelle der verschiedenen Caching Only Nameserver rückt ein einzelner Nameserver auf `ServerA`.
- Die übrigen Maschinen sollen sich `ServerA` als Nameserver bedienen.
- `ServerA` soll die Namensinformationen aus dem lokalen Netz bereitstellen und Anfragen nach externen Adressen bzw.

100 Dies bedeutet zunächst nur eine Verteilung der Daten und noch keine Hierarchie. Auf Details der tatsächlichen Organisation des DNS wird hier verzichtet, da es im Beispielnetz nur einen Nameserver geben soll und somit Mechanismen wie z.B. Delegierung oder Zonentransfer nicht benötigt werden.

101 Berkeley Internet Name Domain. `BIND` ist freie Software und kann vom Internet Systems Consortium unter `www.isc.org/software/bind/` bezogen werden.

102 Resolver-Werkzeug wird hier synonym zum Begriff des DNS-Clientprogramms verwendet. Als Resolver bezeichnet man allgemein diejenigen Komponenten der Systembibliothek, die für die Namensauflösung gebraucht werden, vgl. [Kirch/Dawson, S.92].

103 Domain Information Groper

Dömanennamen an einen Nameserver im Internet weiterleiten.

- Auch reverse lookups sollen möglich sein.
- Der Einfachheit halber wird auf einen Secondary Nameserver verzichtet.

Der erste Punkt bedeutet, dass einfach der BIND-Daemon `named` auf allen Maschinen, mit Ausnahme von `ServerA`, gestoppt werden muss: (am Beispiel von `RouterA`)

```
RouterA:# /etc/init.d/bind9 stop
```

Für den zweiten Punkt ist die Adresse von `ServerA` als Nameserver in der Datei `/etc/resolv.conf`[104] aller virtuellen Maschinen einzutragen. Dort schon vorhandene Nameserveradressen sind zu löschen oder auszukommentieren, damit nicht im Fehlerfall auf andere Nameserver zurückgegriffen wird. Außerdem ist eine `search`-Anweisung sinnvoll; der Resolver hängt die in der hier angegebenen Suchliste vorhandenen Domänennamen an den aufzulösenden Namen an, falls die Namensauflösung nicht ohnehin erfolgreich durchgeführt werden kann, und startet eine neue Anfrage. Die Datei `/etc/resolv.conf` der virtuellen Maschinen könnte demnach etwa folgendermaßen aussehen:

```
# Put the right IP address for DNS server
# nameserver 80.58.61.250
# nameserver 80.58.61.254
search nagios
nameserver 10.0.2.1[105]
```

Dabei ist `nagios` einfach der weiter unten (Punkt drei) gewählte Name der die UML-Maschinen und den Hostrechner umfassenden Zone, die von `ServerA` autoritativ verwaltet werden soll. Durch die Verwendung der `search`-Anweisung kann die Adresse von z.B. `RouterA` aus eben diesem Namen

104 Die Datei `/etc/resolv.conf` wird vom Resolver eingelesen, enthält also Informationen zur Namensauflösung für den DNS-Client

105 Für den NS ServerA selbst empfiehlt sich zusätzlich die Angabe einer gültigen NS-Adresse aus dem Internet, um auch Adressinformationen von außerhalb des virtuellen Netzes bereitstellen zu können. An diesem Punkt werden also erstmals dauerhaft unterschiedliche Dateien auf den einzelnen virtuellen Maschinen benötigt. Damit man weiterhin mit nur einem Dateisystem-Image arbeiten kann, können die abweichenden Dateien z.B. beim Start des DNS mittels exec-Tag an die gewünschten Orte kopiert werden.

ermittelt werden, ohne die Anweisung müsste man immer den vollen Namen `RouterA.nagios` angeben.

Damit Anfragen, für die `ServerA` nicht autoritativ zuständig ist, weitergeleitet werden, verwendet man eine `forward`-Anweisung in der BIND-Konfiguration. Die Adresse für den betreffenden Nameserver definiert man in einer zugehörigen `forwarders`-Klausel. Außerdem ist es notwendig, mithilfe einer `allow-recursion`-Anweisung rekursive Namensauflösung für alle Rechner im Netz zu erlauben, sonst funktioniert dies per default nur auf dem Nameserver selbst und den direkt verbundenen Maschinen, im Beispielnetz also `ServerA` und `RouterA`. Alle diese Einstellungen werden in der Datei `/etc/bind/named.conf.options` vorgenommen, die so folgendes Gesicht erhält:

```
options {
     directory "/var/cache/bind";
     recursion yes;
     allow-query{10.0/16;};
     allow-recursion{10.0/16;};

     forward first;
     forwarders{
          192.168.2.1;
     };
}; [106]
```

Durch Einstellungen in der Datei `/etc/host.conf` bzw. `/etc/nsswitch.conf` bedingt, wird nicht nur `ServerA` bei der Namensauflösung zu Rate gezogen, sondern es wird auch auf Adress-Namenpaare in der Datei `/etc/hosts` zurückgegriffen. Um auch hier sicherzugehen, dass die Antworten auf DNS-Anfragen tatsächlich vom Nameserver stammen, empfiehlt es sich, die betreffenden Informationen aus `/etc/hosts` zu entfernen oder auszukommentieren.

Für Punkt drei ist das Anlegen einer sogenannten Zonendatei auf `ServerA` erforderlich, welche die Adressinformationen des lokalen Netzwerks enthält. Als Zonenname werde `nagios` gewählt, damit ergibt sich `db.nagios`

106 Die hier unter `forwarders` eingetragene Adresse ist zunächst nur die lokale Adresse des Routers, über den eine Verbindung zum Internet besteht. Dieser leitet die Anfragen dann ggf. an einen Nameserver im Internet weiter.

als Dateiname[107]. Die Datei gehört ins Verzeichnis `/etc/bind/` und sieht wie folgt aus:

```
$TTL 604800
@         IN    SOA  ServerA.nagios.     root.localhost. (
                     2         ; Serial
                     604800    ; Refresh
                     86400     ; Retry
                     2419200   ; Expire
                     604800 )  ; Negative Cache TTL
;
@               NS   ServerA.nagios.
Host            A    10.0.1.2
RouterA         A    10.0.1.1
ServerA         A    10.0.2.1
ServerB         A    10.0.3.1
ClientA         A    10.0.4.1
Nagios          A    10.0.5.1
```

Die Zonendatei besteht aus einer Liste sogenannter Resource Records. Obiges Beispiel enthält einen SOA-Record (Start Of Authority - Record), einen NS-Record (Nameserver - Record) und sechs A-Records (Address – Record). In der ersten Zeile ist zusätzlich eine Zeitangabe vorhanden, wie lange die Informationen aus der Zonendatei auf entfernten Rechnern/Servern Gültigkeit besitzen sollen. Der SOA-Record gibt den primären Nameserver (`ServerA.nagios`) und die email-Adresse des Administrators (`root.localhost`, zu lesen als `root@localhost` – der Klammeraffe ist innerhalb der Datei bereits für andere Zwecke reserviert) für die Zone `nagios` (repräsentiert durch das `@`, welches durch die Angabe `zone “nagios“` in der Datei `named.conf` mit der Zeichenkette `nagios` assoziiert ist, siehe unten) an. Daneben enthält er verschiedene Zeitangaben zum Vorhalten von Adressinformationen und zur Steuerung von Zonentransfers, welche wegen des Verzichts auf Secondary Nameserver jedoch nicht stattfinden. Der NS-Record verbindet nochmals den vollen Namen des Nameservers mit dem der Zone, die A-Records enthalten jeweils ein Adress/Namenpaar. Die Zonendatei wird BIND über einen Eintrag in der Konfigurationsdatei `/etc/bind/named.conf.local` bekanntgemacht:

107 Die Zonendateien werden in Debian-Systemen üblicherweise nach dem Muster db.*zonenname* benannt, im Prinzip sind aber beliebige Namen erlaubt.

```
zone “nagios“ {
     notify no
     type master
     file “/etc/bind/db.nagios“
};
```

Die Einstellungen bedeuten, dass der Server primärer Nameserver der Domäne ist (`type master`) und dass keinerlei sekundäre Nameserver über Änderungen informiert werden müssen (`notify no`). Anschließend muss die Konfiguration nur noch neu geladen werden:

```
ServerA:# /etc/init.d/bind9 reload
```

Ob die Namensauflösung funktioniert, kann man mit einem der Werkzeuge `nslookup` oder `dig` getestet werden, z.B.

```
ClientA:# nslookup ServerB
Server:         10.0.2.1
Address:        10.0.2.1#53

Name:   ServerB.nagios
Address: 10.0.3.1
```

Für das Reverse Mapping (Punkt 4) existiert eine spezielle Domäne namens `in-addr.arpa`, die sämtliche Internetadressen auf ihren Hostnamen abbildet[108]. Für das Beispielszenario wird eine eigene Zonendatei benötigt, die aber völlig analog zur obigen aufgebaut ist:

```
$TTL 604800
@     IN   SOA  ServerA.nagios.     root.localhost. (
                     1         ; Serial
                     604800    ; Refresh
                     86400     ; Retry
                     2419200   ; Expire
                     604800 )  ; Negative Cache TTL
;
@         IN       NS       ServerA.nagios.
2.1       IN       PTR      Host.
1.1       IN       PTR      RouterA.
```

108Die IP-Adressen werden in umgekehrter dezimaler Notation angegeben. Die Domäne `in-addr.arpa` entspricht der root-Domäne (`.`) bei den Rechnernamen, der die bekannten Top-Level-Domains `de`, `net`, `com` etc. direkt untergeordnet sind. Die Domäne `in-addr.arpa` enthält also 256 direkte Unterdomänen, welche ihrerseits jeweils wieder 256 Unterdomänen enthalten usw. Da im Beispielszenario alle Adressen aus dem Netz `10.0.0.0/16` stammen, wird also eine Zonendatei für die Domäne `0.10.in-addr.arpa` angelegt.

```
2.2     IN      PTR     RouterA.
2.3     IN      PTR     RouterA.
2.4     IN      PTR     RouterA.
2.5     IN      PTR     RouterA.
1.2     IN      PTR     ServerA.
1.3     IN      PTR     ServerB.
1.4     IN      PTR     ClientA.
1.5     IN      PTR     Nagios.
```

Anstelle der A-Records enthält die Datei eine Reihe von PTR-Records (Pointer – Records), die jeweils einer Adresse einen Rechnernamen zuordnen. In `/etc/bind/named.conf.local` erfolgt wieder ein entsprechender Eintrag:

```
zone "0.10.in-addr.arpa" {
     notify no;
     type master;
     file "/etc/bind/db.10.0";
};
```

Damit ist die Konfiguration des Nameservers abgeschlossen. Auch die Funktion des Reverse Mapping kann (nach Neustart des `named`) mit `nslookup` auf einfache Art überprüft werden:

```
ClientA:# nslookup 10.0.3.1
Server:         10.0.2.1
Address:        10.0.2.1#53

1.3.0.10.in-addr.arpa   name = ServerB.
```
[109]

4.4.2 Überwachung

Für die Überwachung des DNS-Servers stellt Nagios die beiden Plugins `check_dns` und `check_dig` zur Verfügung. `check_dns` startet eine `nslookup`-Anfrage, `check_dig` verwendet den Domain Information Groper. Beide Plugins sind in der Handhabung recht ähnlich und bieten vergleichbare Funktionen, weshalb hier nur das Plugin `check_dns` näher besprochen werden soll. Die verfügbaren Optionen sind folgende:

`-H, --hostname` der aufzulösende Rechnername

109 vgl. (Langfeldt/Walter 2000)

-s, --dns-server	die Adresse des Nameservers
-a, --expected-address	die zu erwartende Adresse
-A, --expect-authority	nur autoritative Antworten akzeptieren

Darüberhinaus unterstützt es die Standardoptionen `-h`, `-V`, `-v`, `-t`, `-w` und `-c`, wobei über `-t`, `-w` und `-c` Zeitspannen in Sekunden übergeben werden. Läßt man die Option `-a` weg, werden beliebige Adressen akzeptiert, gibt man mit `-s` keinen expliziten Nameserver an, bedient sich `check_dns` den in `/etc/resolv.conf` angegebenen. Leider ist es mit keinem der beiden Plugins zurzeit möglich, auch das Reverse Mapping zu überwachen. Ein beispielhafter Aufruf von `check_dns` soll überprüfen, ob man eine autoritative Antwort auf die Anfrage nach der Adresse von `ServerB` erhält:

```
Nagios:# cd /usr/local/nagios/libexec/
Nagios:# ./check_dns -H ServerB -A -s 10.0.2.1
DNS OK: 0.076 seconds response time.
ServerB returns 10.0.3.1|time=0.076407s;;;0.000000
```

Den Abschluss dieses Abschnitts bildet wieder ein Paar Serviceobjekt-Definition/Kommandoobjekt-Definition, mit deren Aufnahme in die Nagios-Konfiguration der DNS-Service auf `ServerA` anhand der Abfrage aus obigem Beispiel überwacht werden kann, zudem wird über `-a` noch die erwartete Adresse übergeben:

```
define command{
     command_name   check_dns
     command_line   $USER1$/check_dns -s $ARG1$ -H $ARG2$
                    -a $ARG3$ -w $ARG4$ -c $ARG5$
     }

define service{
     use                 local-service
     host_name           ServerA
     service_description DNS-Check
     check_command       check_dns!10.0.2.1!ServerB!
                         10.0.3.1!2!5
     }
```

4.5. MySQL

Das Datenbankmanagementsystem *MySQL*[110] ist freie Software und steht für alle gängigen Betriebssysteme zur Verfügung. Es gehört seit vielen Jahren zu den beliebtesten Programmen im Bereich relationaler Datenbanksysteme. Seine Popularität verdankt MySQL seiner Schnelligkeit, auf die von den Entwicklern von vornherein großer Wert gelegt wurde, sowie der Vielzahl von Anbindungsmöglichkeiten an externe Applikationen, z..B. durch vorhandene Schnittstellen in vielen bekannten Programmiersprachen.

5.1. Installation/Konfiguration

Um MySQL im VNUML-Szenario benutzen zu können (mit `ServerB` als MySQL-Server), werden die Pakete `mysql-server` und `mysql-client` benötigt, die beide am einfachsten mit dem Paketmanager installiert werden.

```
ServerB# apt-get install mysql-server mysql-client
```

Nach der Installation befindet sich der MySQL-Server[111] schon in Ausführung, womit eine Kontrolle durch Nagios bereits möglich ist. Das zur Überwachung von MySQL-Datenbanken bereitgestellte Plugin `check_mysql` bietet als „Verfeinerung“ des Tests zusätzlich die Option an, eine bestimmte Datenbank anzusprechen. Eine solche muss aber zunächst noch angelegt werden, z.B.: (Benutzereingaben sind wieder blau hervorgehoben.)

(Einloggen als Administrator[112]:)

```
ServerB:~# mysql --user=root --password=xxxx
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40
Server version: 5.1.49-3 (Debian)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All
rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free
software, and you are welcome to modify and redistribute it
```

110 `http://www.mysql.com`

111 Der Prozessname ist `mysqld`.

112 Das hierbei verwendete Passwort für den MySQL-Administrator wurde während der Installation von MySQL vereinbart, es muss nicht mit dem Root-Passwort auf `ServerB` übereinstimmen.

```
under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.
```

(Anlegen einer neuen Datenbank `nagdb`:)

```
mysql > CREATE DATABASE nagdb;
Query OK, 1 row affected (0.00 sec)
```

(Anlegen eines neuen Benutzers `nagios`:)

```
mysql > CREATE USER nagios;
Query OK, 0 rows affected (0.02 sec)
```

(Lesezugriff ermöglichen und Ausloggen)

```
mysql > GRANT select ON nagdb.* TO nagios@10.0.5.1;
Query OK, 0 rows affected (0.00 sec)
mysql > exit
Bye
```

Damit der MySQL-Server auf Anfragen aus dem Netzwerk überhaupt reagiert, sind noch folgende Optionen in der MySQL-Konfigurationsdatei `/etc/mysql/my.cnf` auszukommentieren:

```
bind-address        127.0.0.1
skip-networking
```

Dadurch hat man von allen Maschinen des Netzes Zugriff auf den MySQL-Server. (Will man den Zugriff nur von bestimmten Maschinen ermöglichen, kann man dies über die Dateien `/etc/hosts.deny` und `/etc/hosts.allow` regeln. Um zum Beispiel im VNUML-Netz den Zugriff nur für die Maschinen `ServerB`(=`localhost`) und `Nagios` zu erlauben, können in diesen Dateien folgenden Zeilen ergänzt werden:

(in `/etc/hosts.deny`:)

```
mysql: ALL
```

(in `/etc/hosts.allow`:)

```
mysql: 127.0.0.1
mysql: 10.0.5.1 )
```

4.5.2 Überwachung

Das bereits erwähnte Plugin `check_mysql` kennt nur wenige Optionen, so fehlen beispielsweise Warn- bzw. kritische Schwellwerte, was die Ausführungsdauer eines Tests anbelangt. Eine Übersicht über alle zur Verfügung stehenden Optionen:

`-H, --hostaddress`	IP-Adresse des MySQL-Servers
`-P, --port`	zur Angabe eines alternativen Ports, auf dem MySQL läuft. Voreinstellung ist 3306.
`-d, --database`	zur Angabe einer Datenbank, zu der `check_mysql` verbinden soll, falls erwünscht.[113]
`-u, --user`	Benutzername, unter welchem `check_mysql` zum Datenbankserver verbinden soll.
`-p, --passwort`	(MySQL-)Passwort des mit `-u` angegebenen Benutzers.

Beispiele:

1. Für einen einfachen Aufruf von `check_mysql` benötigt man lediglich die Optionen `-H` und `-u`:

```
(nagios@Nagios:> cd /usr/local/nagios/libexec)
nagios@Nagios:> ./check_mysql -H 10.0.3.1-u nagios; echo $?[114]
Uptime:228  Threads:1  Questions:78  Slow queries:0  Opens:23
Flush tables:1  Open tables:17  Queries per second avg:0.342
0
```

2. Um zusätzlich zur oben angelegten Datenbank `nagdb` eine Verbindung aufzubauen, gibt man mit `-d` zusätzlich den Namen der Datenbank an (Ein Passwort wird nicht benötigt).

113 Ohne diese Option erfolgt lediglich eine Verbindung zum Datenbankprozess.

114 Mit dem `echo`-Befehl wird der Fehlercode ausgegeben, da der zurückgegebene Zustand bei `check_mysql` nicht aus dem zurückgegebenen Text erkennbar ist. Der in der letzten Zeile der Ausgabe erkennbare, zurückgegebene Fehlercode `0` entspricht dem Zustand `OK`.

```
nagios@Nagios:> ./check_mysql -H 10.0.3.1 -u nagios -d nagdb
Uptime:1561  Threads:2  Questions:95  Slow queries:0 Opens:23
Flush tables:1  Open tables:17  Queries per second avg:0.061
nagios@Nagios:> echo $?
0
```

Um die Überwachung des MySQL-Servers (gemäß Beispiel 2) dauerhaft zu gewährleisten, wird die Nagios-Konfiguration nach dem bekannten Schema um ein Service- und ein Kommandoobjekt erweitert:

```
define service{
     use                         local-service
     host_name                   ServerB
     service_description         MySQL
     check_command               check_mysql!nagios!nagdb
     }

define command{
     command_name                check_mysql
     command_line                $USER1$/check_mysql -H $HOSTADDRESS$
                                 -u $ARG1$ -d $ARG2$
     }
```

4.6 Mailserver

Einer der wichtigsten Dienste in Rechnernetzwerken überhaupt ist die Verteilung und Bereitstellung von E-Mails. Damit das in Abschnitt 1.6.2 beschriebene Nagios-Benachrichtigungssystem überhaupt arbeiten kann, soll als letztes Beispiel auf `ServerB` ein „vereinfachter Mailserver“[115] eingerichtet werden. Als Mail Transport Agent (MTA) kommt *Postfix*[116] zum Einsatz. Um den Zugriff auf die E-Mails über ein Browserfenster zu ermöglichen, bietet sich *Squirrelmail*[117] als Clientprogramm an. Dieses benötigt IMAP[118]-Funktionen, die z.B. der *UW IMAP-Server*[119] bereitstellt.

115 E-Mail-Verkehr zwischen Internet und dem virtuellen Netzwerk ist nicht vorgesehen, ebenso bleiben Sicherheitsaspekte beim Umgang mit den internen Nachrichten unberücksichtigt.
116 `http://www.postfix.org`
117 `http://squirrelmail.org`
118 Internet Message Access Protocol
119 `http://www.washington.edu/imap/`

4.6.1 Installation/Konfiguration

Installiert werden die Pakete `postfix`, `uw-imapd` und `squirrelmail`, zusätzlich sollen für Testzwecke noch ein paar Benutzerkonten angelegt werden.

```
ServerB:# apt-get install postfix
ServerB:# apt-get install uw-imapd squirrelmail
ServerB:# groupadd mailusers
ServerB:# mkdir /home/anna /home/berta /home/carla
ServerB:# useradd anna -d /home/anna -g mailusers
ServerB:# useradd berta -d /home/berta -g mailusers
ServerB:# useradd carla -d /home/carla -g mailusers
```
[120]

- **Postfix**

Die Konfiguration von Postfix beginnt bereits mit einer Abfrage während der Installation, für den Mailserver wählt man hier die Option „Internet Site“, sonst „Satellite System“. Letzte Anpassungen werden jeweils in der Datei `etc/postfix/main.cf` vorgenommen[121]. Unterschieden werden muss zwischen der Konfiguration für den Mailserver(`ServerB`) und derjenigen für die übrigen Rechner, wo Postfix sämtlichen Mailverkehr einfach an `ServerB` weiterleiten soll. Die wichtigsten Einstellungen im Überblick:[122]

myhostname: Der Name, unter welchem Client und Server miteinander kommunizieren. Er sollte mit dem vollen Namen identisch sein, den der jeweilige Gegenüber auch bei dem Versuch einer DNS-Adressauflösung erhält. Man kann sich diese Einstellung sparen und den Namen stattdessen von Postfix automatisch bestimmen lassen(s.u.).

mydomain: Der Zonenname des lokalen Netzwerkes, also `nagios`. Ist der

120 Als Passwörter werden für Anna, Berta, Carla aaaa, bbbb, cccc vereinbart. Anna, Berta, Carla werden auch zum Besitzer ihres jeweiligen Homeverzeichnisses gemacht.

121 Alternativ kann man auch eine Konfigurationsroutine mittels `dpkg-reconfigure postfix` aufrufen.

122 Detaillierte Angaben zur Konfiguration von Postfix findet man in (Hildebrandt/Koetter 2005,S.19ff).

Parameter gesetzt, setzt Postfix den Hostnamen aus dem Namen des Rechners[123] und dem Inhalt von `mydomain` zusammen, z.B. ergibt sich für `ServerA`: `myhostname=ServerA.nagios`. So können sich mehrere Rechner eines lokalen Netzwerks ein- und derselben Datei `main.cf` bedienen.

myorigin: Hier definiert man einen Zonennamen, der an nur als lokale Namen angegebe Sender und Empfänger angehängt wird. Die richtige Einstellung im Beispiel ist also wiederum `nagios`. Es besteht dabei die Möglichkeit, auf bereits definierte Parameter zurückzugreifen, indem man dem Namen ein Dollarzeichen voranstellt.[124] Dadurch soll die Anpassung der Konfiguration erleichtert werden, falls sich Adressen, Zonen- oder Hostnamen ändern.

mydestination: Hier gibt man in Form einer kommaseparierten Liste diejenigen Zonen an, für die Mails entgegengenommen und anschließend auf die einzelnen Postfächer verteilt werden sollen. Für den Mailserver muss diese Liste gemäß der bisherigen Einstellungen einen Eintrag `nagios` enthalten, für die übrigen UML-Maschinen bleibt die Liste leer.

relayhost: Die Adresse des Servers, an die die nicht lokal zustellbaren Nachrichten weitergeleitet werden, für den Mailserver bleibt dieses Feld leer, bei den übrigen Maschinen wird die Adresse des Mailservers, `10.0.3.1`, angegeben.

mynetworks: Hier müssen alle Netzwerke angegeben werden (in CIDR-Notation), von denen Mails entgegengenommen werden sollen - im Beispiel können die einzelnen Netze zusammengefasst werden zu `10.0.0.0/16`.

123 Zur Ermittlung des Namens ruft Postfix den Befehl `uname -n` auf.

124 Aus `myorigin=vnuml` wird damit z.B. `myorigin=$mydomain` .

- **Die Dateien `/etc/aliases` und `/etc/aliases.db`[125]**

In der Datei `/etc/aliases` lassen sich „Synonyme“ für die einzelnen Mailkonten definieren, falls man z.B. damit rechnet, dass Absender nicht den genauen Namen der Konten der gewünschten Empfänger kennen. So könnte man beispielsweise Aliase anlegen, die eine Weiterleitung von Nachrichten, welche an den Nachnamen eines Kontoinhabers gesendet wurden auf das reguläre Benutzerpostfach bewirken. Aber auch das Vervielfältigen bzw. einfache Umleiten von Nachrichten kann man hier erledigen. Im Falle des Beispielnetzes hat die Datei `/etc/aliases` (relevant nur auf dem Mailserver `ServerB`) folgendes Aussehen:

```
# /etc/aliases
mailer-daemon: postmaster
postmaster: root
nobody: root
hostmaster: root
usenet: root
news: root
webmaster: root
www: root
ftp: root
abuse: root
noc: root
security: root

"anna.alge": anna
"berta.bison": berta
"carla.columna": carla
root: carla, \root
```

Im oberen Abschnitt erkennt man, dass E-Mails aller virtuellen Benutzer in das Postfach von root umgeleitet werden. Die letzten vier Zeilen sind benutzerdefiniert und bewirken, dass Postfix E-Mails an Anna, Berta und Carla auch dann richtig zustellt, wenn der Absender irrtümlich den Nachnamen (hier Alge, Bison bzw. Columna), abgetrennt durch einen Punkt, mit angegeben hat.[126] Die letzte Zeile bewirkt schließlich das Duplizieren der

125 Manchmal befinden sich die Dateien auch im Verzeichnis `/etc/postfix`.
126 Der Punkt links vom Doppelpunkt erfordert die Verwendung der Anführungszeichen.

Nachrichten an root auf das Postfach von Carla.[127] Durch das Ausführen des Befehls `newaliases` wird auf Basis der oben angegebenen Definitionen die Datei `/etc/aliases.db` erzeugt, auf die Postfix dann zur Laufzeit zugreift.

Zum Testen der Konfiguration verwenden wir eine einfache Nachricht, die Anna an Berta von der Kommandozeile absendet:

```
anna@ClientA>  echo Testnachricht |                    →
               mail -s Testbetreff to-addr berta.bison
```

Ist alles in Ordnung, so wird der Absender auf `ClientA` zu `anna@nagios`, der Empfänger entsprechend zu `berta.bison@nagios` ergänzt, an `ServerB` weitergeleitet und dort schließlich im richtigen Postfach (`/var/spool/mail/berta`) abgelegt, was man leicht überprüft, z.B. indem man den Inhalt des Postfachs mittels `less /var/spool/mail/berta` inspiziert.

- **Squirrelmail**

Wesentlich leichter gestaltet sich die Konfiguration von Squirrelmail mithilfe einer mitgelieferten Konfigurationsroutine:

```
ServerB:# /usr/sbin/squirrelmail-configure
```

Im erscheinenden Menü muss man lediglich unter Punkt D, „Set pre-defined settings for specific IMAP servers “, die Voreinstellungen für die benutzte IMAP-Serversoftware, also „uw = University of Washington's IMAP server “, auswählen - speichern, fertig.[128]

Die Konfigurationsdateien von Squirrelmail liegen im Verzeichnis `/etc/squirrelmail`, eine von ihnen (`apache.conf`) enthält Einstellungen, die den Webserver Apache betreffen. Damit dieser auf die Datei zugreift, verwendet man eine symbolische Verknüpfung:

127 Dies ist zweckmäßig, da Squirrelmail den Zugriff auf das Konto von root nicht erlaubt.

128 Unter Umständen erlaubt IMAP nicht das unverschlüsselte Einloggen mit Squirrelmail. In diesem Fall ändert man in der Squirrelmail-Konfigurationsdatei `/etc/squirrelmail/config.php` die Einstellungen `$imapPort` von 143 auf 993 und `$use_imap_tls` von `false` auf `true`.

```
ln -s /etc/squirrelmail/apache.conf /etc/apache2/conf.d
```

Außerdem ist noch darauf zu achten, dass Squirrelmail für die Ordner, in denen sich die Postfächer befinden, Lese- und Schreibrechte besitzen muss und dass auf dem Mailserver der Webserver läuft. Zum Testen gibt man in der Adressleiste des Browsers `http://10.0.3.1/squirrelmail` ein, worauf man zum Login-Panel des Programms gelangen sollte (Abb. 14 links). Nach Eingabe von `berta` als Benutzername und ihrem Passwort (`bbbb`) sollte man die im letzten Abschnitt gesendete Testnachricht einsehen können (Abb. 14 rechts).

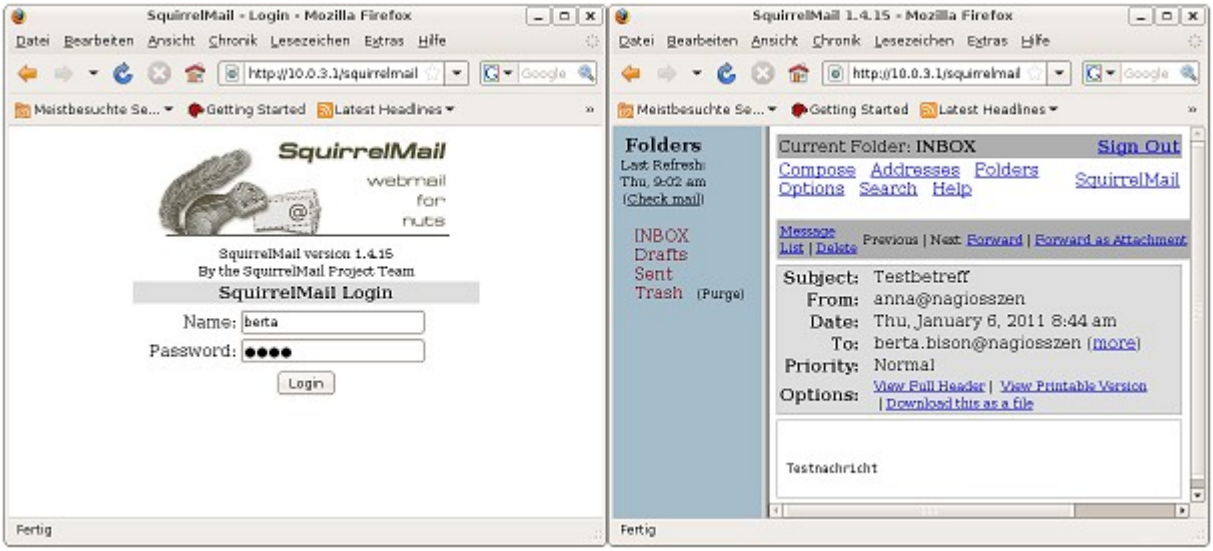

Abb. 14 Squirrelmail

4.6.2 Überwachung

Für die Überwachung des IMAP-Daemons bietet Nagios die Plugins `check_imap` und `check_simap`[129], welche wie schon das Plugin `check_ftp` lediglich Symlinks auf das Plugin `check_tcp` darstellen. Beim Pluginaufruf werden bereits folgende Parameter voreingestellt:

`-p, --port` wird bei `check_imap` auf 143, bei `check_simap` auf 993 gesetzt

`-e, --expect` Zeichenkette, die man in der Nachricht des Servers

129 check_simap für die SSL-verschlüsselte IMAP-Variante(IMAPS)

erwartet - wird auf `"* OK"`[130] gesetzt

`-q, --quit` Zeichenkette, die beim Beenden der Verbindung an den Server gesendet wird - ist auf `"a1 LOGOUT/r/n"` gesetzt

`-S, --ssl` für eine SSL-gesicherte Verbindung - bei `check_simap` gesetzt, bei `check_imap` nicht

Es stehen wieder die üblichen Standardoptionen zur Verfügung, wobei die über `-w` und `-c` angegebenen Schwellwerte für Antwortzeiten des Servers in Sekunden stehen. Die Liste aller verfügbaren Optionen ist lang, eine vollständige Aufstellung findet man in (Barth 2009, S. 116ff). Die einzelnen Postfix-Instanzen kommunizieren über SMTP[131] miteinander - daher wird Postfix mit dem Plugin `check_smtp` überwacht, das neben den Standardoptionen noch folgende weitere kennt:

`-p, --port` Voreingestellt ist der Standard-SMTP-Port 25

`-e, --expect` Zeichenkette, die in der Nachricht des Servers erwartet wird, voreingestellt ist `"220"`[132]

`-c, --command` Über diese Option lässt sich der Test verfeinern; komplette Mailkommandos können so an den Server geschickt werden.

`-R, --response` Zeichenkette, die man in der Antwort auf den mit `-c` angegebenen Befehl erwartet – im Falle einer falschen Antwort reagiert Nagios mit einer Warnmeldung

`-S, --starttls` Gibt man diese Option an, wird die Verbindung zum Server verschlüsselt.[133]

`-D, --certificate` Verwendet man -S, so gibt man über diese Option die Mindestrestgültigkeit des STARTTLS-Zertifikates in Tagen an.

130 Die Standardbegrüßung des IMAP-Servers ist „`* OK IMAP4rev1 Service Ready`“

131 Simple Mail Transfer Protocol

132 Die Standardbegrüßung des SMTP-Servers beginnt mit „220“

133 `STARTTLS` ist ein Mechanismus, der eine nachträgliche `SSL`- oder `TLS`-Verschlüsselung von Netzwerkverbindungen erlaubt. Es wird neben `SMTP` auch von anderen Protokollen wie `POP`, `IMAP` oder `LDAP` unterstützt.

-A, --authtype	Das benutzte Authentifikationsverfahren – will man diese Option verwenden, so ist `LOGIN` der einzige gültige Wert und man benötigt zusätzlich die Optionen `-U` und `-P`.
-U, --authuser	Der beim Login verwendete Benutzername
-P, --authpass	Das beim Login verwendete Passwort

Die mit `-w` bzw. `-c` übergebenen Schwellwerte sind bei `check_smtp` Zeitangaben in Sekunden. Erfolgt die Antwort zu spät, so setzt Nagios den Status des Dienstes entsprechend auf `WARNING` bzw. `CRITICAL`. Mit folgenden einfachen Kommando- und Serviceobjekten sollen die beiden Dienste auf `ServerB` überwacht werden:

```
# IMAP Kommandoobjekt
define command{
     command_name   check_simap
     command_line   $USER1$/check_simap -H $HOSTADDRESS$
     }

# IMAP Serviceobjekt
define service{
     use                 local-service
     host_name           ServerB
     service_description SIMAP
     check_command       check_simap
     }

# SMTP Kommandoobjekt
define command{
     command_name    check_smtp
     command_line    $USER1$/check_smtp -H $HOSTADDRESS$
     }

# SMTP Serviceobjekt
define service{
     use                 local-service
     host_name           ServerB
     service_description SMTP
     check_command       check_smtp
     }
```

Neben den besprochenen Plugins liefert Nagios noch `check_pop` und `check_spop` für die Überwachung von POP3[134]-Servern sowie `check_mailq` zur Überwachung der Mailqueue, die hier alle nicht benötigt werden und daher auch nicht näher erklärt werden sollen.[135]

Zum Abschluß dieses Kapitels ein Screenshot der Nagios-Weboberfläche (Schaltfläche Service Detail), mit dem man einen Überblick über sämtliche in diesem Kapitel definierten Services gewinnt:

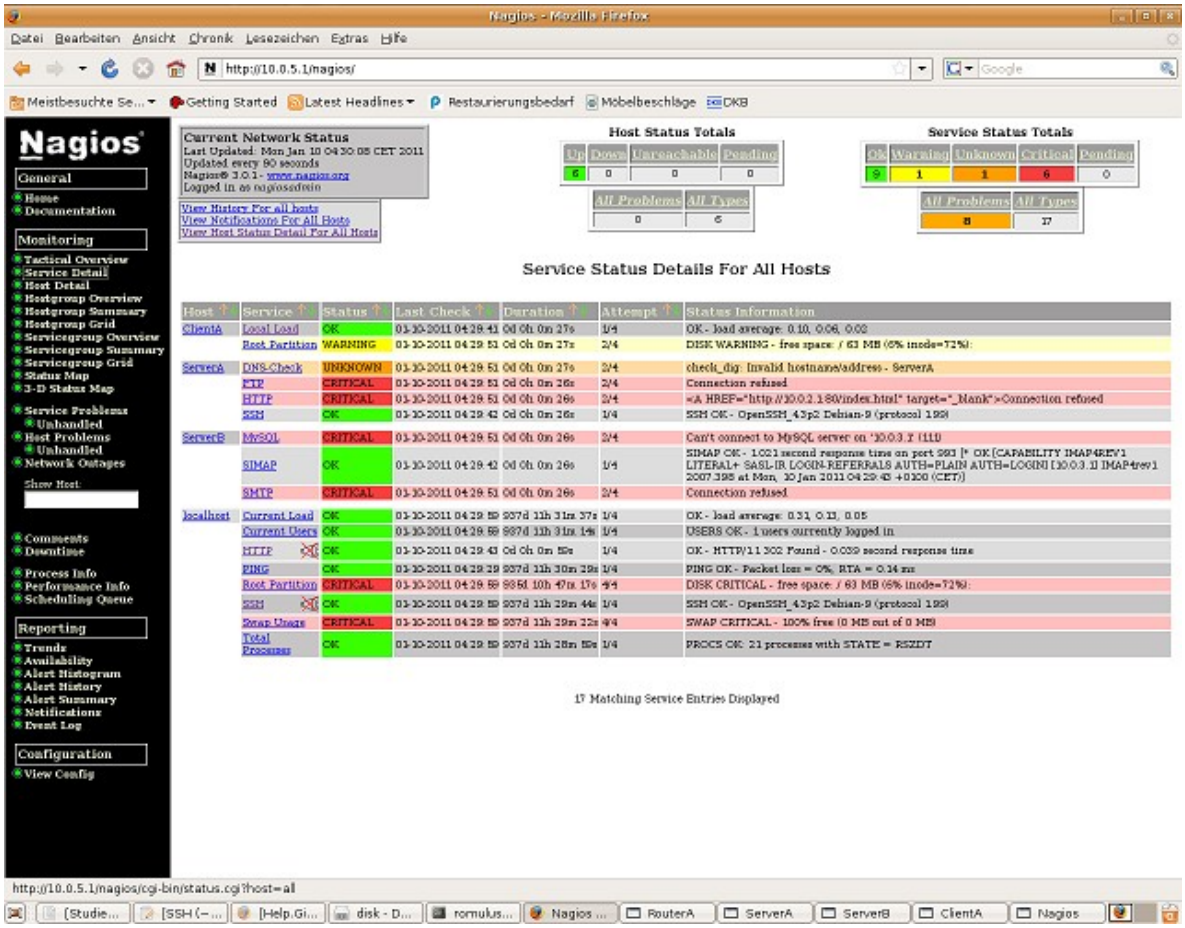

Abb. 15 Service Detail inklusive Netzwerkdiensten

Die Liste beginnt mit den beiden per `check_by_ssh` überwachten lokalen Diensten auf `ClientA`, es folgen die verschiedenen Netzwerkdienste auf den Servern A und B, ganz unten die Services auf `localhost`, also auf dem Nagios-Server, die bereits in der Nagios-Grundkonfiguration definiert waren.[136]Viele der Netzwerkdienste befinden sich im Zustand `CRITICAL`, weil die

134 Post Office Protocol

135 Ausführliche Informationen zu diesen Plugins in (Barth 2009, S 116ff, S.182ff).

136 vgl. Abb7: Hier sieht man dieselbe Seite, vor der Aufnahme der in Kapitel 4 neu erstellten Definitionen in die Nagios-Konfiguration.

entsprechenden Dienste noch nicht gestartet wurden.[137]Eine Ausnahme ist der DNS-Test; er befindet sich im Zustand `UNKNOWN`, da Nagios ohne laufendes DNS schon daran scheitert, den Namen des DNS-Servers zu einer gültigen Adresse aufzulösen.[138]Die Dienste RootPartition auf `localhost` und `ClientA` befinden sich im Zustand `CRITICAL` bzw. `WARNING`, da (wie so oft) der „Festplattenplatz“ der virtuellen Maschinen zur Neige geht.[139]Der Status `CRITICAL` für den Dienst Swap Usage auf `localhost` kommt dadurch zustande, dass auf den virtuellen Maschinen keinerlei Auslagerungsspeicher angelegt ist.

137 Auf das Starten der Dienste ist „für ein wenig mehr Farbe im Bild“ verzichtet worden.

138 Wäre im Plugin-Aufruf statt des Namens die Adresse des DNS-Servers angegeben, befände sich auch dieser Dienst im Zustand CRITICAL.

139 Da beide Maschinen auf (nahezu) identischen Dateisystemen arbeiten, müssen für die beiden Dienste offenbar unterschiedliche Schwellwerte angegeben worden sein. (Daher der unterschiedliche Status.)

Fazit

Nagios ist mittlerweile zu einem sehr leistungsstarken Monitoring-Werkzeug geworden, seine vielseitigen Einsatzgebiete konnten im Rahmen der vorliegenden Arbeit nur auszugsweise vorgestellt werden. So wurde auch die ursprüngliche Absicht, noch zwei der Nagios-Add-Ons, eines für das automatische Erstellen der Nagios-Konfiguration, eines für die graphische Darstellung der gesammelten Daten, zu besprechen, wieder fallengelassen. Die primären Ziele, der Betrieb von Nagios im virtuellen UML-Netzwerk und die Beschreibung derjenigen Nagios-Komponenten, welche für die Überwachung von Netzwerkdiensten zuständig sind, wurden hingegen erreicht.

Da im virtuellen VNUML-Netzwerk – zumindest nach den Erfahrungen des Autors – wesentlich öfter Ausfälle zu beobachten sind als in realen Rechnernetzen, ist die Verwendung eines Monitoringtools wie Nagios sicherlich angezeigt. Auch können, bedingt durch das Fehlen eines eigenen Terminals für die einzelnen Maschinen, die Ausfälle nur umständlich festgestellt werden und bleiben zunächst unbemerkt. Schließlich gewinnt die gesamte Simulation mit dem Einsatz von Nagios an Wert, da durch die regelmäßig durchgeführten Tests für regen Datenverkehr im virtuellen Netzwerk gesorgt ist.

Quellenangaben

Baader, Hans-Joachim (Baader 2003): Ein Beispiel zur Konfiguration des Nameservers BIND, http://www.pro-linux.de/t_netzwerk/dns.html (11.10.2010)

Barth, Wolfgang (Barth 2009): Nagios System- und Netzwerkmonitoring (2.Auflage), München: Open Source Press, 2009.

Franken, Johannes (Franken 2010): OpenSSH Teil 1: Grundlagen, http://www.jfranken.de/homepages/johannes/vortraege/ssh1.en.html/vortraege/ssh1_inhalt.de.html (3.1.2011)

Galstad, Ethan (Galstad 2005): Interview vom 22.2.2005, http://archive.fosdem.org/2005/index/interviews/interviews_galstad.html (3.1.2011)

Hildebrandt, Ralf / Koetter, Patrick (Hildebrandt/Koetter 2005): The Book Of Postfix, San Francisco : No Starch Press, 2005.

Langfeldt, Nikolai / Walter, Thomas (Langfeldt/Walter 2000): DNS HOWTO, http://www.masterscripts.de/howto/DE-DNS-HOWTO.html (3.1.2011)

Lowes, Mark (Lowes 2001): Proftpd A User's Guide, http://www.proftpd.org/localsite/Userguide/linked/userguide.html (11.10.2010)