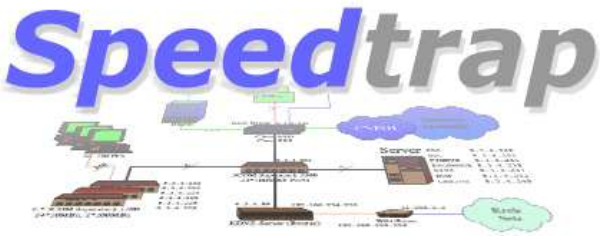
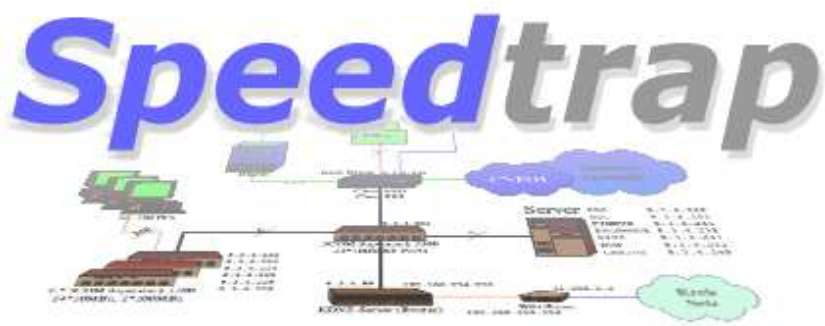
Projektgruppe 422




skilled performant ethernet examination by distributed traffic agents project

Endbericht

Endbericht Projektgruppe 422




# Entwicklung eines Java-basierten Frameworks für verteilte Netzwerk-Messungen

Babak Davani, Hayat Elfassi, Daniel Grabowski, Alexander Harsch
Ulrich Hilmer, Chiagozie Madukife, Firas Raed
Martin Schindler, Björn Schmidt, Christian Winkler

Revision: 2.7.33 22.12. 10:56:50
Universität Dortmund
Fachbereich Informatik
Lehrstuhl IV

Dortmund, 22. Dezember 2003

# Inhaltsverzeichnis

# Änderungen der letzten Revisionen:

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.26 14.12. 1:45:14

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.27 14.12. 1:49:37

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.28 14.12. 1:51:40

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.29 14.12. 1:54:09

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.30 14.12. 2:03:07

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.31 14.12. 2:04:48

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.32 14.12. 2:06:57

../handbuch/administrationshandbuch .
../handbuch/administrationshandbuch/index.tex .

Revision: 2.7.33 22.12. 10:56:50

../fazitpgmitglieder/juergen/index.tex .

# Kapitel 0

# Einführung

Dieser Endbericht wurde von den Teilnehmern der PG 422 „Entwicklung eines Java basierten Frameworks für verteilte Netzwerk-Messungen" erstellt. Er dient der Dokumentation der im Sommersemester 2003 von den Autoren dieses Berichts erzielten Ergebnisse.

Als Grundlage für den Projektablauf im zweiten PG-Semester wird in Kapitel 1 auf das Projektmanagement eingegangen, wobei in diesem Semester schwerpunktmäßig die Ausführungsphase geplant wurde. Nachdem kurz die Motivation und die Projektbeschreibung behandelt wurden, wird im weiteren Verlauf dieses Kapitels auf das erste PG-Semester respektiv eingegangen. Nachdem die zur Verfügung stehenden Ressourcen und Mitarbeiter für dieses Projekt aufgelistet wurden, werden die Ziele für das zweite Semester definiert und der Projektablauf geplant. Hierfür wurde der Projekt- bzw. Terminplan und die Deadlines festgelegt, sowie die einzelnen Arbeitsgruppen eingeteilt und Dokumentationsrichtlinien aufgestellt.

Zu Beginn des zweiten PG-Semesters wurde ein eintägiges Seminar veranstaltet. Die Seminarthemen wurden den besonderen Anforderungen der Ausführungsphase des Projekts entsprechend ausgewählt. Einen Überblick über die Seminarthemen im einzelnen gibt Kapitel 2.

Kapitel 3 befasst sich mit der Implementierung von Speedtrap. Hier beschreiben die einzelnen Arbeitsgruppen Einzelheiten zur Implementierung ihrer jeweiligen Softwarekomponente. Die Softwarekomponenten von Speedtrap sind: Agent, Monitor und Netzplan.

Im vierten Kapitel geht es um den Testeinsatz des fertigen Produkts. Zunächst wurden, gemäß der im PG-Antrag definierten Ziele, Testreihen am LS4 durchgeführt. Hierzu wird das Vorgehensmodell für die Testreihen und die entsprechende Durchführung der Tests beschrieben. Danach werden die Testergebnisse ausführlich dokumentiert und ein abschließendes Fazit gegeben.

Danach (Kapitel 5) wird der Testeinsatz bei der Kreispolizeibehörde Unna beschrieben, die Ergebnisse der Messungen vorgestellt und ausgewertet. Die Auswertung dieser Messungen lassen Schlüsse bzgl. der Schwachpunkte des Polizeinetzwerks zu, die für ein effizientes Netzwerk-Management wertvoll sein können.

In Kapitel 6 werden die Handbücher zu Speedtrap vorgestellt. Es gibt ein Benutzungshandbuch, das den regulären Umgang mit Speedtrap beschreibt und Installationshinweise gibt. Das Administrationshandbuch ist für den versierten Benutzer oder Administrator des Netzwerks vorgesehen. Hier werden detailliertere Installationshinweise und eine Schnittstellenbeschreibung zur Kommunikation zwischen Monitor und Agent gegeben. Die Schnittstellenbeschreibung ermöglicht auch eine Weiterentwicklung des Produkts.

Mit dem Fazit der PG-Teilnehmer schließt der Endbericht ab.

# Kapitel 1

# Projektmanagement

Im folgenden Kapitel soll, nach einem kurzen Rückblick auf das erste Semester, die Projektarbeit im zweiten Semester aus Sicht des Projektmanagementes dokumentiert werden. Der Projektgegenstand und das Ziel der Projektgruppe wird kurz beschrieben. Die einzelnen Arbeitsschritte im zweiten Semester und die Gruppen- und Ressourcenaufteilung werden dokumentiert. Das Projekt gliedert sich grob in Einleitungs-, Planungs- und Ausführungsphase. Während Einleitungs- und Planungsphase im ersten Semester abgeschlossen wurden, beschäftigt sich das zweite Semester mit der Ausführungsphase.

## 1.1 Motivation

Für Planung, Betrieb und Überwachung von Rechnernetzen sind aufgrund des schnellen technologischen Fortschritts Softwarewerkzeuge erforderlich, die ein schnelles Planen und Eingreifen ermöglichen. Da sich in heutigen Client-/Server-Netzwerken die Topologie relativ häufig ändert, ist eine Schätzung oder Vorhersage der genauen Route, die ein ausgezeichnetes Datenpaket einschlägt meist sehr schwierig. Noch schwieriger erweist sich, die zugehörige Antwortzeit zu bestimmen. In heterogenen Hard- und Softwareplattformen wird das Zusammenspiel aller vernetzten Komponenten durch sogenannte Middleware erreicht. Die so genannten Leistungs-Managementsysteme hinken dieser Entwicklung aber noch hinterher. Diese Systeme ermitteln die Systembelastung und zeigen Leistungsengpässe auf. Mit ihrer Hilfe lassen sich Fehler in der Systemplanung (hier im speziellen Fehler in der Netzwerkplanung) erkennen, weiterhin helfen Leistungs-Managementsysteme bei der Planung und dem Ausbau der überwachten Netzwerke in der Zukunft. An dieser Stelle soll die Arbeit der PG 422 ansetzen und ein Framework für ein Leistungs-Managementsystem entwickeln.

## 1.2 Projektbeschreibung

Die Projektgruppe 422, „Entwicklung eines Java-basierten Frameworks für verteilte Netzwerkmessungen", wird sich schwerpunktmäßig mit der Ermittlung der Systembelastung in heterogenen Netzen beschäftigen. Ziel ist es, Leistungsengpässe anzuzeigen und Lösungsvorschläge aufzuzeigen. Unter Berücksichtigung des Verhaltens der Antwortzeit, der Verweilzeit und der Verzögerungszeit, sowie der theoretischen Leistungsgrenzen soll die Netzlast ermittelt werden. Diese zu messenden Parameter werden durch viele übertragungstechnische Eigenschaften beeinflusst, so beispielsweise von der Flusskontrolle, den Zugangsverfahren, dem Dämpfungsverhalten oder den Paketverlustraten. Das zu entwickelnde Tool soll Aufschluss über die Netzauslastung geben und zur Ermittlung

der Leistungstrends für die weitere Netzplanung dienen. Ziel wird es sein, ein Java-basiertes Tool zu entwerfen, und mit dessen Hilfe im Rechnernetz der Kreispolizeibehörde Unna verteilte Meßexperimente durchzuführen. Dieses Tool wird nur eine Komponente bzw. ein Teil eines ebenfalls von der PG 422 zu entwerfenden Frameworks sein. Das Framework soll nach der Entwicklung zunächst im lokalen Netz des Lehrstuhls Informatik IV getestet werden. Das Tool soll im zu überwachenden Rechnernetz mittels verteilter Agenten Daten von intelligenten Switchen dezentral messen und einer zentralen Monitoring-Komponente zur weiteren Bearbeitung zur Verfügung stellen.

## 1.3 Das erste Semster

Im ersten Semester wurde von den Teilnehmern der PG 422 „Speedtrap" zunächst ein zweitägiges Seminar durchgeführt, wobei jede(r) PG-Teilnehmer(in) einen Vortrag zu einem für die PG relevanten Thema gehalten hat. Die einzelnen Seminarausarbeitungen, wie auch alle anderen im folgenden kurz skizzierten Ergebnisse des ersten Semesters sind in den Zwischenbericht [P422ZB] eingeflossen.

Zur Planung des Projektablaufs wurde wurde das Projekt zunächst beschrieben und auf die zur Verfügung stehenden Ressourcen bzw. deren Verwaltung eingegangen, die benötigten Werkzeuge (z.B. Entwicklungsumgebung, Dokumentationswerkzeug) festgelegt und der Projektablauf geplant. Es wurde ein Projekt- bzw. Terminplan für das erste Semster erstellt und ein Vorgehensmodell skizziert.

Zur Vertiefung der Seminarthemen wurden im Anschluss an das Seminar zwei Praktika durchgeführt. Im Rahmen des Modellierungspraktikums wurde in kleinen Gruppen mit Hilfe des vom Lehrstuhl Informatik 4 entwickelten Modellierungstools HIT mehrere Aufgaben aus den Bereichen Warteschlangentheorie, Simulation und Leistungsbewertung praktisch vertieft. Im darauf folgenden Messpraktikum wurden Aufgaben im Bereich Leistungsmessung bzw. Messtools bearbeitet.

Um einen Einblick in die Infrastruktur der Einsatzumgebungen zu erhalten, wurden die Topologien der beiden später zu messenden Rechnernetze am Lehstuhl IV und bei der Kreispolizeibehörde Unna durch die Erstellung von Netzplänen dieser Netze dokumentiert.

Auf Basis der durch die Netzpläne zusätzlich gewonnenen Informationen konnte nun eine Systemstrukturierung für das zu entwickelnde Tool erstellt werden. Hierfür wurden die zu messenden Größen festgelegt sowie Eingaben, Ausgaben und Schnittstellen definiert. Die Voraussetzung zur Erstellung des Anforderungs- und Analysemodells war nun in Form der Anforderungsdefinition gegeben.

Als Grundgerüst für den späteren Entwurf (und die anschließende Implementierung) wurden in diesem Kontext ein Problembereichsmodell, Anwendungsfalldiagramme, GUI-Definition und Strukturmodell als Teile des Entwurfsmodells erarbeitet.

Nachdem im Rahmen des Entwurfs JavaSpaces und SNMP hinsichtlich der Verwendbarkeit für Speedtrap untersucht worden waren, wurde das Strukturmodell weiterentwicklet. Dieses Modell, welches die Basis für die Implementierung im diesem Semester bildet, schloss das erste PG-Semester ab.

In den Semesterferien wurde ein weiteres Seminar abgehalten. Im Rahmen dieses Seminars wurden die für die Implementierung wichtigen Themen erarbeitet und den PG-Teilnehmern präsentiert. Diese Themen sind: Stochastische Analyse von Messwerten (hinsichtlich der Verteilungsfunktion), Tools zur automatischen Erkennung des Netzwerks, SNMP Grundlagen, Analyse der Einsatzmöglichkeiten von JavaSpaces und fortgeschrittene Java-Grundlagen. Die Ergebnisse dieses Seminars finden sich in Kapitel 2 dieses Endberichtes.

Im zweiten Semester rückt die Implementierung in den Vordergrund. Nach der Testphase, dem erfolgreichen Testeinsatz am Lehrstuhl IV und bei der Kreispolizeibehörde Unna und der Verfizierung der Messungen mittels des Tools SimulaNT, welches von der Vorgänger PG entwickelt wurde, schließt die PG mit diesem Endbericht ab.

## 1.4 Ausführungsphase

Die Ausführungsphase ist die dritte und letzte Phase des Projekts. Die beiden vorhergehenden Phasen "Planung" und "Einleitung" wurden bereits im Zwischenbericht dokumentiert. Die im zweiten Semester verbleibenden arbeiten beschäftigen sich mit Implementierung, Testeinsatz und Inbetriebnahme. Bevor auf die eigentliche Planung der Ausführungsphase eingegangen wird, wird kurz auf die Ressourcen und Schnittstellen eingegangen.

### 1.4.1 Ressourcen

#### Personal

Die PG 422 setzt sich aus einem Projektleiter und 10 Projektmitgliedern zusammen. Die Personen im einzelnen:

- Jürgen Mäter (Projektleiter)
- Babak Davani
- Hayat Elfassi
- Daniel Grabowski
- Alexander Harsch
- Ulrich Hilmer
- Chiagozie Madukife
- Firas Raed
- Martin Schindler
- Björn Schmidt
- Christian Winkler

#### Material

Der PG stehen weiterhin folgende materielle Ressourcen zur Verfügung:

- Ein PG Schrank
- Ein PG Sitzungsraum
- Linux und Solaris Pool-Rechner
- 3000 Seiten Druckkontingent
- Windows NT Rechner bei der Kreispolizeibehörde sowie am Lehstuhl 4

### 1.4.2 Ressourcenverwaltung

Die im ersten Semester delegierten Verantwortlichkeiten an einen oder mehrere PG Mitglieder bleiben wie folgt unverändert:

- Christian als Qualitätsmanager
- Alex als Konfigurationsmanager und Ansprechpartner für die IRB
- Babak und Björn als Projektmanager
- Daniel als Ansprechpartner für die Kreispolizeibehörde Unna
- Martin als Schrankverwalter
- Chiagozie als Ressourcenverwalter
- Ulrich als Ansprechpartner für den Lehrstuhl 4

### 1.4.3 Statusberichte

Die Mitglieder der PG 422 treffen sich wöchentlich in einer Gruppensitzungen im PG Sitzungsraum R13, Pavillion 6:

- Montag von 12:15 bis ca. 13:45
- Freitag von 12:15 bis ca. 13:45 als Ausweichtermin

Bei jeder Sitzung wird ein Protokoll über die Inhalte erstellt. Der Protokollant wechselt zyklisch in alphabetischer Reihenfolge der Vornamen. Die Sitzung wird grundsätzlich vom letzten Protkollant geleitet.

Nach Verlesen des vorherigen Protokolls geben der Konfigurationsmanager, Qualitätsmanager und die Projektmanager ein Statement über den Stand der Dinge ab und leiten damit die Veranstaltung ein.

### 1.4.4 Ziele im zweiten Semester

Zu Beginn des Semsters wurden die noch verbleibenden Minimalziele für dieses Semester erörtert, dies sind im einzelnen:

**Erstellung einer JavaSpaces Analyse:** Gefragt ist, ob JavaSpces geeignet ist, um plattformübergreifende Messungen zu unterstützen, d.h. für unser Messtool sinnvoll einsetzbar ist Die Untersuchung wurde bereits im Seminar abgehandelt und wird in Kapitel 2 dargestellt.

**Entwicklung und Implementierung eines Frameworks für verteilte Messexperimente:** Kapitel 3 wird hierauf näher eingehen.

**Analyse einer verteilten Anwendung am LS4:** Gefragt ist, warum der Speedup bei 10 Rechnern nur 1,7 beträgt. Der Testeinsatz unseres Tools wird in Kapitel 4 dokumentiert.

**Installation und Einsatz des entwickelten Tools in Unna:** Hier soll unser Tool im Einsatz bei der Polizei seine Brauchbarkeit beweisen, dokumentiert in Kapitel 5.

**Verifizierung der Messdaten vom LS4 und aus Unna mit SimulaNT:** Die gemessenen Werte sollen mit den von uns durch das Tool SimulaNT der Vorgänger PG berechneten Werten verglichen werden, dokumentiert jeweils in Kapitel 4 und 5.

**Erstellung des Endberichtes:** Zum Ende der PG wird der hier vorliegende Endbericht als PG Abschluss vollendet werden.

### 1.4.5 Projektablauf und Vorgehensmodell

Die PG 422 konnte sich am Anfang des zweiten Semesters recht schnell auf das weitere Vorgehen einigen. Diagramm 1.1 stellt es dar.

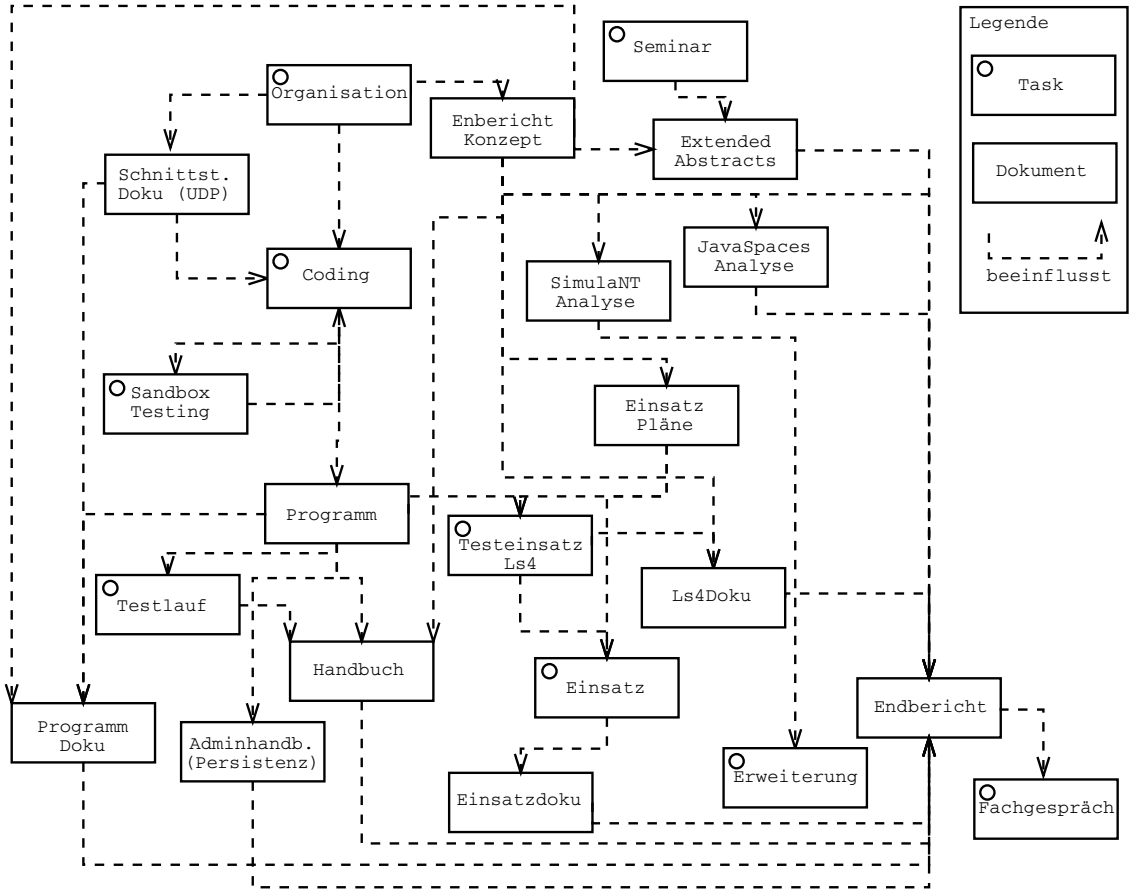



Abbildung 1.1: Der Projektktplan im zweiten Semester im Überblick

Neben dem bereits angesprochenen *Seminar* in dem auch schon die Verwendbarkeit von *JavaSpaces* für unser Projekt erörtert wurde und der Erstellung der *Extended Abstracts*, die direkt in den *Endbericht* einfliessen wird das weitere Vorgehen im Task *Organisation* geplant.

Das daraus resultierende *Endberichtskonzept* sowie die *Schnittstellendoku*, die die Kommunikation zwischen den Softwarekomponenten „Monitor" und „Agent" genau festlegt, fliessen einerseits direkt in den *Endbericht* ein, beeinflussen andererseits die Java Programmierung, hier *Coding* genannt. Nach Abschluss der Programmierung und internen *Testläufen* (SandboxTesting) wird das entwickelte Progamm zunächst am Lehrstuhl IV einem *Testlauf* unterzogen und danach bei der Kreispolizeibehörde Unna eingesetzt *(Einsatz)*.

Dazu werden parallel zur Programmierungsphase *Einsatzpläne* fuer den Lehstuhl IV und Unna erstellt. Sie fliessen genauso, wie die spätere *Einsatzdokumentation* der beiden Einsätze in den Endbericht ein. Nach den Einsätzen werden die erhobenen Daten mit dem Tool „SimulaNT" der Vorgängerprojektgruppe verifiziert bzw. auf Plausibilität geprüft *SimulaNT. Erweiterungen* unseres Tools Speedtrap sind an dieser Stelle möglich, bleiben aber optional. auch die Ergebnisse dieses Tasks fliessen in den Endbericht ein.

Nach Fertigstellung der Programmierung werden Administrations- und Benutzungshandbücher erstellt, welche dann auch als Kapitel im Endbericht platziert werden. Das *Fachgespräch* bildet nach Fertigstellung des Endberichts den Abschluss der PG und stellt die Präsentation der Projektarbeit in Form eines Vortrages dar.

### 1.4.6 Terminplan

Der im letzten Abschnitt beschriebene Ablauf ist in Abbildung 1.2 im Zeitdiagramm und folgend noch einmal tabellarisch dargestellt.

**21.4.-27.4.:** Organisation, Endberichtkonzept

**21.4.-27.4.:** Extended Abstracts des Seminars

**21.4.-04.5.:** Schnittstellendokumentation

**21.4.-22.6.:** Java Coding, Java Dokomentation

**21.4.-25.5.:** Einsatzpläne für den LS4 und Unna incl. erweiterter bzw. angepasster Netzpläne

**05.5.-29.6.:** Sandboxtests, d.h. Test der einzelnen Programmteile

**26.5.-20.7.:** Benutzungshandbuch, Administrationshandbuch u.a. bestehend aus Schnittstellendokumentation und Persistenzdokumentation

**16.6.-29.6.:** PG interne Testläufe

**23.6.-06.7.:** Testeinsatz am LS4

**07.7.-13.7.:** Einsatz in Unna, Ls4 Einsatzdokumentation des Messauftrages

**23.6.-20.7.:** SimulaNT Analyse bzw. Verifikation für die Einsatzergebnisse am LS4 und in Unna

**02.7.-20.7.:** Einsatzdokumentation Unna

**22.4.-28.7.:** Endbericht

**21.7.-11.8.:** Fachgesprächsvorbereitung

### 1.4.7 Deadlines

Aus dem im letzten Abschnitt vorgestellten Projektablauf resultieren die folgenden Deadlines, die hier noch einmal übersichtlich dargestellt sind.

**27.4.:** Endberichtkonzept, Extended Abstracts

**04.5.:** Schnittstellendokumentation

**25.5.:** Einsatzpläne (LS4 und Unna incl. Netzpläne)

**22.6.:** Programmierung, Java Dokumentation, Persistendokumentation

**06.7.:** Testeinsatz am LS4

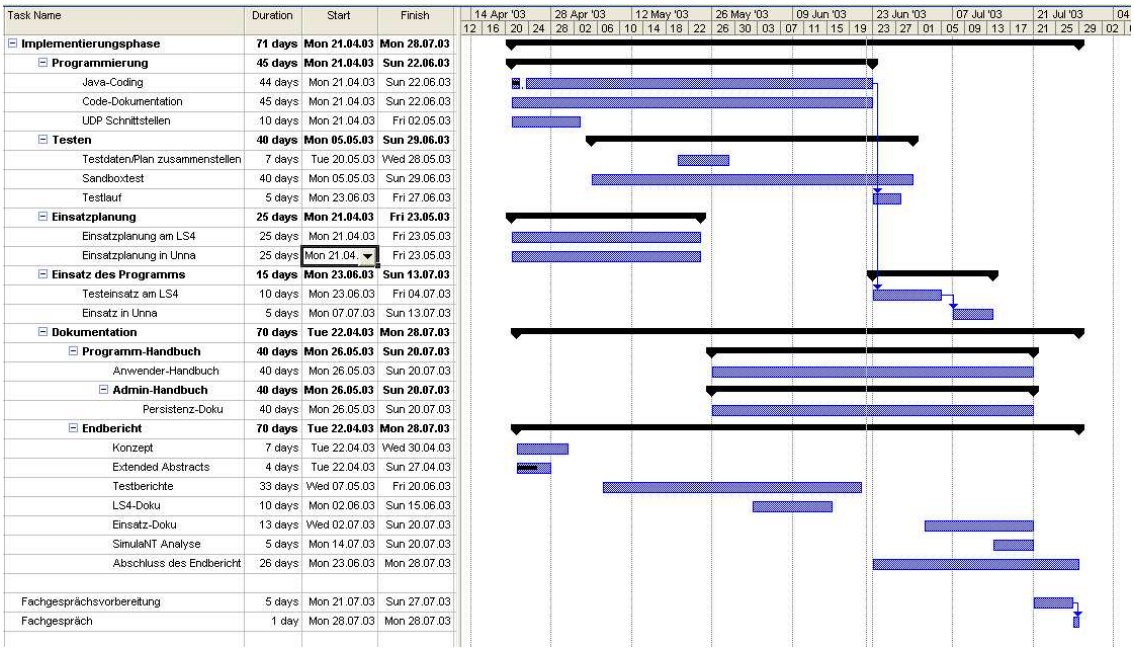
**13.7.:** Einsatz in Unna, SimulaNT Analyse für den LS4, Einsatzdokumentation für den LS4

**20.7.:** Einsatzdokumentation Unna

**20.7.:** Benutzungshandbuch, Administrationshandbuch, SimulaNT Analyse für Unna

**28.7.:** Endbericht

**11.8.:** Fachgespräch



| Task Name | Duration | Start | Finish |
|---|---|---|---|
| **Implementierungsphase** | **71 days** | **Mon 21.04.03** | **Mon 28.07.03** |
| **Programmierung** | **45 days** | **Mon 21.04.03** | **Sun 22.06.03** |
| Java-Coding | 44 days | Mon 21.04.03 | Sun 22.06.03 |
| Code-Dokumentation | 45 days | Mon 21.04.03 | Sun 22.06.03 |
| UDP Schnittstellen | 10 days | Mon 21.04.03 | Fri 02.05.03 |
| **Testen** | **40 days** | **Mon 05.05.03** | **Sun 29.06.03** |
| Testdaten/Plan zusammenstellen | 7 days | Tue 20.05.03 | Wed 28.05.03 |
| Sandboxtest | 40 days | Mon 05.05.03 | Sun 29.06.03 |
| Testlauf | 5 days | Mon 23.06.03 | Fri 27.06.03 |
| **Einsatzplanung** | **25 days** | **Mon 21.04.03** | **Fri 23.05.03** |
| Einsatzplanung am LS4 | 25 days | Mon 21.04.03 | Fri 23.05.03 |
| Einsatzplanung in Unna | 25 days | Mon 21.04.. | Fri 23.05.03 |
| **Einsatz des Programms** | **15 days** | **Mon 23.06.03** | **Sun 13.07.03** |
| Testeinsatz am LS4 | 10 days | Mon 23.06.03 | Fri 04.07.03 |
| Einsatz in Unna | 5 days | Mon 07.07.03 | Sun 13.07.03 |
| **Dokumentation** | **70 days** | **Tue 22.04.03** | **Mon 28.07.03** |
| **Programm-Handbuch** | **40 days** | **Mon 26.05.03** | **Sun 20.07.03** |
| Anwender-Handbuch | 40 days | Mon 26.05.03 | Sun 20.07.03 |
| **Admin-Handbuch** | **40 days** | **Mon 26.05.03** | **Sun 20.07.03** |
| Persistenz-Doku | 40 days | Mon 26.05.03 | Sun 20.07.03 |
| **Endbericht** | **70 days** | **Tue 22.04.03** | **Mon 28.07.03** |
| Konzept | 7 days | Tue 22.04.03 | Wed 30.04.03 |
| Extended Abstracts | 4 days | Tue 22.04.03 | Sun 27.04.03 |
| Testberichte | 33 days | Wed 07.05.03 | Fri 20.06.03 |
| LS4-Doku | 10 days | Mon 02.06.03 | Sun 15.06.03 |
| Einsatz-Doku | 13 days | Wed 02.07.03 | Sun 20.07.03 |
| SimulaNT Analyse | 5 days | Mon 14.07.03 | Sun 20.07.03 |
| Abschluss des Endbericht | 26 days | Mon 23.06.03 | Mon 28.07.03 |
| Fachgesprächsvorbereitung | 5 days | Mon 21.07.03 | Sun 27.07.03 |
| Fachgespräch | 1 day | Mon 28.07.03 | Mon 28.07.03 |



Abbildung 1.2: Der Zeitplan im zweiten Semester im Überblick

### 1.4.8 Arbeitsgruppen und Zuständigkeiten

Die in den vorigen Abschnitten vorgestellten Aufgaben wurden auf die PG Mitglieder verteilt. für die wichtigsten Tasks wurden Arbeitsgruppen gebildet, die nachfolgend kurz beschrieben werden:

**Einsatzplan LS4:** Diese Gruppe entwickelt einen Einsatzplan für den LS4, entscheidet, wo Agenten aufgestellt werden und wie bzw. was genau gemessen werden soll. Weiterhin werden die für die Messung benötigten Ressourcen bei der IRB[1] angefordert und abgestimmt.

- Mitglieder: Uli, Daniel, Firas, Hayat

**Einsatzplan Unna:** Auch für Unna muss ein detailierter Einsatzplan entwickelt werden, was diese Gruppe leisten soll

- Mitglieder: Firas, Babak

**Messauftrag LS4:** Nach dem Entwickelten Einsatzplan für den LS4 soll von den Mitgliedern dieser Gruppe der Test-Einsatz bzw. die Messung durchgeführt und dokumentiert werden.

- Mitglieder: Uli, Daniel, Hayat

**Einsatzbericht Unna:** Diese Gruppe ist für den Einsatz des endgültigen Produkts "Speedtrap" in Unna verantwortlich und wird diesen in einem Erfahrungsbericht dokumentieren.

- Mitglieder: Babak, Chiagozie

**Agent:** Die Agentengruppe ist die am stärksten besetzte Gruppe, die sich mit den eigentlichen Agenten befasst. Sie entwickelt eigenständig die für sie relevante Programmierung, Java Dokumentation, Benutzungshandbuch und Testberichte. und nimmt massgeblich an der Schnittstellenentwicklung teil, welche im Administrationshandbuch dokumentiert wird.

[1] Informatik Rechner Betriebsgruppe

- Mitglieder: Daniel, Firas, Babak, Uli

**Monitor:** Die Monitorgruppe befasst sich mit dem Bereich des Softwaremonitors und der GUI. Auch sie entwickelt eigenständig die für sie relevante Programmierung, Java Dokumentation, das Benutzungshandbuch und Testberichte. und nimmt massgeblich an der Schnittstellenentwicklung teil.

- Mitglieder: Christian, Chiagozie, Hayat

**Netzplan:** Die Gruppe Netzplan befasst sich mit dem Netzplanmodul, wozu auch die automatische Netzerkennung zählt. Entwickelt wird die für sie relevante Programmierung, Java Dokumentation, Benutzungshandbuch und Testberichte.

- Mitglieder: Martin, Björn, Alex

**SimulaNT:** Für die Verifizierung bzw. den Vergleich der gemessenen Ergebnisse sollen mittels SimulaNT die von uns analysierten Netze durchgerechnet Werden. Die SimulaNT Gruppe wird sich um Berechnung und Vergleich kümmern.

- Mitglieder: Alex, Martin

**Fachgespräch:** Das Fachgespräch ist neben dem Endbericht der Abschluss der PG und entscheidet über die Scheinvergabe. Die Fachgesprächsgruppe wird es vorbereiten und den Vortrag halten.

- Mitglieder: Chiagozie, Christian, Hayat

**Endbericht:** Die Endberichtgruppe kümmert sich um die Erstellung und Formatierung des Enberichts und kontroliert die von allen PG Mitgleidern eingebrachten Dokumente im Zusammenhang.

- Mitglieder: Björn, Christian, Martin

**Projektmanagement:** Das Projektmanagement befasst sich mit der Planung des Semester, der Erstellung von Arbeitsgruppen und Mitgliederzuweisung und kümmert sich um die Einhaltung von Projektzielen.

- Mitglieder: Björn, Babak

Tabelle 1.1 gibt nochmals eine Übersicht der Zuständigkeiten der einzelnen PG Mitglieder während des zweiten Semesters.

| | AH | BD | BS | CM | CW | DG | HE | FR | MS | UH |
|---|---|---|---|---|---|---|---|---|---|---|
| Gruppe Netzplan | * | | * | | | | | | * | |
| Gruppe Agent | | * | | | | * | | * | | * |
| Gruppe Monitor | | | | * | * | | * | | | |
| Gruppe Einsatplan LS4 | | | | | | * | * | * | | * |
| Gruppe Einsatzplan Unna | | * | | | | | | * | | |
| Gruppe Einsatz LS4 | | | | | | * | * | | | * |
| Gruppe Einsatz Unna | | * | | * | | | | | | |
| Gruppe SimulaNT | * | | | | | | | | * | |
| Gruppe Projektmanagement | | * | * | | | | | | | |
| Gruppe Endbericht | | | * | | * | | | | * | |
| Gruppe Fachgespräch | | | | * | * | | * | | | |

Tabelle 1.1: Zuständigkeiten

### 1.4.9 Dokumentationsrichtlinien

Für unsere PG werden wir folgende Richtlinien für die Quellcode-Dokumentation verwenden.

- Die Beschreibung der einzelnen Klassen und Methoden sollte kurz und präzise sein.
- Sie sollte in der dritten Person geschrieben werden. Z.B. sollte anstatt: „nimm den Namen" „nimmt den Namen" geschrieben werden.
- Der erste Satz sollte eine kurze Zusammenfassung der Methode beinhalten, da dieser später als Überblick angezeigt wird und erst beim draufklicken der ganze Text sichtbar wird.
- Mehrere Paragraphen werden mit dem Tag <p> getrennt.

### 1.4.10 Projektüberwachung

Der Projektfortschritt sollte ständig kontrolliert werden, damit sich der Endtermin des Projektes nicht hinaus verzögert. Falls das Projekt hinter den Zeitplan gerät, müssen unter Umständen Ressourcen und Aktivitäten verändert werden, um dem entgegenzuwirken. Es wurden Strafen bei Überschreitung einer Deadline eingeführt, damit motiviert wird, diese zu vermeiden, um den Projektfluß nicht zu gefährden. Weiterhin wird bei jeder Projektsitzung der Implementierungsstand jeder der drei Gruppen eingeholt, um gegenbenfalls die Arbeitskräfte umzuverteilen, falls bei einer Gruppe größere Probleme aufgetreten sollten. Unter Umständen sollte auch der Projektplan angepasst werden. Dabei kann die Information über tatsächliche und zuvor berechnete Bearbeitungsdauer der einzelnen Aktivitäten für die folgenden Aktivitäten von Hilfe sein, um sie neu zu schätzen.

### 1.4.11 Resümee

Nachdem in den vorigen Abschnitten die Projektplanung beschrieben wurde, soll hier kurz auf den tatsächlichen Projektablauf eingegangen werden. Die einzelnen Gruppen konnten alle geplanten Aktivitäten erfolgreich beenden, allerdings wurden bei fast allen Tasks die hier gesetzten Termine teils geringfügig, teils erheblich überschritten. Neben diversen, als durchaus normal zu bezeichnenden, kleinen Verzögerungen beim Erstellen dieses Endberichtes und der Code Dokumentationen, wurde der gesamte Projektfortschritt durch erhebliche Verzögerungen im Bereich der Monitor-Agent-Schnittstelle um einige Wochen behindert. Dadurch verschob sich vor allem der, letztendlich aber erfolgreiche, Testeinsatz am LS4 und folgend der Einsatz in Unna. Alles in allem konnten die hier geplanten Tasks dann mit ca. einem Monat Verspätung beendet werden.

# Kapitel 2

# Seminar

Nach dem Abschluss des ersten Semester steht fest, was unser Tool können soll und wie es grundlegend arbeitet. Für die Implementierung im zweiten Semester blieben allerdings noch einige wichtige implementierungsspezifische Fragen, insbesondere zu den Themen SNMP, JavaSpaces, Java im allgemeinen, Netzwerkerkennung und explorative Statistik offen.

Die PG 422 hielt zu dem Zweck ein zweitägiges kompaktes Seminar an der Uni Dortmund ab. Jeweils zwei PG Teinehmer hielten einen ca. einstündigen Vortrag über ein Thema. Die Austragung als Kompaktseminar gab den PG Teilnehmen zusätzlich die Möglichkeit, die gewonnenen Erkenntnisse zu diskutieren und letzte Unklarheiten zu beseitigen.

Die nachfolgenden Abschnitte geben die grundlegenden Inhalte der Vorträge in kompakter Form wieder. Für weitere Informationen wird auf die jeweiligen Ausarbeitungen in Form von Folienvorträgen verwiesen, die am Lehrstuhl Informatik IV einsehbar sind.

## 2.1 Explorative Datenanalyse in der Statistik

*von Daniel Grabowski und Chiogozie Madukife*

In der Statistik werden die beiden Teilgebiete deskriptive und explorative Statistik unter dem Begriff „explorative Datenanalyse" zusammengefasst. Die fünf Grundbegriffe der explorativen Datenanalyse sind die statistischen Einheiten, die Variablen (Merkmale), die Grundgesamtheit, die Stichproben und das Messen.

- Die statistischen Einheiten sind Objekte an denen interessierenden Grössen erfasst werden, auch Merkmalsträger genannt.
- Die Variablen (Merkmale) sind die an den statistischen Einheiten gemessenen Grössen.
- Die Grundgesamtheit ist die Menge aller für eine Fragestellung relevanten statistischen Einheiten. Grundgesamtheit kann endlich oder undenlich sein.
- Stichproben sind die tatsächlich untersuchten Teilmengen der Grundgesamtheit.
- Das Messen ist die Erhebung von physikalische Grössen an den statischen Einheiten.

Die Variablen lassen sich unter Verwendung von unterschiedlichen Kriterien einteilen. Zuerst können Variablen entweder als quantitative oder als qualitative Variable bezeichnet werden. Man unterscheidet auch zwischen diskreten und kontinuierlichen/stetigen Variablen. Schließlich gibt es

die Einteilung in manifeste oder latente Variablen. Während die quantitativen Variablen sich durch eine Ordinal-, eine Intervall- oder eine Verhältnisskala messen lassen, werden qualitative Variablen auf der Nominalskala und eventuell bei Verschlüsselung durch Zahlen auch auf der Ordinalskala gemessen.

Die Erhebung von Daten kann in drei verschiedenen Formen erfolgen, nämlich durch Befragung (Fragebogen), Messen (physikalischer Größen) oder durch Verwendung bereits erhobener Daten. Die Erhebung von bereits existierenden Daten kann entweder durch Vollerhebung oder Stichproben erfasst werden. Bei Stichproben kommen die einfache Stichprobe, die geschichtete Zufallsstichprobe, die Klumpenstichprobe, die mehrstufige Stichprobe und das bewusste Auswahlverfahren zum Einsatz.

Die ersten beiden Schritte nach der Datenerhebung sind die Einteilung von Daten in Klassen um einen Überblick über die Verteilung zu bekommen und die Beobachtung der absoluten Häufigkeit. Die relative Häufigkeit wird am einfachsten durch graphische Darstellung erkennbar. Die Art des verwendeten Diagramms hängt hauptsächlich von den interessierenden Merkmalen der Daten ab. Zahlreiche Diagrammtypen, z.B. das Balkendiagramm, das Tortendiagramm und die Kurve, haben sich über die Jahre etabliert.

Bei der Untersuchung von Stichproben gibt es Kenngrößen, um die Verteilung qualitativ oder quantitativ zu beschreiben. Dazu gehören Lageparameter, Quantile, Streuungsparameter zur Beschreibung der Tendenz der Verteilung und Schiefemaße, die den Grad der Asymmetrie der Verteilung charakterisieren. Zur Symmetrisierung, Stabilisierung (Varianzhomogenisierung) und zur Vereinfachung der Interpretation einer Verteilung kann eine Transformation der Daten durchgeführt werden. Ein wichtiges Hilfsmittel stellen in diesem Kontext die Box-Cox-Transformation und die Familie der Potenzleiter-Transformationen dar, mit deren Hilfe man eine gegebene Verteilung auf die jeweils angemessene Weise transformieren kann. Liegen die Daten in Form von Zahlenpaaren (also als Werte zweier Merkmale) vor, und vermutet man einen linearen Zusammenhang, so kann dieser mit dem Korrelationskoeffizienten nach Bravais-Pearson bestätigt oder zerstreut werden. Liegt ein linearer Zusammenhang vor, interessiert die Gerade (Regressionsgerade), die den „wahren Zusammenhang" möglichst gut approximiert. Hierzu ist auch der Bruchpunkt als ein Maß für die (Ausreisser-) Resistenz der angepassten Geraden wichtig.

Eine Methode zur Bestimmung der Regressionsgeraden ist z.B. die Methode der kleinsten Quadrate. Mit Hilfe des Bruchpunktes kann jedoch nachgewiesen werden, dass die durch die Methode der kleinsten Quadrate gebildeten Regressionsgeraden sehr ausreisserempfindlich sind. Resistente Geraden erhält man z.B. durch die Methode der kleinsten absoluten Residuen, oder durch die Methode der kleinsten getrimmten Quadrate.

Literatur: [Kobo00] , [Schm01]

## 2.2 Simple Network Managemant Protocol & RMON Management

*von Alexander Harsch und Ulrich Hilmer*

SNMP ist ein Protokoll, mit dem der Status vieler in einem Netz vorhandener Geräte von einer zentralen Station abgefragt werden kann. SNMP-fähig sind in der Regel alle besseren Switches, Hubs und Router sowie PCs, die über eine entsprechende Softwarekomponente verfügen. In letzterem Fall lässt sich nicht nur die Funktionsfähigkeit der Hardware bestimmen sondern auch der Umfang an Software, die installiert wurde. Auf diese Weise vereinfacht sich auch das Lizenzmanagment erheblich. Das Fernabfragen von Geräten ist erforderlich geworden, nachdem in der IT-Welt die Mainframes, an denen alle Mitarbeiter über ein Terminal gearbeitet haben, nahezu ausgestorben sind. Die heutige IT Infrastruktur vieler Unternehmen besteht aus verteilten Systemen.

Jeder Mitarbeiter hat heute entweder seinen eigenen PC oder einen sog. Networkcomputer auf dem Schreibtisch stehen. Diese Rechner müssen natürlich alle vernetzt werden. Im Laufe der Zeit wachsen diese Netze bis zur absoluten Unübersichtlichkeit an. Dennoch besteht für die Administratoren der Zwang, die Übersicht zu behalten und Ausfälle in den häufig redundant angelegten Netzen zu erkennen und zu beheben. SNMP ermöglicht nun die Überwachung des Netzes und der angeschlossenen Geräte von einem zentralen Arbeitsplatz aus. Mit der entsprechenden Software wird die Überwachung dann zudem soweit automatisiert, dass sich das System selbst meldet, wenn ein Defekt aufgetreten ist. [Kmie01]

SNMP nimmt einen gewichtigen Teil der Arbeit der PG422 an Speedtrap ein, denn nur mit SNMP können die relevanten Größen für die Leistungsbewertung des Netzes von den beteiligten Netzkomponenten (Switches) gewonnen werden. Zusätzlich implementiert die Agentengruppe eine eigene, abgespeckte SNMP Suite für PCs, deren Netzwerkkarte direkt überwacht wird.

### Terminologieabwandelung SNMP PG422

SNMP steht für Simple Network Management Protocol. Es wurde von der Internet Engineering Taskforce (IETF) in den Requests for Comments (RFCs) 1157, 1905, 2271 und weiteren spezifiziert. Jedes dieser Dokumente steht für eine SNMP-Version. Im Laufe der Versionen wurde jeweils der Funktionsumfang erweitert und der Sicherheitsstandard erhöht. SNMP enthält den Begriff Simple, denn das Protokoll wurde bewusst einfach gehalten im Vergleich zum Standard CMIS/CMIP, der ansonsten als Grundlage für Network Managment Software angewendet wird. SNMP verwendet das User Datagram Protocol (UDP) zum Versenden von Daten über das Netz auf den IP Ports 161 für Anfragen und die entsprechenden Antworten und den Port 162 für das Versenden von Traps. Traps sind von den zu überwachenden Geräten selbst ausgelöste Warnmeldungen, wenn z. B. eine wichtige Netzverbindung ausgefallen ist oder die Betriebstemperatur eines Gerätes in den kritischen Bereich kommt. [Cisc03][Nets03]

Zwischen der üblichen Terminologie von SNMP und dem Sprachgebrauch der PG422 gibt es gewisse Unterschiede, auf die im folgenden eingegangen wird. Bei SNMP wird die überwachende Station Network Management System (NMS) genannt. Auf den zu überwachenden Geräten, wie Switches, PCs, Druckern usw. läuft eine Software, die Agent genannt wird. Jeder Agent kennt eine gewisse Anzahl Befehle, die vom NMS abgefragt werden können. Diese Befehle werden für jedes Gerät separat in einer Management Information Base (MIB) gespeichert. Bei Speedtrap wird das Network Management System „Monitor" genannt. Der Monitor stellt die für die Leistungsfähigkeit des Netzes notwendigen Größen graphisch da und leitet die Messaufträge des Anwenders an die installierten Agenten weiter. Von den Agenten gibt es 2 verschiedene Typen. Der erste Typ ist auf einem Arbeitsplatzrechner installiert und überwacht die Netzwerkkarte (NIC). Darüber hinaus ist er in der Lage, auch Lasten in Form von Ping-Paketen zu erzeugen. Der zweite Agententyp wird ebenfalls auf einem Arbeitsplatzrechner installiert. Seine Aufgabe ist allerdings die Überwachung eines SNMP-fähigen Switches. Anfragen gehen also nicht mehr direkt an die Netzwerkkomponente sondern werden von diesem zwischengeschaltetem Agenten bearbeitet. Dazu dekodiert der Agent die Anfrage des Monitors, befragt dann den ihm zugeordneten Switch, bearbeitet die Antwort des Switches und sendet danach die Daten an den Monitor zurück. Dies ist sinnvoll, da viele Messungen über einen längeren Zeitraum stattfinden und der Agent dann die anfallenden Datenmengen vorverarbeitet und danach komprimiert an den Monitor sendet, um die Messungen im Netz durch diesen Overhead nicht zu sehr zu verfälschen. Aus diesem Grunde hat sich die PG auch entschieden, nicht Java Spaces einzusetzen sondern alle Informationen über TCP bzw. UDP Verbindungen zu versenden.

### OIDs

Unter Object Identificators (OIDs) versteht man, welche Register mit Informationen von einem Gerät zur Verfügung gestellt werden und wie man sie abfragt. OIDs werden im Format „Ab-

stract Syntax Notation 1" (ASN.1) gespeichert. Mit Hilfe der MIBs werden die Zahlenketten (z. B. 1.3.6.1.2.1.11) zu Befehlsworten aufgelöst (hier iso.org.dot.internet.mgmt.mib-2.snmp.snmpInPkts). snmpInPkts steht dabei für eine Funktion, die angibt, wie viele snmp-Befehle ein Gerät von der Transportschicht IP bekommen hat. Speedtrap wird nur einen geringen Teil aller OIDs verwenden, die nach SNMPv3 spezifiziert sind und aller Wahrscheinlichkeit keine, die herstellerspezifisch sind. In diesem Fall ist die Kenntnis spezieller MIBs beim Monitor wie den Agenten nicht unbedingt erforderlich. [Alve97]

#### SNMP Befehle

SNMP kennt im Wesentlichen nur 3 Befehle:

**read data:** (z.b. get 192.168.50.177 <communitystring> 1.3.6.1.2.1.11) erfragt von dem Agenten auf dem Gerät mit der IP 192.168.50.177 die OID 1.3.6.1.2.1.11 . Um sich zu legitimieren sendet das NMS als Passwort den communitystring mit. Dieser ist bei der ersten SNMP Version noch im Klartext übermittelt worden während bei fortgeschrittenen Versionen von SNMP eine Verschlüsselung erfolgt.

**write data:** (z.b. set 192.168.45.1 <communitystring> 1.3.6.1.2.1.11 <value>) schreibt den Wert <value> in die Object ID 1.3.6.1.2.1.11. Das ist immer dann sinnvoll, wenn ein Zähler wieder auf Null gesetzt werden soll.

Diese Befehle werden von den Softwarekomponenten von Speedtrap primär verwendet werden. Ausserdem gibt es noch den Trap-Befehl, mit dem sich wie oben schon erwähnt, ein Gerät direkt beim NMS melden kann. Da diese Befehle unmittelbar einen kritischen Zustand melden, sind sie für Speedtrap von geringer Bedeutung, da Speedtrap an sich keine kritischen Zustände auslösen soll, wenn man vom gewollten Flodden eines Switches zur Maximumsbestimmung einmal absieht. Aufbauend auf dem get Befehl ist get-next, mit dem man den gesamten OID-Baum der Reihe nach durchgehen kann. Für Speedtrap ist der get-next Befehl nicht interessant. Inzwischen gibt es viele freie MNS und SNMP-Implementierungen, z. B. net-snmp, dass aus UCD-snmp (University of California at Davis) hervorgegangen ist und dessen Funktionsspektrum den gesamten Funktionsumfang von Speedtrap in Bezug auf SNMP abdeckt bzw. weit darüber hinaus geht. Speedtrap wird allerdings der Portabilität wegen ein Java-Programm werden. Allerdings gibt es auch für Java Klassen, die IP, UDP und SNMP Funktionalität bereit stellen und der PG die Arbeit beim Implementieren vereinfachen werden, sofern diese Klassen als Freeware oder Opensource zur Verfügung stehen. [Sour02]

### RMON

RMON (Remote Monitoring) ist eine Ergänzung zur MIB-2. RMON, sofern vom Gerät unterstützt, ermöglicht sowohl internes als auch externes „polling". Wir können das Gerät also von einer NMS aus abfragen (external polling) oder man lässt den lokalen RMON Agenten sich selbst in regelmäßigen Abständen abfragen und Fehler loggen. Man kann den „Remote Monitor" also so konfigurieren, das er bei bestimmten Alarmen unterschiedliche Aktionen wie Logdateien schreiben, „Traps" senden usw. ausführt. Cisco unterstützt die RMON-Kategorien „Events" und „Alarms". Man kann die Alarme so konfigurieren, das sie auf bestimmte Art und Weise auf steigende oder fallende Werte reagieren. Jeder dieser Schwellwerte hat die Möglichkeit einen internen „Event" auszulösen. Der Unterschied zwischen Alarmen und Events ist wichtig. Jeder Alarm wird an ein bestimmtes Ereignis gekoppelt, das bestimmt welche Aktion ausgelöst wird wenn der Alarm los geht. Solche Aktionen sind Traps an die NMS senden und einen Eintrag ins Log schreiben. Standard SNMP traps sind von dem Hersteller vorkonfiguriert, was dem Manager keine Gelegenheit gibt die Schwellwerte selbst festzulegen. Die Alternative zu „Traps" ist das Logging. Das kann sehr

hilfreich sein, wenn man Konfigurationen für SNMP „Traps" testen will. Ist eine Konfiguration zu sensibel, wird das Netz mit Traps durch die so genannten „trigger-happy" Pakete geflutet. Auch für unsere PG ist RMON eine sehr interessante Angelegenheit. Wir können mit RMON Daten sammeln ohne Netzlast zu erzeugen. Obwohl RMON von allen neueren Geräten unterstützt wird, sind die Agenten aus unserem Framework aber nicht wegzudenken. Wir können uns nicht darauf verlassen, in einer Netzwerkumgebung zu arbeiten, in der von jedem Gerät RMON ausgeführt werden kann. [Doug01],[Perk97]

### SNMP in der Praxis

Schnell und einfach können SNMP-Abfragen mit den im Linux-SNMP Paket enthaltenen Kommandos vorgenommen werden. Die wichtigsten SNMP-Befehle die getätigt werden können sind „snmpget" und „snmpwalk".

Das „snmpget" Kommando liefert den Wert eines Blattes des MIB-Baums zurück, im Beispiel den der System-Beschreibung

```
snmpget localhost public .1.3.6.1.2.1.1.1.0
system.sysDescr.0 = Linux notebook1 2.4.18-bf2.4 #2 Thu Dec 19 09:37:17 CET
```

Das snmpwalk Kommando traversiert den Zweig 1.3.6.1.2.1 (MIB-2), und liefert alle darin enthaltenen Parameter zurück. Die mib-2 hält natürlich wesentlich mehr mögliche Parameter, aber nicht immer ist der Agent so konfiguriert, das er auch alles Preis gibt.

```
snmpwalk localhost public .1.3.6.1.2.1

system.sysDescr.0 = Linux notebook1 2.4.18-bf2.4 #2 Thu Dec 19 09:37:17 CET 2002
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
system.sysUpTime.0 = Timeticks: (700673) 1:56:46.73
system.sysContact.0 = Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
system.sysName.0 = notebook1
...
...
System.sysORTable.sysOREntry.sysORDescr.1 = The MIB module to describe generic ..
...objects for network interface sub-layers
system.sysORTable.sysOREntry.sysORDescr.2 = The MIB module for SNMPv2 entities
system.sysORTable.sysOREntry.sysORDescr.3 = The MIB module for managing TCP ..
...implementations
...
...
End of MIB
```

Wenn wir eine solche Abfrage im Netzwerk protokollieren (sniffen), bekommen wir einen Auszug wie diesen von „ngrep" erstellten. Es ist zu ersehen das SNMP nicht schwer zu knacken ist. Der „Community Name", hier public, ist im Klartext übertragen.

```
interface: eth0 (192.168.100.0/255.255.255.0)
U 192.168.100.99:32795 -> 192.168.100.1:161
  04.....public.'..M.........0.0...+.............B......
U 192.168.100.1:161 -> 192.168.100.99:32795
  0:.....public.-..M.........0.0...+.............B............
```

Der Rechner mit der IP 192.168.100.99 stellt SNMP Agenten 192.168.100.1 auf Port 161 eine Anfrage. Der SNMP Agent antwortet dem selben Port zurück.

Um einen SNMP-Agenten im Netz ausfindig zu machen, muss der SNMP Port 161 mit UDP gescannt werden. Switches und Router haben oft die Ports für HTTP, Telnet (TCP) für die Konfiguration offen, sowie TFTP und SNMP (UDP). Über TFTP können Konfigurations Dateien eingespielt und gesichert werden. Die Scanns bei einem Cisco Catalyst 2900 Series sehen so aus:

```
nmap 192.168.100.1     /Addresse des Switches
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.100.1):
(The 1598 ports scanned but not shown below are in state: closed)
Port       State       Service
23/tcp     open        telnet
79/tcp     open        finger
80/tcp     open        http

Nmap run completed -- 1 IP address (1 host up) scanned in 7 seconds

nmap -sU 192.168.100.1
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.100.1):
(The 1467 ports scanned but not shown below are in state: closed)
Port       State       Service
161/udp    open        snmp

Nmap run completed -- 1 IP address (1 host up) scanned in 1246 seconds
```

## 2.3 Analyse von JavaSpaces für den Einsatz verteilter Messungen

*von Firas Raed und Björn Schmidt*

Bei der Netzwerkmessung mittels Speedtrap ist es unter anderm auch wichtig, die Messungen durch notwendige Netzwerkkommunikation unter den verteilen Systemkomponenten nicht oder zumindest so gering wie möglich zu beeinflussen.

Vom PG Veranstalter wurde JavaSpaces als Kommunikationskanal vorgeschlagen. Ziel der JavaSpaces Analyse ist es, herauszufinden, ob JavaSpaces eine - vor allem unter dem o.g. Aspekt - für Speedtrap sinnvolle Kommunikationsmöglichkeit bietet.

### C/S Architekturen, Tuple Spaces und Java Spaces

Während klassische C/S Architekturen Vorteile durch paralleles Rechnen, Redundanz und Ausfallsicherheit und Kostenvorteile gegenüber Supercomputern bieten, haben sie allerdings auch Nachteile durch Teilausfälle, evtl. blockierte verteilte Ressourcen, erhöhten Wartungsaufwand und schwierige Konsitenzerhaltung.

Tuple Spaces versuchen viele der Probleme zu lösen. Ein TupleSpace (Abbildung 2.1) ist ein virtueller Raum, ähnlich einem Shared Memory, in bzw. aus dem Tuple von Daten geschrieben, gelesen und wieder gelöscht werden können. Durch atomare Operationen und gegenseitigen Ausschluss durch Semaphoren ist es verteilten Prozessen möglich, Daten über den TupleSpace auszutauschen. Dabei können beim lesen (Befehl *Out*) und schreiben (Befehl *In*) von Tupeln neben „normalen" Werten oder Parametern auch sogenannte „formale" Parameter, also lediglich eine Typangabe benutzt werden. Die Benutzung des Tupleraums T ist mit folgenden atomaren Anweisungen möglich:

- out t bewirkt T´=T+t
- in t bewirkt t´=x, sobald ein zu t passendes x in T vorhanden ist, ausserdem T´= T - x

Die Elemente von x sind Werte, die von t sind Ausdrücke. t fungiert als Muster (template, pattern) für die Tupel-Auswahl.

Die Musteranpassung bei in t ist dann wie folgt:

(x1,...,xn) passt zu (t1,...,tm), wenn

- n=m
- die Typen von xi und ti für alle i gleich sind
- xi=ti

Ein durch *Out(4,i:integer)* geschriebenes Tuple wird dann sowohl per *In(i:integer,77)* als auch per *In(4,88)* gefunden, wobei *i:integer* in dem Fall einen formalen Parameter darstellt.

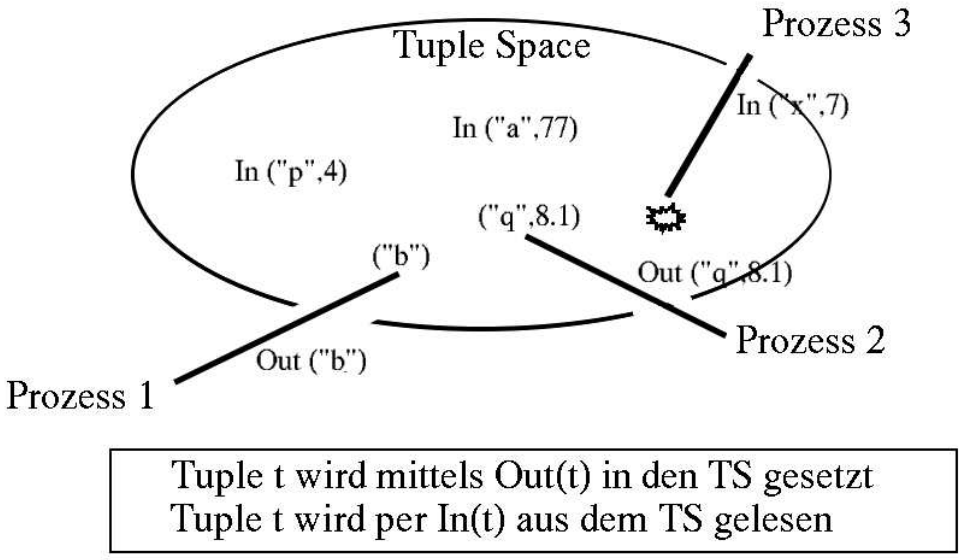



Abbildung 2.1: Tuple Spaces

Neben der Programmiersprache Linda, die TupleSpaces implementiert, aber Tuple immer auf einem einzigen TupleSpace Server darstellt, wobei die "Clients" per Broadcast Tuple anfordern, stellt JavaSpaces eine konsequente Weiterentwicklung von TupleSpaces dar. Während unter Linda Soft- und Hardwarefehler zum Crash des Systems führen, ist JavaSpaces durch fehlertolerante Auslegung, Einbeziehung von Sicherheitsaspekten und die Möglichkeit nicht nur Tuple, sondern ganze Objekte in den Space zu schreiben, besser an heutige Bedürfnisse angepasst. JavaSpaces greift dazu auf die Java Objektserialisierung zurück, durch die Objekte rekursiv in Byteströme verwandelt werden, und so in den TupleSpace geschrieben werden können.

### JavaSpaces aus Benutzersicht

Aus Benutzersicht ist JavaSpaces ein extrem einfaches Konzept. Ein JavaSpace ist ein Objekt, welches die Funktion eines „schwarzen Brettes" innerhalb eines verteilten Systems einnimmt. Ein Space (Abbildung 2.2) ist ein fortdauerndes Objekte-Lager, das von verschiedenen Prozessen gemeinsam genutzt wird und über das Netzwerk zugänglich ist. Der Benutzer betrachtet nur die Objekte, die in und aus dem JavaSpace migriert werden und kann über sogenannte Entry Objekte gewünschte Daten im Space suchen. Entry Objekte sind im Prinzip Interfaces, die die Struktur eines gesuchten Objektes wiedergeben. Um die Regelung der Persistenz und Konsistenz, die Deadlockbehandlung und Sicherheitsaspekte muss sich der Benutzer nicht selber kümmern.

Die folgenden Operationen kann man in einem JavaSpace ausführen:

**write:** Ein Entry in den *JavaSpace* schreiben

**read:** Nachschauen, ob ein bestimmtes Entry im *JavaSpace* liegt

**take:** Ein bestimmtes Entry aus dem *JavaSpace* entfernen

**notify:** Eine Benachrichtigung bekommen, wenn ein Entry von Interesse in den *JavaSpace* geschrieben wird

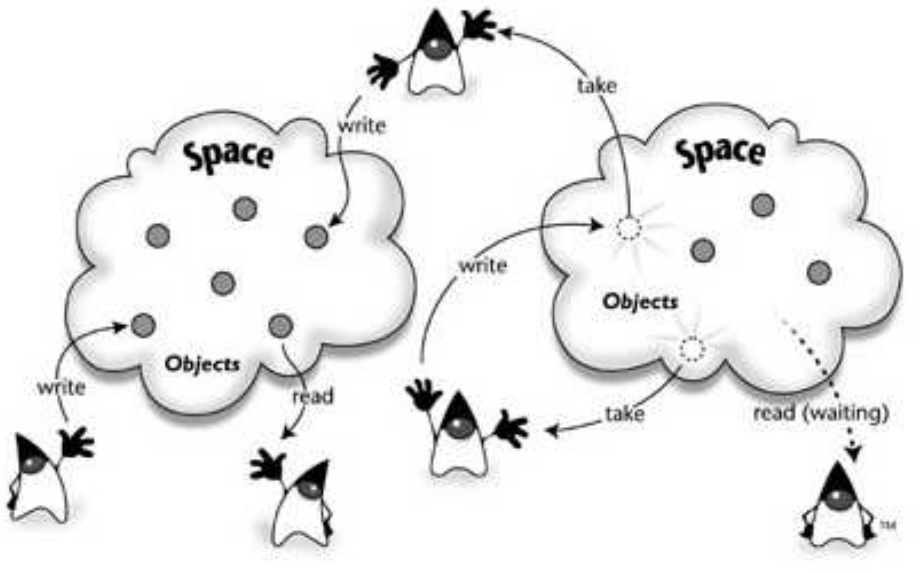



Abbildung 2.2: Java Spaces, aus [JSpa01]

Die JavaSpaces-Technologie ist also ein einfaches und zugleich mächtiges Tool. Die Eigenschaften der Spaces leiten sich aus den Eigenschaften der Jini-Technologie, welche im nächsten Abschnitt beschrieben werden, her:

**Spaces sind Gemeinschaftsräume:** Spaces können über das Netzwerk betreten werden. Sie sind frei verfügbarer, gemeinsamer Speicherplatz, in dem der gleichzeitige Zugriff geregelt ist.

**Spaces sind fortdauernd:** Spaces bieten zuverlässigen Lagerraum für Objekte. Befindet sich ein Objekt erst einmal in einem Space, so bleibt es dort, bis es ausdrücklich entfernt wird, oder bis seine Leasing-Zeit verstrichen ist.

**Spaces sind verbindend:** Objekte werden in Spaces per „associative lookup" gefunden. Unwichtig ist, wie das Objekt heisst, wer es geschaffen hat, oder wo es gespeichert ist, lediglich sein Inhalt ist von Interesse. Das passende Objekt wird über Template-Matching gefunden.

**Spaces sind sicher durch Transaktionen:** Spaces unterstützen atomare Operationen. Entweder wird die gesamte Operation, die aus einzelnen Unteroperationen bestehen kann, ausgeführt, oder gar nicht. Transaktionen können sich auch über mehrere Spaces erstrecken.

**Spaces dienen dem Austausch ausführbarer Programme:** Ein Objekt in einem Space ist passiv. Sobald es jedoch aus dem Space herausgenommen wurde, können seine Methoden aufgerufen werden und / oder es verändern.

### Jini, RMI und JavaSpaces

Um zu verstehen, wie Objektserialisierung und RMI bei der JavaSpaces Technologie zusammenarbeiten, ist ein grober Jini Überblick nötig. Allerdings kann an dieser Stelle lediglich eine sehr kompakte Beschreibung folgen. Zum besseren Verständnis wird auf die Literaturliste verwiesen.

Remote Method Invocation oder kurz RMI ist das Java System für verteilte Anwendungen. Mit RMI können Methoden von Objekten auf entfernten Rechner ausgeführt und deren Rückgabeparameter erhalten werden. RMI nutzt zu diesem Zweck die oben angesprochene Serialisierung, um Methodenaufrufe zu codieren. Für die Kommunikation über das Netz, wird unterhalb der Serialisierung ein Java eigener Netzwerkstack aus Stub/Skeleton Layer und Remote Reference Layer oberhalb von TCP verwendet.

Jini[1] (Abbildung 2.3) erweitert nun das RMI Konzept durch Discovery und Lookup Mechanismen. Jini ist eine Plattform, in der Dienste von verschiedenen "Servern" angeboten werden. Diese Dienste können einfache Aufgaben ähnlich der RMI Remote Objekte übernehmen, andererseits z.b. auch durch Jini fähige Kaffeemaschinen - Der Dienst ist in dem Fall Kaffee kochen - angeboten werden. Die eigentliche Erweiterung zu RMI besteht darin, dass Jini Lookup Server zur Verfügung stellt, in denen sich Dienstanbieter registrieren und von Clients gefunden werden können. Ein Client sucht dann bei dem Lookup Server nach einem Dienst, bekommt die Informationen, ob und wo der gesuchte Dienst zu finden ist und bekommt das Skeleton (in Jini ProxyObjekt genannt) dieses Dienstes nach Anfrage vom ihm geliefert. Das Objekt wird dann lokal ausgeführt, ist also selber lauffähig. Für die funktionsweise von Jini sind natürlich Lookup Services, also Jini Server nötig.

JavaSpaces ist nun Java intern einfach ein Jini Dienst, der anbietet, Objekte entgegenzunehmen und im Space zu speichern. Nach der Suche nach dem JavaSpace Service und der Übertragung des ProxyObjekts können diesem also zu speichernde Objekte übergeben werden. Das Proxy Objekt überträgt diese "Nutzdaten" wiederum durch die oben angesprochene Serialisierung zu seinem Dienstserver und zwar wiederum per Serialisierung der entfernten Methodenaufrufe (hier wird also gleich zweifach serialisiert).

### JavaSpaces und der Netzwerkverkehr

Für die PG422 ist es von entscheidender Bedeutung, welche Netzwerklasten die im letzten Abschnitt vorgestellte Jini/JavaSpaces Technologie im Bezug auf die relevante Anwendung Speedtrap erzeugt. Es wird davon ausgegangen, dass Jini und JavaSpaces auf dem zentralen Monitorrechner installiert werden. Interessant ist dann die Netzwerkkommunikation zwischen Monitor und

[1] Jini: "jee'nee" Sun's Java-basiertes System für verteilte Anwendungen

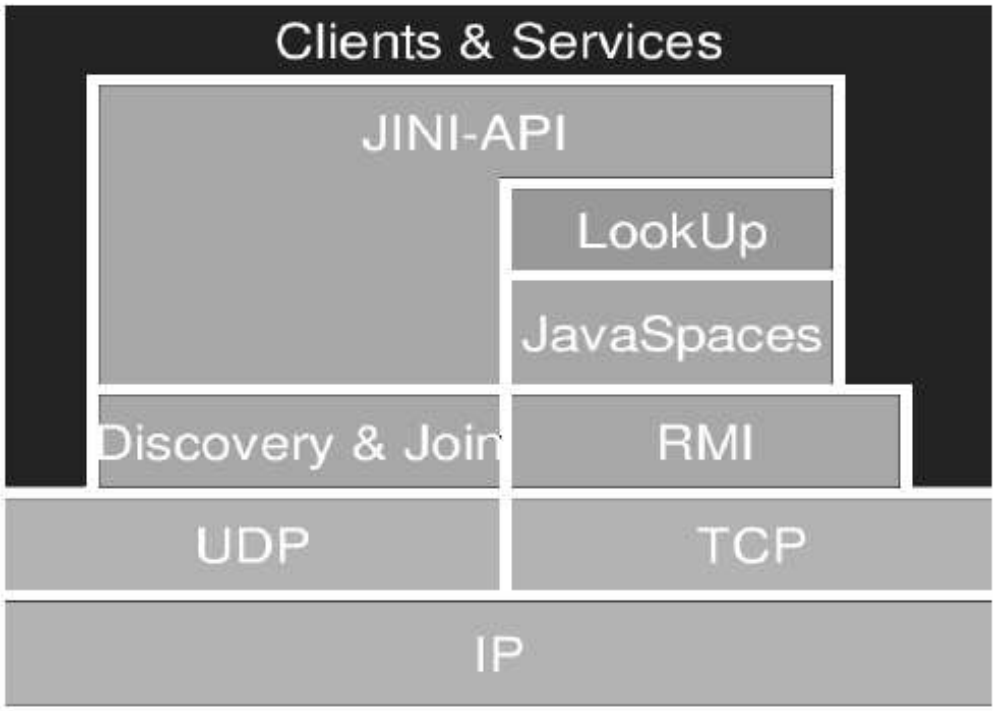



Abbildung 2.3: Der Jini Stack

Agenten. Um dazu qualitative Aussagen zum Ressourcenverbrauch machen zu können wurde hier auf eine Studie der TH Zürich (vergl. [Schw02]) zurückgegriffen. Um die daraus gewonnenen Informationen quantitativ zu untermauern, wurde die Java Stream Grammatik (vergl. [JSpa01]) analysiert. Die Kommunikaton lässt sich demnach aufteilen in:

**Suche nach dem JavaSpace Service:** Hierbei wird das Multicast Request Protokoll verwendet. Dieses versendet pro Lookup ein einziges 512 Byte UDP Paket, der Netzwerkverkehr ist also vernachlässigbar.

**Übertragung der JavaSpace Proxyobjekte an die Agenten:** Hierzu zählt in dem Fall auch die Proxy Registrierung beim Jini Server. In Messungen ergab sich dafür eine Aufkommen von ca. 2kByte für das Auffinden des Service und ca. 5kByte für die Service Registrierung. Diese Werte skalieren linear mit der Anzahl der Services, da bei der PG422 aber nur ein Service benutzt wird ist dies eher uninteressant. Das Datenaufkommen für die eigentliche Übertragung der Proxy Objekte hängt natürlich von deren Größe ab. Werte zwischen 5kByte und 50kByte sind möglich. Durch das Jini Leasing Modell wird zudem die mehrfache Übertragung der Proxy Objekte während einer Messung nötig. Insgesamt handelt es sich allerdings um einmalige oder nur gelegentliche Aktionen, die für die PG422 nicht die wirklich großen Probleme darstellen können und somit auch vernachlässigt werden können.

**Senden von Messkommandos über den JavaSpace:** Dies kann einerseits durch RMI, oder auch per JavaSpaces geschehen. In beiden Fällen wird, wie im letzten Abschnitt schon angedeutet RMI und die dazugehörige Objektserialisierung verwendet. Dieser Punkt kann mit dem folgenden zusammengefasst werden.

**Senden der Messdaten durch die Agenten:** Hiervon wird bei der Anwendung Speedtrap stark Gebrauch gemacht. Die Analyse konzentriert sich also auf diesen Punkt. Um die Messdaten in den Space zu schreiben müssen diese serialisiert werden und dann per RMI (JavaSpace intern) verschickt werden, wobei wiederum der RMI Methodenaufruf serialisiert wird. Messungen der TH Zürich ergaben, dass für die Übertragung eines einzelnen Integer Wertes durch Serialisierung und Protokolloverhead 575 Byte Netzlast entstehen. Überträgt man mehrere Werte in Folge (ein Messdatum von Speedtrap hat ca. 50 Byte Umfang) ergibt sich ein linearer Zusammenhang mit Faktor 40, also ca. 4000% Overhead. Dies lässt sich zum einen durch den Overhead der Objektserialisierung erklären, zum anderen durch den Jini Protokollstack aus JRMP (Java Remote Protocol) und RTL (RMI Transport Layer). Auf den Protokolloverhead soll hier nicht weiter eingegangen werden, was betrachtet wird, ist die von Java zur Serialisierung verwendete Streamgrammatik.

Die Java Stream Grammatik legt fest, wie ein Objekt in einen Stream verwandelt wird. Eine genaue Beschreibung macht an dieser Stelle keinen Sinn, es wird auf den entsprechenden Folienvortrag verwiesen. Bei Analyse der Serialisierung einer einfach verlinkten Liste mit 2 Integer Werten ergibt sich jedoch, dass nach der Serialisierung 69Byte Datenvolumen für die 4 Nutzdaten Bytes entstehen. Hierbei ist einmaliger Overhead enthalten. Jede weiteren 2Byte (ein Integer) erzeugen allerdings mindestens weitere 10Byte Datenvolumen. Der so gezeigte Faktor 5 bzw. die 400% Overhead im Best Case sind ausreichend um JavaSpaces für das Projekt Speedtrap als ungeeignet einzustufen. Eine reine UDP Kommunikation zur Übertragung der Werte liefert, den Netzwerkverkehr betreffend, wesentlich bessere Werte. So liegt der Nutzdatenanteil hier bei ca. 96%, was einem Faktor von 1,04 entspräche (4% Overhead). Auf UDP Kommunikation soll an dieser Stelle allerdings nicht weiter eingegangen werden.

#### JavaSpaces und die PG422

Viele Schwierigkeiten der gemeinsamen Nutzung eines Speichers würden der PG422 durch den Einsatz von JavaSpaces also abgenommen. Java Spaces kann genutzt werden, um die Messergebnisse zu speichern und um die Kommunikation zwischen Monitor und Agenten zu koordinieren. Ein JavaSpace Server kann als Speicherraum benutzt werden, auf den viele Agent-Prozesse zugreifen, die dann parallel an den Daten arbeiten können. Der Monitor-Prozess generiert eine Anzahl Tasks und schreibt diese in ein JavaSpace. Agent-Prozesse beobachten den Space und entnehmen Aufgaben sobald diese anfallen, also im Space abgelegt werden. Im Prinzip ist JavaSpaces in unserem Projekt also einsetzbar, vor allem würde das Speicherproblem elegant gelöst werden.

Für den Einsatz für unser Messtool Speedtrap scheint JavaSpaces dennoch ungeeignet. Die Analyse der Java Stream Grammatik ergab, daß alleine durch die Java Objektserialisierung ein Overhead von 400% zusätzlichem Datenverkehr entsteht. Hinzu kommt der Overhead der diversen Jini bzw. RMI Protokollschichten. Messungen haben gezeigt, das der tatsächliche Overhead sogar bei 4000% liegt, was für unsere Anwendung nicht aktzeptabel ist. JavaSpaces bietet viele Vorteile für verteilte Algorithmen. In unserem speziellen Fall ist allerdings eine reine UDP-Kommunikation zur notwendigen Übermittlung der Messdaten die deutlich sinnvollere Alternative, da es bei der Messung darauf ankommt, die gemessenen Werte selber so wenig wie möglich zu beeinflussen.

Weiterführende Literatur zu diesem Thema: [Schw02], [JSpa01], [SunM99].

## 2.4 Wie arbeiten Meßtools zur automatischen Erkennung von Netzwerkkomponenten?

*von Babak Davani und Hayat El-Fassi*

Dieser Extended Abstract stellt eine kurze Einführung in die Methoden der automatischen Netzwerkerkennung vor. Im Anschluss wird das Programm HP OpenView vorgestellt, welches eins der bekanntesten Tools zur Netzwerkerkennung ist.

Mit „Scanning" wird das Erkennen von Geräten im Netzwerk bezeichnet. Es wird ermittelt, welche Geräte erreichbar sind, mit welcher Art von Systemen man es zu tun hat und welche Dienste auf den Systemen laufen. Dabei werden Methoden wie „Ping sweeps", und „Port scans" benutzt.

„Ping" verwendet ICMP Pakete um zu ermitteln ob eine Ziel-IP-Adresse erreichbar ist. Dabei wird eine ICMP ECHO Anfrage ans Zielsystem gesendet (ICMP type 8) und auf eine ICMP ECHO Antwort gewartet (ICMP type 0). Falls eine solche Antwort empfangen wurde, heißt das, dass der Host erreichbar ist. Falls keine Antwort empfangen wurde, kann es entweder sein, dass das System nicht erreichbar ist, es kann aber auch nicht ausgeschlossen werden, dass diese ICMP ECHO aus Sicherheitsgründen gefiltert werden. Werden mehrere Anfragen an viele Hosts gesendet wird dies als „Ping Sweep" bezeichnet.

Im nächsten Schritt geht es darum zu erkennen, welche Dienste auf dem Zielrechner laufen, die an einem TCP- oder UDP-Port auf Pakete warten. Dies wird als „Port Scanning" bezeichnet.

Das UDP (User Datagram Protocol) ist zwar einfacher als das TCP (Transmission Control Protocol), aber dafür weniger geeignet um das Netzwerk zu scannen. Z.B. brauchen offene Ports keine Bestätigung auf eine Anfrage zu senden. Nicht offene Ports müssen nicht unbedingt eine Fehlermeldung senden, dass der Port geschlossen ist. Glücklicherweise melden die meisten Hosts aber, dass deren Ports geschlossen sind. Bei einem UDP-Portscan wird ein UDP-Paket zum Ziel-Port übertragen. Wenn der Ziel-Port mit der Nachricht „ICMP Port unreachable" den Erhalt quittiert, ist der Port inaktiv und somit geschlossen. Falls diese ICMP-Nachricht nicht an den scannenden Clienten zurückgeschickt wird, kann eventuell davon ausgegangen werden, dass der Port offen ist.

Das TCP Protokoll ist im Gegensatz zum UDP Protokoll verbindungsorientiert. Vor der eigentlichen Datenübertragung findet deshalb ein drei Wege Handshake statt. Erst nach der erfolgreichen Verbindung werden Daten übertragen und abschließend wird durch vier weitere Meldungen die Verbindung wieder getrennt. Die einfachste Art des Portscannens ist der vollständige TCP-Connect-Scan, bei dem eine echte Verbindung zum Ziel-Port zustande kommt. Die Prozedur des 3-Wege-Handshakes von TCP-Verbindungen wird dabei komplett abgearbeitet. Der Client überträgt also zunächst ein SYN-Paket an den Server, dieser quittiert den Empfang mit einem SYN-/ACK-Paket und wartet dann auf eine weitere Bestätigung von Seiten des Clients durch ein ACK-Paket. Diese Scanmethode ist die langsamste, kann jedoch ohne Administrationsrechte durchgeführt werden. Bei einem „halb offenen" Scan (TCP SYN Scanning) wird keine vollständige Verbindung mit dem Server hergestellt. Es wird zunächst wieder ein SYN Paket zum Server geschickt, wenn dieser mit SYN/ACK antwortet handelt es sich i.A. um einen offenen Port und die noch nicht vollständige Verbindung wird vom Client sofort durch senden eines RST/ACKPaketes unterbrochen. Sendet der Server ein RST/ACK zurück, so ist der Port geschlossen.

Es existieren noch eine Vielzahl von weiteren Scan-Methoden wie Stealth-, Xmas-, Fin-Scan, usw., die aber speziell dafür geeignet sind, um unerkannte Scans durchzuführen.

Um Betriebssysteme zu erkennen wird ausgenutzt, dass Betriebssysteme unterschiedliche Initialwerte für die TTL (Time to live) und der TCP Fenstergröße in den zu versendenden Paketen setzen. Z.B. benutzt Windows NT 4.0 Service Pack 6 ein Initialwert von TTL 128 für TCP und 32 für ICMP Pakete.

Um ein Switch im Netzwerk zu erkennen, kann per SNMP geprüft werden, ob das System auf eine SNMP Anfrage antwortet. Falls nicht, ist entweder das Gerät nicht SNMP-fähig, oder die Funktion wurde deaktiviert. Falls der Switch keine SNMP Unterstützung bietet, kann nur mittels „sniffing" die Netzwerktopologie ermittelt werden. Dies ist aber weniger geeignet wegen der Einsatzgebiete des Netzwerkerkennungs-Tool (Polizei, IRB). Bei der SNMP-Abfrage wird getestet, ob der Port 161 offen ist und eine Antwort gesendet wird. Dieses Pakete müssen unter umständen mehrmals gesendet werden, falls keine Antwort geschickt wird, da UDP nicht verbindungsorientiert ist. Falls

eine SNMP-Antwort gesendet wird, kann in der MIB unter anderem ermittelt werden, um welchen Gerätetyp es sich handelt, und welche Hosts an diesem Gerät angeschlossen sind.

Mit all diesen Informationen über das Netzwerk kann dann die Netzwerktopologie grafisch dargestellt werden. Im folgenden wird das HP Open View vorgestellt, welches die vorgestellten Methoden zur Netzwerkerkennung verwendet.

### HP OpenView

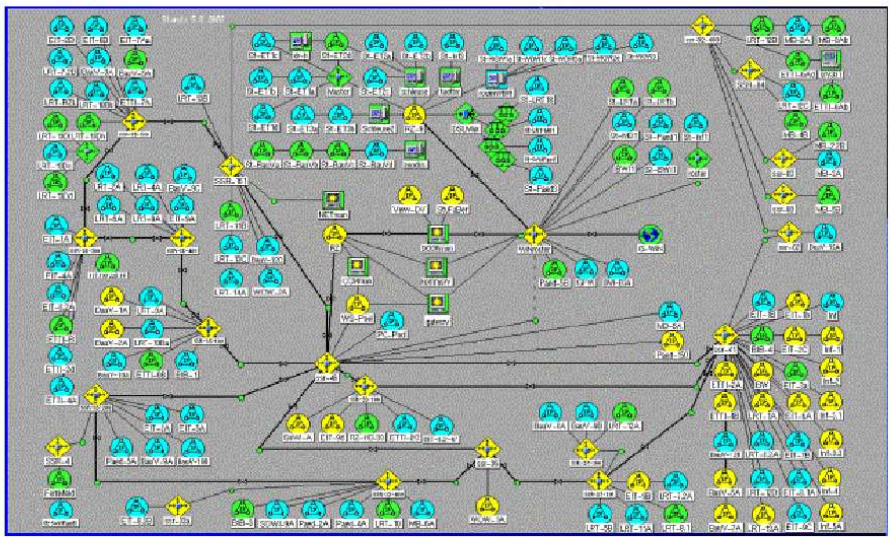


Abbildung 2.4: HP OpenView- Map

HP-Open View ist die meist verwendete Netzwerkmanagement Software. 70% des Internets wird von OpenView gemanagt. Das Basismodul von Open View ist der Network Node Manager NNM. Er unterstützt SNMP, CMIP[2] und CMIP über TCP/IP. Das Topologiemanagement von NNM für IP-Netze umfasst die automatische Entdeckung der Netzwerkkomponenten (auto-Discovery) und die automatische Darstellung der Netzplans (auto-mapping). Sie ermöglichen eine aktuelle Auflistung aller Netzwerkgeräte, und die dazugehörige graphische Darstellung.

Die Hintergrundprozesse bzw. Dienste heissen **ovstart, ovstop und ovstatus.** Sie überwachen kontinuierlich den Status des Netzwerkknotens. Sie aktualisieren ständig die Daten in der OpenView Datenbank. Die zentrale Rolle bei der IP-Discovery spielt der „Netmon" Dämon. Dieser „entdeckt" IP-Komponenten(Discovery-Prozess). Dafür verwendet Netmon u.a. den ARP-Cache, ICMP-Requests (ping), die MIBs von bereits bekannten Komponenten über SNMP.

[2] CMIP-Protokoll = common management information protocol CMIP ermöglicht den Austausch von Informationen und Kommandos über das Netzwerkmanagement zwischen zwei gleichberechtigten Anwendungen auf gleicher Ebene

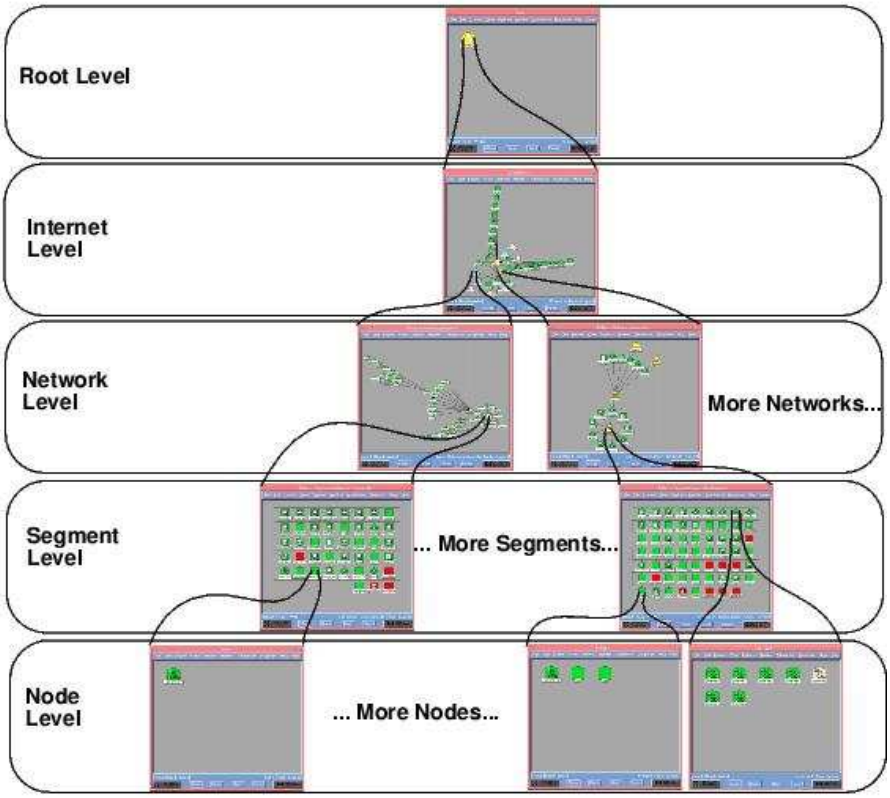



Abbildung 2.5: HP OpenView- Topologie Map

Die MIB II liefert hierbei die relevantesten Informationen. Um die effiziente und effektive Durchführung und Aktualität des Topologiemanagements zu gewährleisten, ist eine Konfiguration der SNMP-Parameter und die Konfiguration der Polling-Intervalle nötig, um z.B. die Statusmeldung und die Feststellung einer Konfigurationsänderung von Ressourcen (z.B. periodisches Testen der Erreichbarkeit) von wichtigen Knoten zu erreichen. Ein kompletter Discovery-Prozess würde eine grössere Last im Netz erzeugen als der Einsatz der Polling-Technologie. Ein Discovery-Prozess wird nur zu Beginn für die automatische Erstellung der Map durchgeführt.

Nachdem die Netzwerkkomponenten erkannt wurden, werden sie als Objekte mit ihren Attributten in der Datenbank gespeichert. Ein Objekt existiert nur einmal in der Objektdatenbank, unabhängig davon, wie oft seine Symboldarstellung auf den verschiedenen Submaps vorhanden ist. Ein Objekt kann von mehreren Symbolen dargestellt werden, aber ein Symbol repräsentiert immer nur ein einziges Objekt. Ein Symbol ist die graphische Darstellung eines Objektes. Zu der jeweiligen Literatur siehe ([HPOp97]).

Eine Submap ist ein sichtbares Fenster mit den Symbolen und Symbolverknüpfungen auf einer niedrigeren Ebene der Netzwerkhierachie.

Eine Map (Abbildung 2.4) ist eine Ansammlung von sichtbaren Symbolen, die zusammen eine hierachische Ansicht eines Netzwerkes bilden. Wenn eine Map in HP OpenView geöffnet wird, wird sie synchronisiert, d.h. sie enthält alle Aktualisierungsfunktionen und -informationen über Veränderungen, die seit dem letzten Öffnen der Map eingetreten sind. Bei jeder Sitzung ist jeweils immer nur eine Map zur gleichen Zeit geöffnet.

Die hierachische Darstellung (Abbildung 2.5) des Netzes mit seinen Segmenten und Subnetzen

erfolgt in 4 Ebenen:

- Internet Level : IP-Netze und Gateways
- Network Level : Gateways, Router, Hubs, Subnetze
- Segment Level : Hosts, Gateways, Router, Hubs, Bridges
- Node Level : Interfaces

## 2.5 Grundlagen und Techniken der Java-Programmierung

*von Martin Schindler und Christian Winkler*

Da den meisten PG Teilnehmern Java bereits aus ihrem Software Praktikum bekannt ist, gibt dieser Beitrag keine Einführung in die Programmiersprache Java, sondern eine Übersicht über die Grundlagen und Techniken der Java Programmierung. Techniken, die nicht so häufig angewandt werden, sollen durch diesen Beitrag wieder in Erinnerung gerufen werden, so dass sie bei der Implementierung wieder im Bewusstsein sind. Insbesondere werden auch Techniken angesprochen, die für den Einsatz bei Speedtrap nützlich erscheinen.

Zu den Grundlagen der Java Programmierung gehören die Konventionen, nach denen Variablennamen mit einem Kleinbuchstaben beginnen und jedes weitere Wort in diesem Namen mit einem Großbuchstaben beginnt (z. B. `derNetzplan`). Ebenso werden Klassennamen behandelt, nur dass sie generell mit einem Großbuchstaben beginnen (z. B. `NetzplanDatentypen`). Konstanten werden vollständig in Großbuchstaben notiert, wobei einzelne Worte durch einen Unterstrich (_) getrennt werden (z. B. `EURO_FAKTOR`). Die von Java reservierten Schlüsselworte dürfen bei der Namensvergabe nicht verwendet werden.

Klassen verfügen in Java über einen oder mehrere *Konstruktoren*, deren Name identisch mit dem Klassennamen ist und die keinen Rückgabetyp haben. Verschiedene Konstruktoren müssen sich in ihrer Parameterliste unterscheiden. Wird kein Konstruktor definiert, so erstellt Java einen Standardkonstruktor. Mit der *Signatur* einer Methode wird die Kombination aus Name, Parameterliste und Rückgabetyp einer Methode bezeichnet. In einer Klasse dürfen keine Methoden mit gleicher Signatur vorkommen. Methoden in Subklassen überschreiben die Methoden mit gleicher Signatur in ihrer Superklasse. Der Zugriff auf die Originalmethode ist mit `super.methode()` möglich. Attribute und Methoden können als Instanz- oder Klassen- Attribute/ Methoden definiert werden. Bei Instanzattributen wird je eine Ausprägung je Objekt gespeichert, während bei Klassenattributen nur eine Ausprägung für die ganze Klasse gespeichert wird. Klassenattribute werden mit dem Schlüsselwort `static` deklariert und der Zugriff von außerhalb des Objekts erfolgt mit `Klassenname.Variablenname`. Analoges gilt für Klassenmethoden. *Abstrakte Klassen* werden mit dem *Modifizierer* `abstract` deklariert und dienen als Basisklasse für die Vererbung. Von abstrakten Klassen können keine Objekte erzeugt werden. Sie stellen grundlegende Funktionen bereit, die aber nur sinnvoll genutzt werden können, wenn sie von Subklassen ergänzt werden. Weitere Modifizierer sind: `public, protected, private, final, native, synchronized, transient` und `volatile`.

*Interfaces* stellen eine Alternative für die in Java nicht mögliche Mehrfachvererbung dar, da Klassen zwar nur von einer Klasse erben, aber mehrere Interfaces implementieren können. Dabei können Interfaces ihrerseits mehrere Interfaces erweitern. Eine Klasse, die ein Interface implementiert, muss alle in diesem Interface deklarierten abstrakten Methoden implementieren. In Interfaces deklarierte Attribute sind implizit `public static final` und somit Konstanten, die initialisiert werden müssen. Methoden sind implizit `public abstract` und müssen überschrieben werden.

Java stellt einen Mechanismus zur Behandlung von *Ausnahmen* bereit. Dabei liegt folgendes Grundprinzip zugrunde: Eine Methode einer Klasse erkennt eine Ausnahmesituation, erzeugt ein Objekt einer Ausnahmeklasse und löst damit die Ausnahme aus. Die Ausnahme kann entweder in derselben Methode abgefangen und behandelt werden (*catch*), oder sie kann an die aufrufende Methode weitergereicht werden (*throw*). Die Methode, an die die Ausnahme weitergereicht wurde, hat nun die Möglichkeit, sie entweder abzufangen, oder sie ihrerseits weiterzureichen. Wird die Ausnahme nie behandelt, sondern immer nur weitergereicht, bricht das Programm mit einer Fehlermeldung ab.

Das `java.io`-Paket stellt Klassen für das Arbeiten mit Ein- und Ausgabestreams bereit. Die Klasse `File` repräsentiert dabei einen File- oder Verzeichnisnamen und stellt Funktionen bereit, um mit diesem zu arbeiten. Die Klassen `Reader` und `Writer` sind abstrakte Klassen, die als Basisklassen für die Klassen dienen, die zeichenorientierten Zugriff über Streams realisieren. Die korrekte Transformation von Unicode-Zeichen wird von diesen Klassen gewährleistet. Die byteorientierten Alternativen sind die abstrakten Klassen `InputStream` und `OutputStream` und ihre Subklassen, zu denen auch `ObjectInputStrem` und `ObjectOutputStream` gehören, die die *Objektserialisierung* in Java implementieren. Bei der Objektserialisierung wird der Zustand eines Objekts, mit allen Objekten, auf die in ihm verwiesen wird, rekursiv in eine Datei gespeichert, wobei unendliche Rekursionen verhindert werden. Als `transient` deklarierte Attribute werden dabei nicht berücksichtig. Um serialisiert werden zu können, muss eine Klasse das `Serializable` Interface implementieren.

Java unterstützt *Multithreading*. Dabei existieren in Java zwei Alternative um Threads zu erzeugen. Zum einen kann eine Subklasse von `Thread` erzeugt werden und zum anderen kann eine Klasse das `Runnable` Interface implementieren. In beiden Fällen muss die `run()` Methode implementiert werden, die die Anweisungen enthält, die der Thread ausführen soll. Im zweiten Fall muss anschließend ein `Thread`-Objekt erzeugt werden, dass das Objekt, das die `run()` Methode enthält übergeben bekommt. Die Methode `start()` startet einen Thread, `join()` wartet auf dessen Ende und `sleep(long m)` hält einen Thread für `m` Millisekunden an. Um Objekte vor *konkurrierendem Zugriff* verschiedener Threads zu schützen, können Methoden und Anweisungsblöcke `synchronized` definiert werden. Java sorgt dann dafür, dass zu einem Zeitpunkt immer nur ein Thread auf die Ressource zugreifen kann.

Für grafische Darstellungen bringt Java zwei Toolkits mit. Zum einen das *Abstract Window Toolkit* (*AWT*, `java.awt`) das für die Darstellung Betriebssystemkomponenten nutzt und zum anderen *Swing* (`javax.swing`), bei dem die grafischen Komponenten komplett in Java programmiert und deshalb plattformunabhängig sind. Für nicht grafische Dinge benutzt Swing jedoch auch noch AWT-Komponenten (z. B. für die Ereignisbehandlung). Das Ereignismodell von Java basiert darauf, dass jedes *Ereignis* (*Event*), wie zum Beispiel ein Mausklick, mit einer *Ereignisquelle* verbunden ist (z. B. ein Fenster). Die Reaktion auf ein Ereignis erfolgt dann durch einen *Ereignisempfänger* (*Listener*), der bei der Ereignisquelle registriert sein muss.

Es gibt einige fertig implementierte Datenstrukturen in Java. Die Klasse `Vector` ist eine Erweiterung eines *Arrays*. Sie enthält bereits alle wünschenswerten Zugriffs- und Aktualisierungsmethoden, wie z.B. `contains()`, `insertElementAt()` oder `lastElement()`. Für *Listen* gibt es die `LinkedList`. Diese Klasse ist eine Implementation einer doppelt verketteten Liste mit einer Vielzahl an Methoden. Eine fertige Klasse `{Queue` gibt es nicht, hier bietet sich die Nutzung eines `Vector` oder einer `LinkedList`, diese enthalten unter anderem auch die typischen Queue-Operationen. Als Implementierung eines `Stacks` gibt es die Klasse `Stack`. Sie enthält lediglich die vier üblichen Stack-Methoden `empty()`, `peek()` (entspricht dem sonst üblichen top), `pop()` und `push()` und zusätzlich die Suchfunktion `search()`.

Bei *Such- und Sortierproblemen* eignet sich die Klasse `TreeSet`. Dabei handelt es sich um eine fertige Implementation eines Suchbaums. Der Entwickler kann bequem Methoden wie `add()`, `contains()` oder `remove()` nutzen. Durch Methoden wie `tailSet()` oder `subSet()` erhält man Teilbäume. Durch interne Darstellung der Datenstruktur als Baum ist der Zeitaufwand für `add()`,

`contains()` und `remove()` garantiert *O(log n)*. Die Klasse implementiert das Interface `Set`. Deshalb darf ein Element nicht doppelt vorkommen.

Zusätzlich gibt es eine abstrakte Klasse `{Dictionary`. Sie funktioniert wie ein Wörterbuch. Man erhält zu einem Schlüsselwert einen Eintrag, falls vorhanden. Einmal implementiert kann man einfach mit Methoden wie `get()`, `put()` oder `remove()` arbeiten.

*Datagramme* (UDP-Pakete) lassen sich durch die Klassen `DatagramSocket` und `DatagramPacket` aus `java.net` senden und empfangen. Zunächst wird eine Instanz von `DatagramSocket` erzeugt. Der Konstruktor bekommt eine Portnummer übergeben. Ein Datagramm wird erzeugt mit einem Byte-Array und der Länge dieses Arrays als Parameter. Soll das Datagramm versendet werden, braucht es zusätzlich die Parameter Adresse und Portnummer. Ein Datagramm kann dann mit den Methoden `send()` und `receive()` des *DatagramSocket* versendet und empfangen werden. Zuletzt muss der *DatagramSocket* noch mit `close()` geschlossen werden.

Bei *Sockets* (TCP-Verbindungen) muss zwischen Client und Server unterschieden werden. Benötigt werden die Klassen `Socket` und `ServerSocket` aus `java.net`, sowie Writer- und Reader-Klassen aus `java.io` für die Output- und InputStreams. Ein Client erzeugt eine Instanz vom Typ `Socket` mit einer Adresse und Portnummer. Dadurch wird eine Verbindung aufgebaut. Die Ein- und Ausgabeströme werden an gewünschte Reader- und Writerklassen (z.B. `BufferedReader` und `PrintWriter`) gebunden. Dies erfolgt durch die Methode `getInputStream()` und die analoge Methode `getOutputStream()`. In diese Ströme können dann passende Objekte geschrieben werden oder daraus gelesen werden. Die Verbindung wird beendet durch einen Aufruf der Methode `close()`.

Für eine Serveranwendung wird zuerst eine Instanz von `ServerSocket` erzeugt. Der Konstruktor bekommt als Parameter nur eine Portnummer, an der er auf eine Verbindung warten soll. Die Methode `accept()` wartet dann solange, bis eine Verbindung vom einem entfernten Rechner aufgebaut wird und übergibt eine Instanz eines `Socket`. Dieser wird dann genauso benutzt wie bei der Clientanwendung.

Eine weitere Möglichkeit für verteiltes Rechnen ist *Java IDL (Java Interface Definition Language)*. Es handelt sich dabei um eine Technologie für verteilte Objekte. Unabhängig von Plattform und Programmiersprache. Es ähnelt *RMI (Remote Method Invocation)*, ist aber nicht auf Java-Programme beschränkt. IDL ist wesentlicher Bestandteil von *CORBA (Common Object Request Brokerage Architecture)*. *Java IDL* ist die Abbildung (Mapping) der sprachneutralen *IDL* auf Java. *CORBA* und die IDL-Mappings stammen von einem Industrie-Konsortium „Object Management Group" (OMG) zu dem auch Sun (als Gründungsmitglied) gehört. Eine Java-Anwendung kann durch *Java IDL* Methoden auf Objekte von entfernten Rechnern zugreifen. Ein *Object Request Broker* vermittelt Links auf die entsprechenden Objekte. Diese Technologie ist wegen des großen Overhead für das Projekt Speedtrap allerdings ungeeignet.

Das Java Development Kit bietet Sicherheitsmerkmale für Anwender und Entwickler. Sicherheitsmerkmale für den Anwender sind: Schutz vor schädlichen Programmen, Bewahrung des Datenschutz der eigenen Dateien und gespeicherten Informationen. So hat seit JDK 1.2 der Anwender die Möglichkeit Policies zu Definieren, nach denen Applikationen eingeschränkten Zugriff auf Systemresourcen erhalten. In Version 1.0 gab es nur die Möglichkeit vollen oder gar keinen Zugriff zu gewähren. Für Entwickler gibt es API-Methoden zur Integration von Sicherheitsfunktionen, wie Kryptographie oder Zugriffskontrolle auf spezielle Ressourcen. Diese Funktionen werden realisiert durch die *Java Cryptography Architecture (JCA)*. Dabei handelt es sich um ein Framework zur Nutzung und Entwicklung von kryptographischen Funktionen. Eine *provider architecture* ermöglicht mehrere und dialogfähige Implementationen. Ein *cryptographic service provider (CSP)* ist dabei ein Package, das eine konkrete Implementation einer oder mehrerer Funktionen bietet. (z.b. Algorithmen für digitale Signaturen oder Key-Generation) Ein Entwickler kann gleichzeitig mehrere *CSP*s nutzen. Suns Referenzimplementation mit dem Namen *SUN* bietet bereits eine Vielzahl von Funktionen.

Für Administratoren, Entwickler und Anwender gleichermaßen nützlich sind drei wichtige Tools: Ein Keytool für die Erstellung von Paaren aus öffenltichen und privaten Schlüsseln, Import und Anzeige von *certificate chains* und Export und Verifizierung von Zertifikaten. Ein Jarsigner tool für das Signieren von *JAR (Java ARchive format)* files und das Verifizieren von signierten JAR files. Und ein Policy Tool für das Erstellen und Bearbeiten von security policy configuration files.

Bei der Java Programmierung – wie bei der Programmierung im Allgemeinen – ist es sinnvoll verschiedene Punkte im Hinblick auf die *Performance* eines Programms zu berücksichtigen. So sind bei Java String-Manipulationen nach Möglichkeit zu vermeiden, da sie recht teuer (im Bezug auf die Laufzeit) sind. Bei Vergleichen sollten die logischen Operatoren `&&` und `||` an Stelle von `&` und `|` benutzt werden, da bei diesen der rechte Operand nur ausgewertet wird, wenn der linke noch nicht allein zu einem Ergebnis geführt hat (*short circuiting*). Bei aufeinanderfolgenden if-Anweisungen sollte geprüft werden, ob ein Fall überhaupt eintreten kann, nachdem bereits ein anderer eingetreten ist. Ist dies nicht der Fall, sollte das `else if` -Konstrukt verwendet werden und die Fälle nach Häufigkeit sortiert werden. In Bezug auf Schleifen sollte darauf geachtet werden, dass Invarianzen nicht innerhalb einer Schleife und somit unnötig oft ausgeführt werden (*Loop Invariant Code Motion* zum Teil auch durch die JVM[3]). Dabei ist zu beachten, dass auch das Abbruchkriterium Teil einer Schleife ist. Ferner sollten wiederholte Array-Zugriffe in Schleifen vermieden werden. Das Auftreten von Ausnahmen sollte nach Möglichkeit vermieden werden, was oft durch sinnvolle Schnittstellen erreicht werden kann (Feststellen der Ausnahme, bevor sie eintritt).

Weiterführende Literatur zu diesen Themen: [Abts00], [Flan98], [Kred02], [Schr02].

[3] JVM: Java Virtual Machine

# Kapitel 3

# Implementierung

In der Implementierungsphase soll das Ergebnis der bisherigen Entwurfsphasen in die Realität umgesetzt werden. Dazu implementieren die drei Gruppen *Monitor, Netzplan* und *Agent* ihre Entwürfe in *Java*. Es ist zu erwarten, dass an einigen Stellen vom bisherigen Entwurf abgewichen wird. Die Gruppen haben in diesem Kapitel die Gelegenheit, kurz ihre Implementierung darzustellen und auf Besonderheiten einzugehen.

## 3.1 Monitor-Agent Schnittstelle

Während der Implementierungsphase war es notwendig, die Schnittstelle zwischen dem Monitor und den Agenten genau zu spezifizieren. Zunächst einigte man sich auf ein einheitliches Kommunikationsformat, wobei hauptsächlich das Nachrichtenformat und einige Details wie Trenn- und Endzeichen festgelegt wurden. Anschließend wurde ein grober Entwurf der Schnittstelle definiert, der sich allerdings schnell als unzureichend erwies. Während der Implementierung wurde daher dieser Entwurf praktisch vollständig überarbeitet. Das Ergebnis dieser Entwicklung ist im Administrationshandbuch in Kapitel 6.2.4 zu finden.

## 3.2 Agent

Die Implementierung der Agenten wurde auf Basis der im Zwischenbericht angegebenen Klassendiagramme durchgeführt. Die Arbeit wurde zu Beginn der Implementierungsphase in drei Aufgabenbereiche eingeteilt, die von unterschiedlichen PG-Mitgliedern bearbeitet wurden:

1. Frontend, Kommunikationsschnittstelle zum Monitor, Lastgenerator
2. Switch-Backend, Datenstruktur zum Speichern der Messdaten
3. NIC-Backend

Für Punkt 1 wurde Daniel eingesetzt, der zunächst das Frontend implementiert hat. Das Frontend kontrolliert alle anderen Komponenten des Agenten und regelt den Kommunikationsverkehr zum Monitor; daher war die Implementierung des Frontends am Anfang von entscheidender Wichtigkeit. So konnte der Agent schon recht früh auf seine Funktionalität getestet werden ("Sandbox-Tests"). Danach wurde intensiv an der Kommunikationsschnittstelle gearbeitet. Da der Agent nur auf Kommandos vom Monitor reagiert und keine eigenmächtigen Aktionen durchführt, wird der

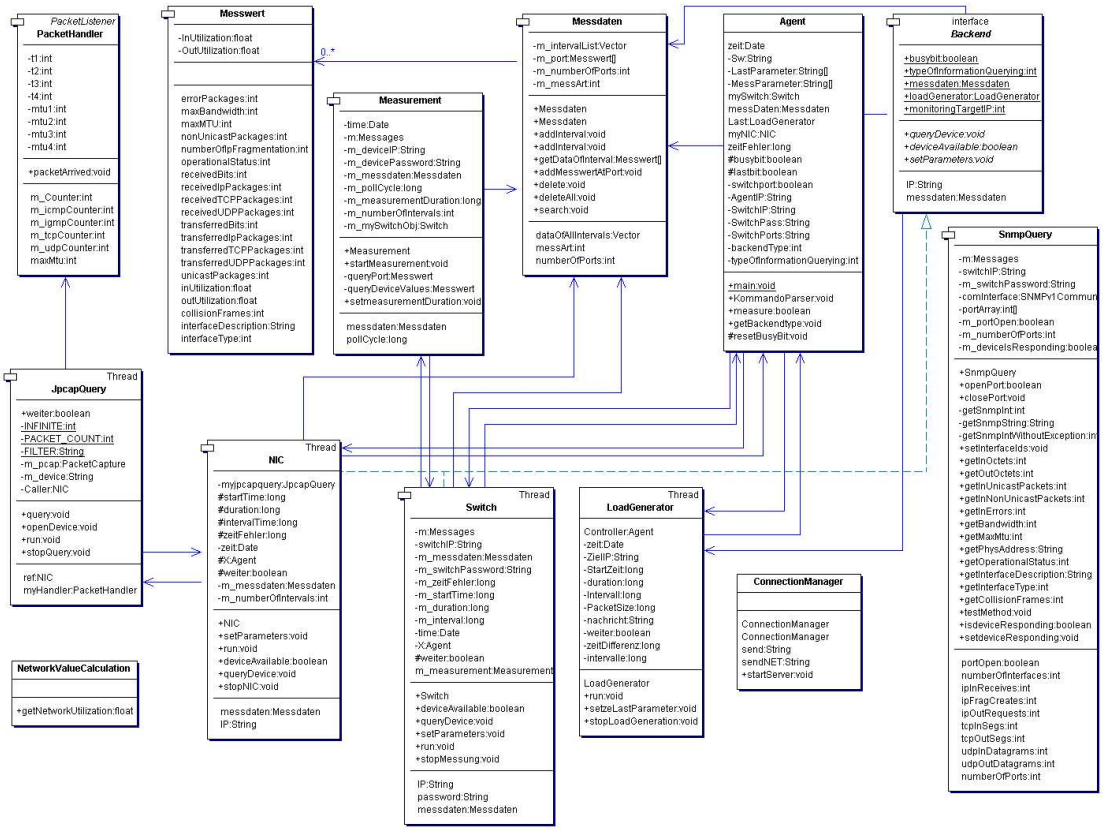



Abbildung 3.1: Das Package Agent

Agent letztlich über die Kommunikationsschnittstelle gesteuert. Dadurch war der Implementierungsaufwand hier sehr hoch und hat einige Zeit in Anspruch genommen. Erschwerend kam hinzu, dass bei diesem Punkt permanent eine Abstimmung mit der Monitorgruppe notwendig war (,was wir aber auch erwartet haben), wodurch es teilweise zu Blockaden bzw. Wartezeiten kam. So konnte in den "Wartepausen" parallel der Lastgenerator implementiert werden. Als diese Aufgaben erledigt waren, wurden an allen Komponenten noch Tests, Verbesserungen und Veränderungen durchgeführt, bis auf dieser Arbeitsebene keine Fehler mehr aufgetreten sind.

Für Aufgabenbereich 2 war Babak zuständig, der zunächst das Switch-Backend implementiert hat. Hierfür waren einige vorbereitende Schritte notwendig: Zunächst mussten die allgemeinen Dokumentationen für Switches und deren MIB-Datenbanken analysiert werden, um festzustellen, welche Daten überhaupt per SNMP verfügbar sind. Dann wurden die Daten identifiziert, die bei allen Switches zur Verfügung stehen, da es wenig wünschenswert ist, das Switch-Backend nur für ein Modell von Switches zu optimieren. Dann wurden die Abfragemethoden implementiert, die für das "Sammeln" der MIB-Managementdaten notwendig sind. Danach wurde die Datenstruktur zum Speichern der Messdaten angelegt, die als lineare Liste implementiert wurde. Jedes Listenelement stellt selbst wieder ein Feld vom Typ Messwert dar, welches die gesammelten Managementdaten für jeden einzelnen Switchport enthält. Da für diese Datenstruktur eine wichtige Anforderung war, dass die Daten auch schon während der Messung zur Verfügung stehen, bot sich diese Art der Speicherung an. Nachdem die erforderlichen Anpassungen für die Kommunikation mit dem Frontend fertiggestellt waren, wurden abschließend für Aufgabenbereich 2 einige Tests an dem uns von der IRB zur Verfügung gestellten Cisco Switch durchgeführt.

Für den dritten Aufgabenbereich waren Firas und Ulrich eingeteilt. Dieser Aufgabenbereich erforderte einige Vorbereitungszeit und es mussten knifflige Probleme gelöst werden, so z.B.: Wie kann man mit Java auf die Netzwerkkarte zugreifen? Alle diese Probleme konnten letztlich gelöst werden unter Zuhilfenahme von externen (und nativen) Methoden, die in c++ implementiert waren. Somit bestand die wesentliche Aufgabe am Anfang der Implementierungsphase darin, die gegebenen zusätzlichen Klassen zu verstehen und den Anforderungen unseres Agenten anzupassen. Außerdem mussten die Schnittstellen zum Frontend und zur Datenstruktur zum Speichern der Messdaten implementiert werden. Diese Anpassungs- und Änderungsaufgaben konnten nach der o.g. Vorbereitungszeit relativ zügig durchgeführt werden. Danach wurden einige Tests erfolgreich durchgeführt.

Nachdem nun der Agent vollständig implementiert war, wurden in Zusammenarbeit mit den anderen beiden Gruppen noch einige Anpassungen und Fehlerkorrekturen durchgeführt, bis das Produkt reif war für die realen Testeinsätze.

Das endgültige Klassendiagramm des Agenten-Package ist in Abbildung 3.1 zu sehen.

## 3.3 Monitor

Die Implementierungsgruppe des Monitors hat versucht sich grob an das Entwurfsmodell des Zwischenberichts ([P422ZB]) zu halten. Die grobe Klassenstruktur und die Namen der Klassen wurden weitgehend beibehalten, jedoch wurde sehr schnell klar, dass starke Erweiterungen und Änderungen des ursprünglichen Konzepts notwendig würden. Die genauen Änderungen des Entwurfs sind im *JavaDoc* des Monitors nachzulesen. Hier soll nur das grundsätzliche Vorgehen bei der Implementierung erläutert werden.

Zunächst wurden Untergruppen gebildet und die Aufgaben weiter aufgeteilt. Eine Person kümmert sich um die Kommunikation des Monitors mit den Agenten, eine um die Darstellung des Netzplans und der zugehörigen Daten sowie die GUI Struktur und eine Person um den Entwurf der Eingabe Dialoge. Für eine vierte Teilaufgabe, die Auswertung der Messdaten, war zunächst kein Personal verfügbar, später wurde die Gruppe jedoch durch eine zusätzliche Person verstärkt, so dass auch dieser Bereich abgedeckt werden konnte.

Beim Entwurf der einzelnen Komponenten war es notwendig Dummy-Klassen, die noch nicht implementierte Funktionalitäten nachbilden, zu entwerfen. Diese wurden für die Tests der Klassen verwendet und auch so angepasst, dass Sonderfälle überprüft werden konnten. Bei diesen Tests wurden auch immer wieder neue wünschenswerte Funktionen festgestellt, die dann auch meist implementiert wurden.

Nachdem die Monitorfunktionalitäten weitgehend implementiert und getestet waren, kam es nur im Bereich der Kommunikation mit den Agenten häufig zu Schwierigkeiten, die eine Überarbeitung der Monitor/Agentschnittstelle notwendig machten, was den Testbetrieb am LS-4 und Echteinsatz bei der Polizei verzögerte. Nachdem auch dieses Problem gelöst war, lief der Monitor stabil und es wurden lediglich kleinere „Goodies" zugefügt, die bei der Implementierung aufgeschoben worden waren.

Die Klassen des *Monitor*-Package (Abbildung 3.2) stellen Funktionen bereit, die von den GUI-Klassen genutzt werden. Die Klasse *Utils* enthält dabei Funktionen, die in keiner direkten Beziehung zu anderen Klassen stehen. Die Klasse *LogFile* stellt die Funktionalitäten des Log-Fensters bereit. *CustomFileFilter* wird von den Laden-/ Speichern Dialogen verwendet und *Messwerte* unterstützt die Handhabung der Messwerte der Anwendung. *Timeserver* stellt eine Timeserverfunktion bereit, die die Agenten mit dem Monitor zeitlich synchronisiert. Schließlich enthält das Package auch die weiteren Unterpackages *GUI, Communication* und *Messung.*

Die Klassen des *GUI*-Package (Abbildung 3.3) stellen die grafischen Objekte der Anwendung bereit. Wie im Zwischenbericht geplant, ist die Hauptfensterklasse *GUIRahmen.* Hier sind die

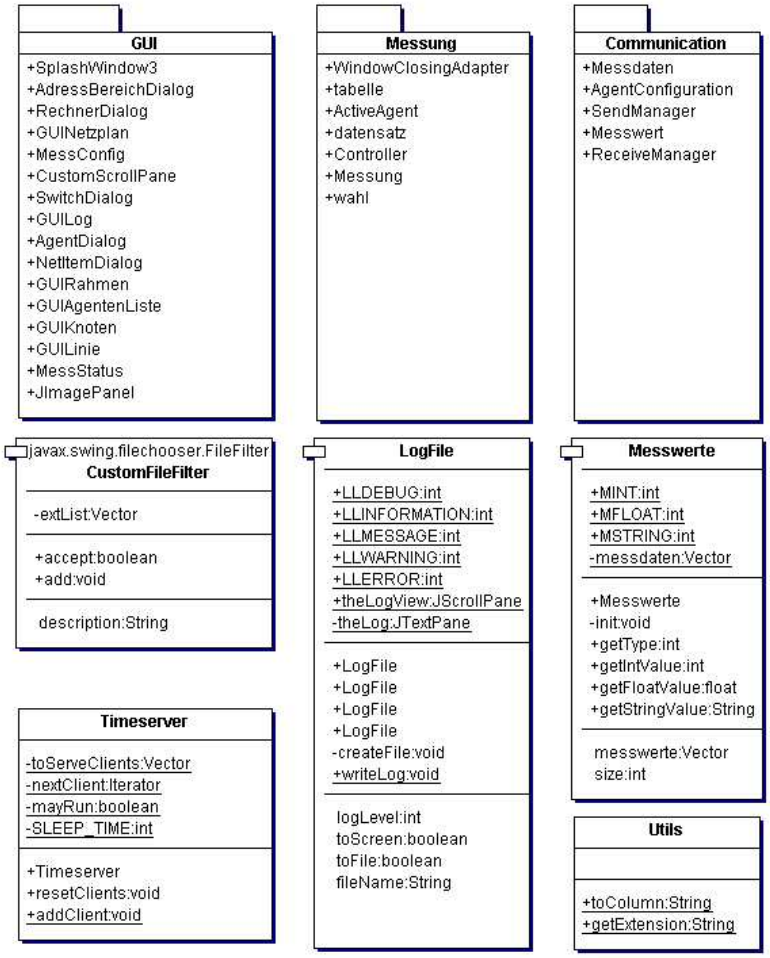



Abbildung 3.2: Das Package Monitor

Menüleisten, sowie einige Funktionen, wie das Laden und Speichern der Netzpläne realisiert. Die Hauptklasse für die Darstellung der Netzpläne ist wie *GUINetzplan*. Diese Klasse stellt den Netzplan grafisch dar, bietet Möglichkeiten ihn zu bearbeiten und stellt auch während einer Messung ein Messdatum in Echtzeit dar. Außerdem werden noch einige zusätzliche Funktionen, wie *zoomem, Rechner suchen, Agenten vorschlagen, etc.* bereitgestellt. Die *-Dialog* Klassen werden von *GUINetzplan* genutzt, um die Knotenobjekte des Netzplans zu konfigurieren. Die weiteren *GUI* Klassen stellen weitere Funktionalitäten bereit, die von den bereits erwähnten Klassen *GUIRahmen* und *GUINetzplan* genutzt werden. So stellt die Klasse *GUILog* den Logbereich des Hauptfensters und dessen Funktionen bereit. *AdressBereichDialog* stellt einen Eingabedialog für den automatischen Netzplanscan zur Verfügung. Die Klassen *GUIKnoten* und *GUILinie* sind jeweils die GUI-Darstellungen der Knoten- und Linienobjekte des Netzplans mit ihren Funktionen. *MessConfig* und *MessStatus* stellen Dialoge bereit, die zur Konfiguration der Messung dienen, bzw. den aktuellen Status während einer Messung wiedergeben. Die Klasse *CustomScrollPane* ist eine Hilfsklasse, die die Scrollfunktionen der Netzplanansicht zufügt.

Die Kommunikation zwischen dem Monitor und den Agenten erfolgt über die Klassen die im Package Communication (Abbildung 3.4) enthalten sind. In diesem Package sind auch die Datenstrukturen enthalten, die der Speicherung der Messdaten dienen. Im folgenden werden die einzelnen

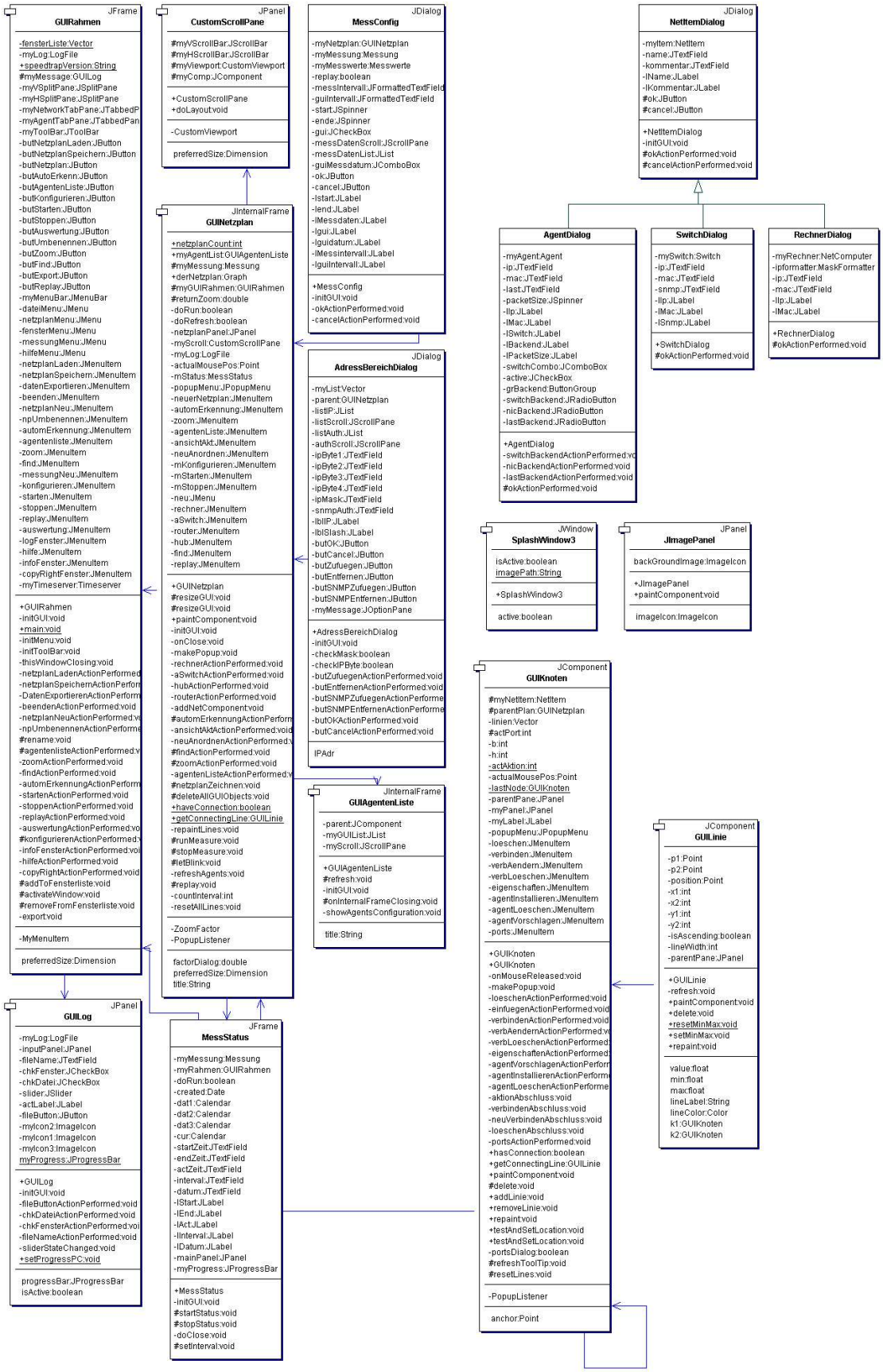



Abbildung 3.3: Das Package GUI

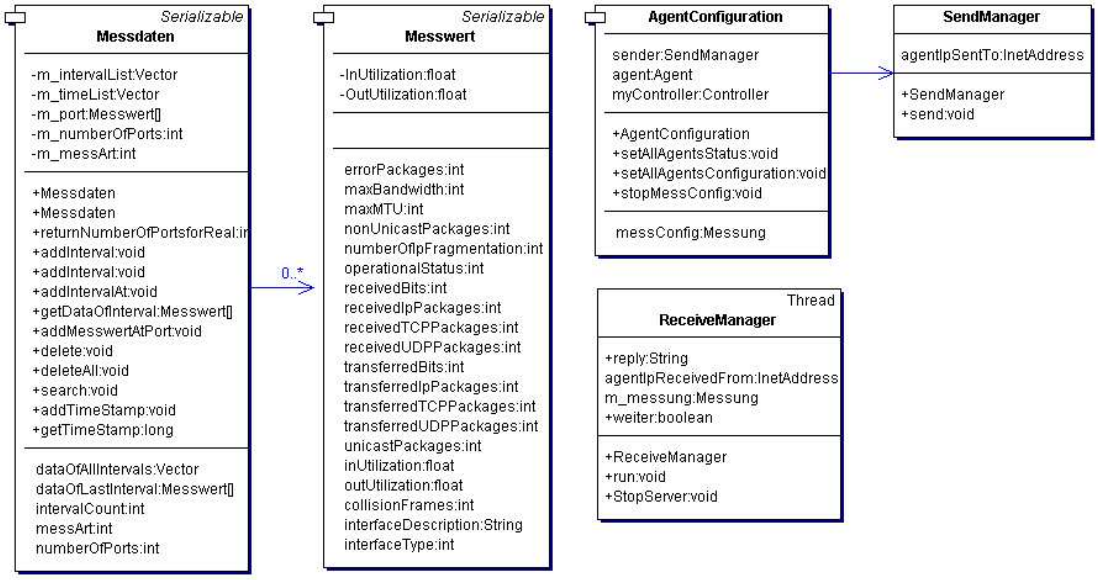



Abbildung 3.4: Das Package Communication

Klassen aufgelistet und ihre Aufgaben kurz erläutert.

- SendManager
  Die Klasse *SendManager* ist der Ausgangskanal des Monitors. Kein Paket verlässt den Monitor ohne diese Klasse zu passieren.
- ReceiveManager
  Den *ReceiveManager* kann man als Gegenstück zum SendManager betrachten. Alle Pakete die zum Monitor hereinkommen, werden vom ReceiveManager empfangen.
- AgentConfiguration
  Verschiedene Pakete haben unterschiedliche Inhalte. In der Klasse *AgentKonfiguration* wird der Inhalt der unterschiedlichen Pakete zusammengestellt, bevor die Pakete über der *SendManager* gesendet werden.

Das Package Messung (Abbildung 3.5) enthält die Klassen die der Monitor braucht um eine Messung zu verwalten. In diesem Package stellt sich die Klasse *Messung* als die zentrale Klasse dar. Alle Pakete die vom ReceiveManager empfangen werden, werden hier ausgewertet und entsprechende Reaktion darauf werden hier ausgelöst. Auch die Messdaten werden in der Klasse Messung ausgelesen und gespeichert. Die Klasse *ActiveAgent* enthält bei jeder Messung eine Liste von Agenten die gerade an der Messung teilnehmen. Diese Klasse ist sehr wichtig für die Klasse *Controller*, die diese Liste durchläuft und mit den Agenten, die darin enthalten sind während die Messung kommuniziert, um etwa Messdaten abzuholen.

Die Klassen für die Auswertung der Daten befinden sich ebenfalls im *Messung*-Package. Die Klasse *Wahl* stellt dem Benutzer hierbei einen Konfigurationsdialog zur Verfügung. In diesem Dialog kann bestimmt werden, welche der gemessenen Daten angezeigt werden sollen. Man kann weiter noch den Zeitraum bestimmen, dessen Daten präsentiert werden. Die Klasse *Tabelle* wertet die Daten des Benutzers aus und erstellt die eigentliche Ansicht. Diese besteht aus einer zweidimensionalen Tabelle, die alle Netzwerkgeräte enthält und die Netzlast im Segment der jeweiligen Komponente auflistet.

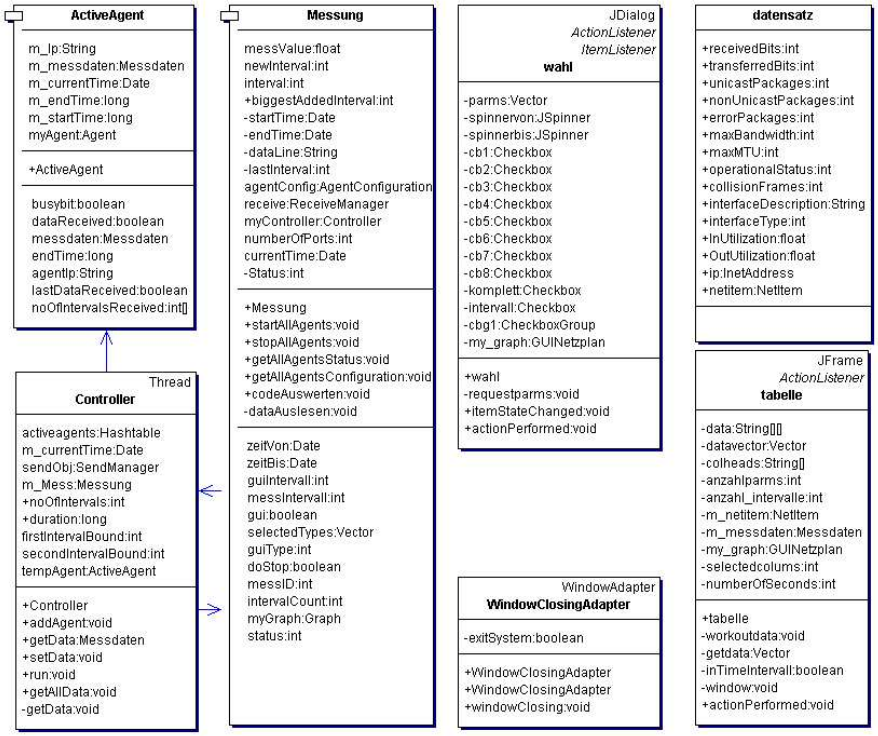



Abbildung 3.5: Das Package Messung

## 3.4 Netzplan

Im folgenden Abschnitt wird ein kurzer Überblick über die grobe Funktionsweise und den endgültigen Aufbau des Netzplanmoduls gegeben. Die Implementierung richtet sich dabei nach dem Entwurfsmodell, welches im Zwischenbericht festgehalten wurde [P422ZB].

Während die Struktur der Planung praktisch 1:1 implementiert wurde, wurden zahlreichen Erweiterungen, deren Notwendigkeit erst während der Implementierungsphase klar wurde, hinzugefügt. Daneben wurden den Klassen englische Namen gegeben, da sich die PG 422 nach Fertigstellung des Zwischenberichts entschied, auch die Dokumentation in Englisch zu erstellen. Da die angesprochene Dokumentation im JavaDoc Format auf alle Details der Implementierung eingeht, soll an dieser Stelle lediglich eine grobe Übersicht über Funktionalität des Netzplans und die Funktionsweise des Netzplanscans gegeben werden.

Im Package Netzplan (Abbildung 3.6) wurden die Klassen *Main* und *Graph* zu Graph zusammengelegt. Die Klasse Graph ist die repräsentation des Netzwerkes mit allen Elementen, die in verscheidenen Listen gehalten werden und enthält zusätzlich alle Messdaten. Die Klasse *Netdetection* ist die umbenannte Klasse *Abfragemethoden* und ist nun vollständig für die Netzwerkerkennung verantwortlich. Jeder Graph enthält ein Objekt dieser Klasse. Die neue Klasse *Macserver* ist ein „Helferthread" der im Kontext des „Agenten" läuft und dort dokumentiert ist.

Im Unterpaket Datentypen (Abbildung 3.7) sind ausser den neuen englischen Namen für alle Klassen die beiden Klassen *Layer3Subnet* und *RadialFreeTreeCalculationValues* neu. Erstere wird lediglich zur Darstellung eines IP Subnetzes benötigt, so wie es von den Switches per „Netdetection" ermittelt wird und enthält auch Methoden zu Maskenanwendung auf IP Subnetze. Zweitere ist eine Klasse, deren Instanzen mit internen Berechnungswerten während der radialen Darstel-

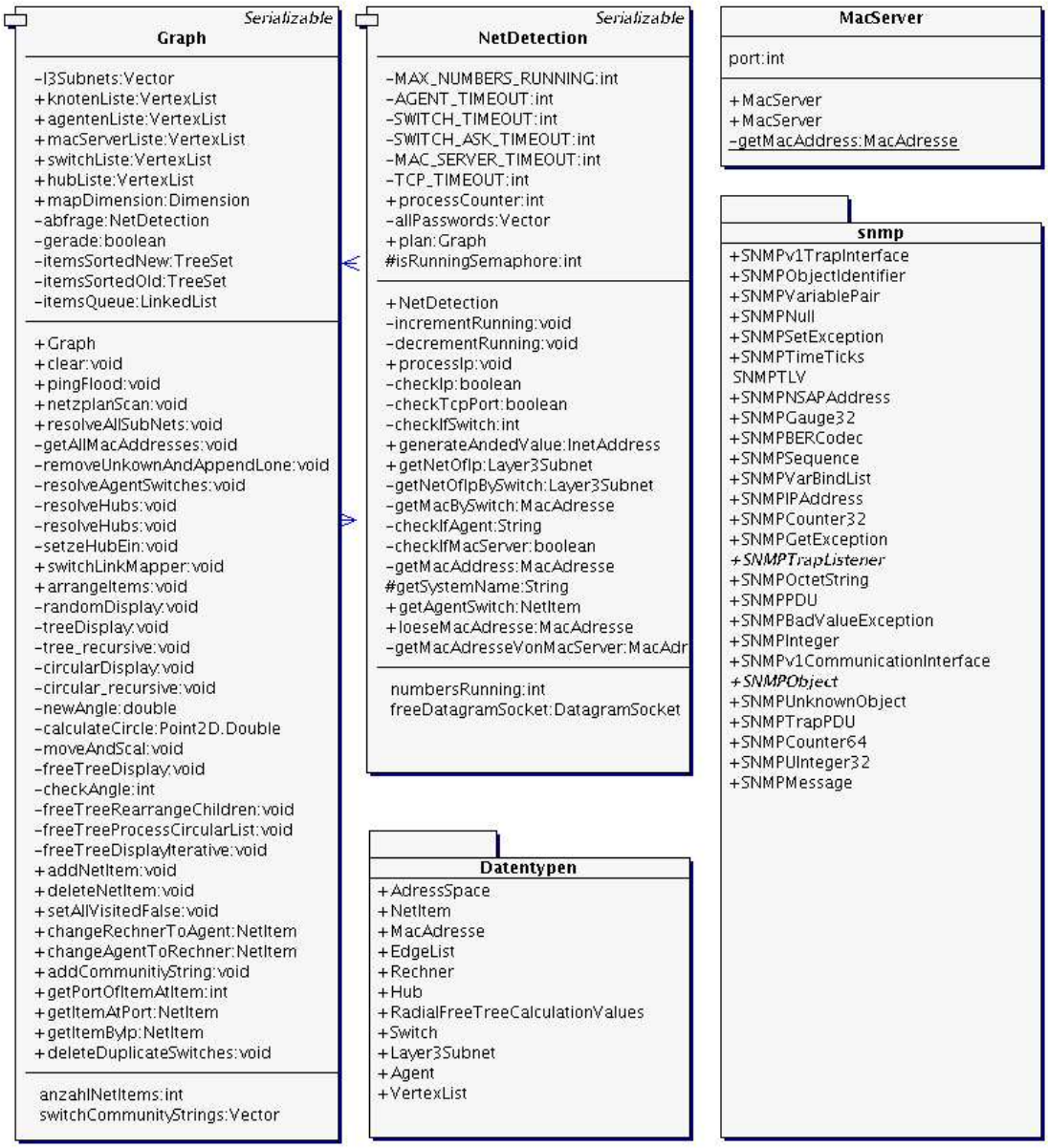



Abbildung 3.6: Das endgültige Package Netzplan

lungsberechnung des Graphen gefüllt werden. Während der Berechnung erhält jedes Netitem ein Objekt dieser Klasse, das anschliessend sofort wieder gelöscht wird.

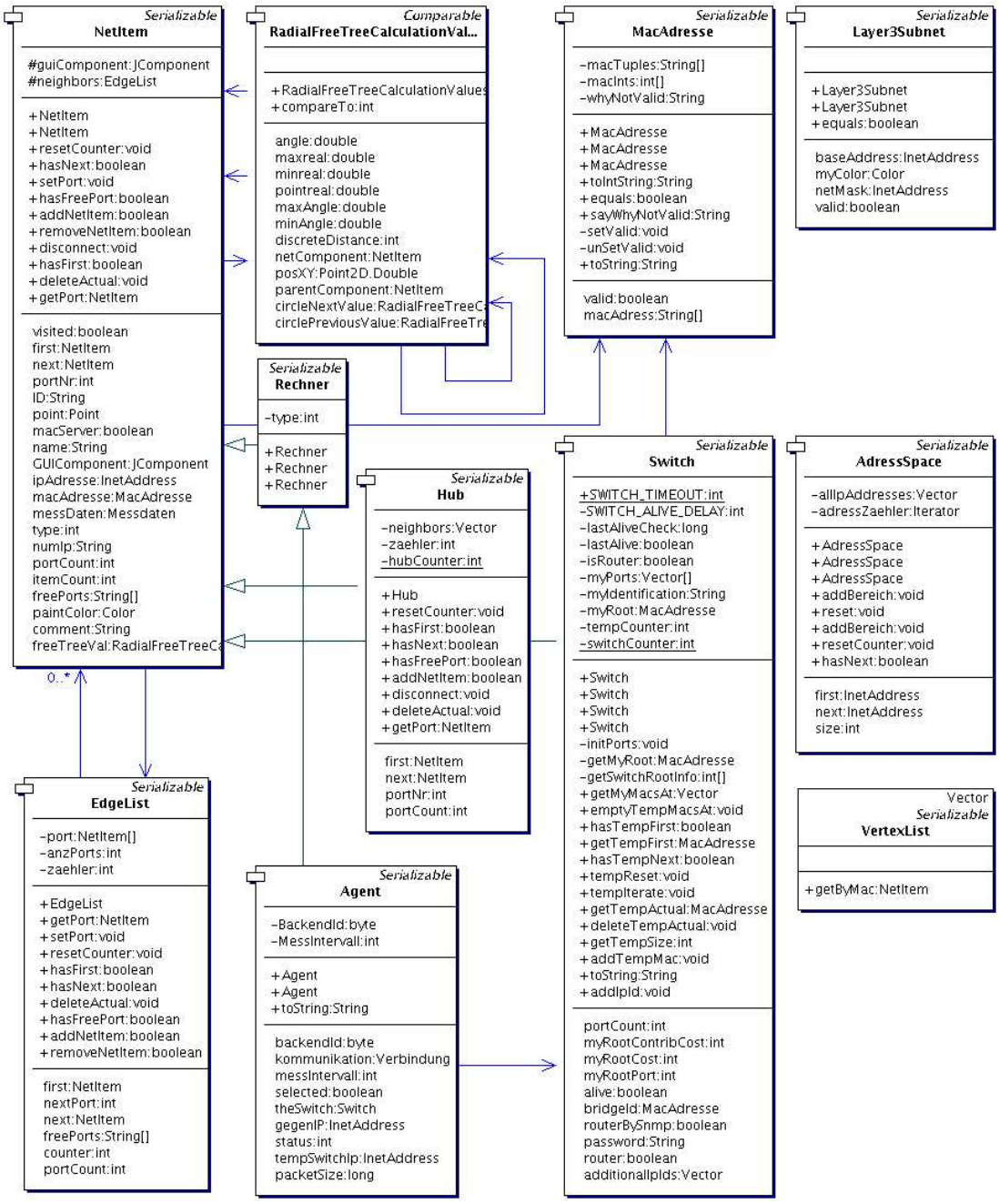



Abbildung 3.7: Package Netzplan.Datantypen

# Kapitel 4

# Testeinsatz

Speedtrap kommt nach seiner Fertigstellung zuerst am Lehrstuhl 4 zum Einsatz. Hier können unter Umständen noch letzte Bugs im Code beseitigt werden, bevor die eigentlichen Messungen starten. Die Messungen am LS4 sind zudem die Generalprobe für den Einsatz im Netz der Kreispolizeibehörde Unna. Ausserdem soll mit Speedtrap am LS4 erforscht werden, warum der Performancegewinn in Abhängigkeit von der Rechneranzahl bei der verteilten numerischen Analyse von Markov Ketten vergleichsweise bescheiden ist. Die Steigerung beim Einsatz von 10 Sun Workstations gegenüber einer einzigen Maschine liegt nach bisherigen Erfahrungen gerade einmal bei 70%. (Abbildung 4.1)

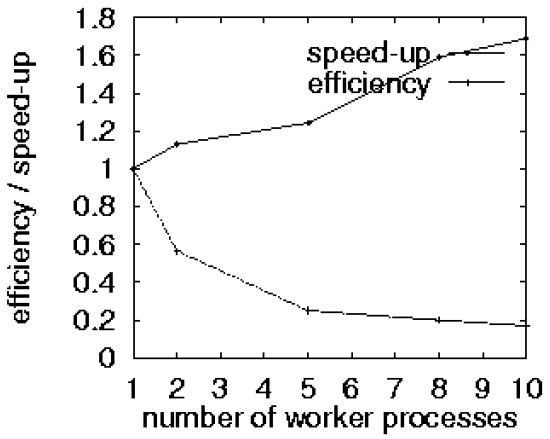



Abbildung 4.1: Performancediagramm

## 4.1 Einsatzplan LS4

### Vorgehensmodell

Im folgendem wird ein Vorgehensmodell für den Einsatz des Java-basierten Netzwerk-Analysetools beschrieben:

**Vorbereitung:**

Zwei Vorraussetzungen müssen durch die Informatik Rechner Betriebsgruppe (IRB), die die Pflege und Wartung aller Suns im Bereich der Informatik an der Universität Dortmund durchführt, geschaffen werden, damit Speedtrap zum Einsatz kommen kann:

**Zentraler Switch Cisco Catalyst 5500:**

Ein Mitarbeiter der IRB konfiguriert den zentralen Switch Cisco Catalyst 5500 (Abbildung 4.2), so dass ein Speedtrap Agent SNMP Abfragen tätigen kann. Der Agent wird auf dem Rechner "werner.cs.uni-dortmund.de"(129.217.16.83) installiert. Nur diese IP Adresse benötigt Zugriff auf die SNMP Ports 160/161 des Cisco Switches. Das Passwort (Community String) erhält der Agent vom Monitor bei der Initialisierung.

**Sun SNMP Agent**

Ein Mitarbeiter der IRB installiert auf den meisten Sun Workstations des Lehrstuhls 4 den original Sun SNMP Agenten oder eine andere vertrauenswürdige SNMP Agentensoftware (z.b. net-snmp von http://net-snmp.sourceforge.net). Damit sollten Sicherheitsbedenken bzgl. Speedtrap ausgeräumt werden, da uns im Vorfeld schon mitgeteilt wurde, dass wir weder Rootzugang noch direkten Zugang zu den Netzwerkkarten (NIC) der Sun Workstations bekommen können. Einen Softwareagenten klassischer SNMP Definition können wir von unserem Agenten mit dem Switch-Backend abfragen. Dadurch wird unser eigenes NIC-Backend allerdings obsolet. Die Auswahl wurde so getroffen, dass an jedem der "kleinen" Switche mindestens an einem Rechner ein Agent installiert ist, sofern nicht nur Server oder Sun Ultra 1 an diesem Switch hängen.

Der Netzplan des LS4 wurde gegenüber der Version vom Wintersemester überarbeitet und einige Korrekturen vorgenommen, zumal einige Rechner nun mit 100Mbit/s anstelle von 10Mbit/s an das Backbonenetz angeschlossen sind. (Siehe Abbildung 4.2).

Auf den folgenden Rechnern brauchen wir SNMP Agenten von der IRB:

franz, reinhard, brigitte, michael
detlef, gerd, klaus, arnulf,
bruno, astrid, roland, christa,
helmut, otti, olaf, christel,
martin und stefan

Anmerkung: Die Sun Maschinen am LS4 tragen als Namen die Vornamen von ehemaligen Mitarbeitern.

Auf dem Rechner "werner" wird der Agent für den Cisco 5500 installiert.
Auf dem Rechner "ingo" wird der Monitor installiert, der die Agenten ansteuert und die Messungen überwacht. (Siehe Abbildung 4.3)

Sobald alle vorbereitenden Arbeiten abgeschlossen wurden, kann mit ersten Tests in einer größeren Umgebung als die uns zur Verfügung stehenden Testumgebung begonnen werden. Dazu gehört zuerst einmal, dass auf den oben genannten Rechnern der Speedtrap Agent installiert wird. Dies kann remote per SSH vom gleichen Rechner geschehen, auf dem der Monitor installiert ist. Das File "Agent.jar" liegt im Homeverzeichnis des PG Accounts für den LS4. Da die Homeverzeichnisse auf einem zentralen Fileserver liegen, steht das File auf jedem Rechner zur Verfügung, auf dem man sich mit dem PG Account einloggt. Gestartet wird der Agent mit dem Kommando "java -jar Agent.jar". Danach wird die automatische Netzerkennung gestartet. Eventuelle Unterschiede zu dem Netzplan, der in Abbildung 4.2 aufgezeichnet ist, können manuell korrigiert werden. Diese Monitoroption wird auf jeden Fall benutzt werden müssen, da die vielen kleinen Switche und Hubs im Netz des LS4 nicht abgefragt werden können und der Monitor hier die Struktur des Teilbaumes

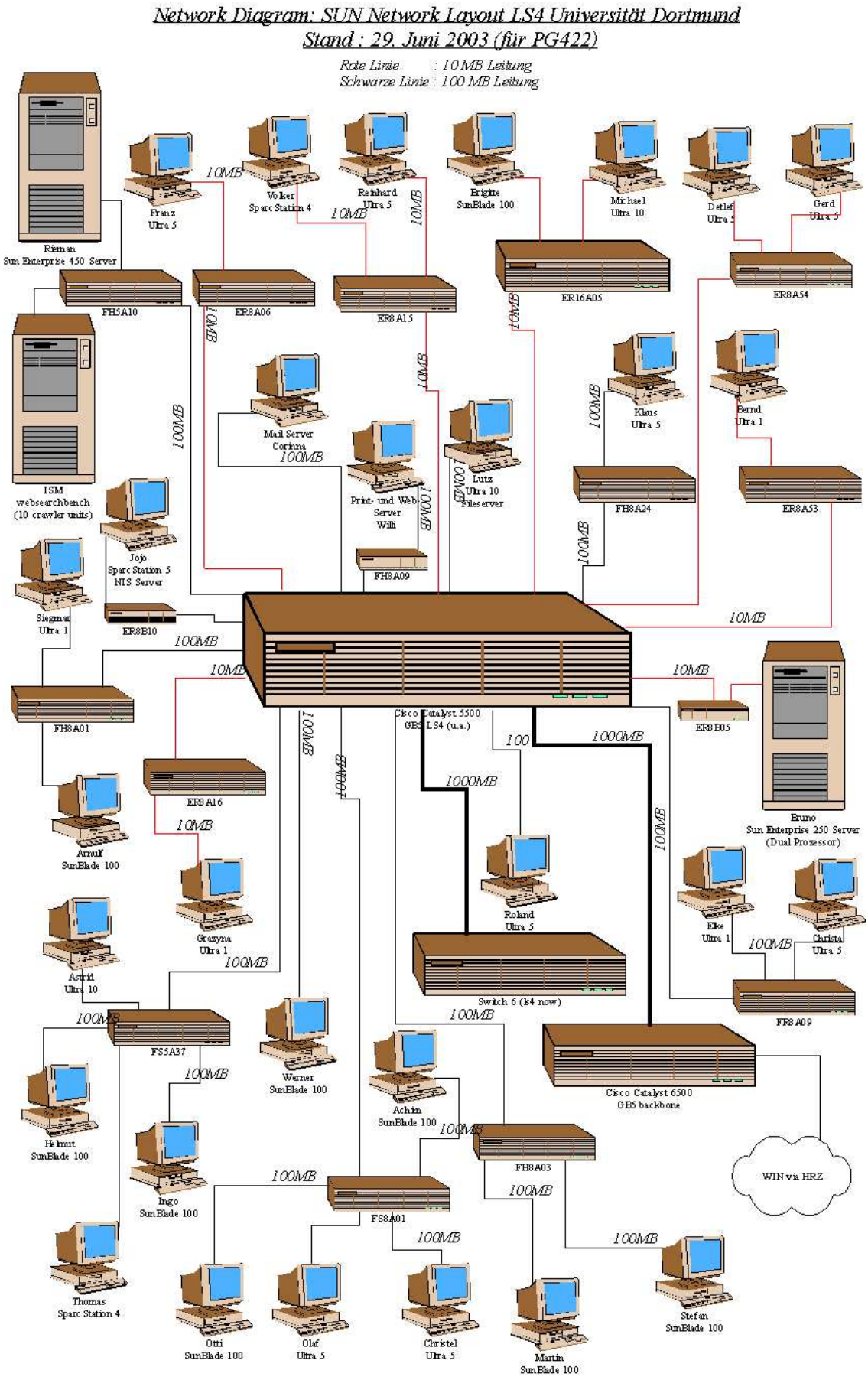



Abbildung 4.2: Das Sun-Netzwerk des LS4

nicht erkennen kann. Die Netzplanerkennung sollte alle Agenten erkennen, sowohl die eigenen, als auch die von der IRB installierten SNMP Agenten klassischer Definition. Darüberhinaus ist damit zu rechnen, dass sich auch noch weitere SNMP fähige Geräte melden werden, wie z.B. Drucker oder Windows Terminal Server. Diese sollen vom Monitor ignoriert werden können. Als letzte Vorbereitung muss der Monitor alle Speedtrap Agenten erfolgreich initialisieren.

**Anmerkung zum Netzplan (Abbildung 4.2 / 4.3):**

Für verteiltes Rechnen am Informatiklehrstuhl 4 werden ausschließlich die Sun Workstations benutzt. Daher bleiben x86- kompatible PCs, Apple Macintosh Computer und auch die Netzwerkdrucker in diesem Plan unberücksichtigt. Die Sun Server "jojo", "lutz" und "Willi" werden nur der Vollständigkeit halber erwähnt, auch sie werden nicht für verteiltes Rechnen benutzt. Ansonsten werden noch zwei Rechnersysteme erwähnt, die zwar ebenfalls nicht für das verteilte Rechnen benutzt werden, aber viel Netzlast erzeugen können: Der Rechner "rieman", eine Sun Enterprise 450, die als Fileserver dient und die Internet Suchmaschine "Websearchbench" (http://websearchbench.cs.uni-dortmund.de) mit ihren zehn Crawlereinheiten (Forschungsgebiet Prof. Lindemann). Der Testswitch Cisco Catalyst 2900, der für die Entwicklung von Speedtrap in Raum 438 GB5 installiert wurde, muss vor der Messung wieder aus dem Lehrstuhlnetz entfernt werden und die drei Sun Workstations wieder an ihre ursprünglichen Dosen angeschlossen werden.

Ein entscheidener Punkt ist natürlich die Anbindung jeder einzelnen Workstation. Eine Vielzahl ist noch mit 10Mbit/s angeschlossen. Dabei handelt es sich hauptsächlich um ältere Maschinen. Die Informatik Rechner Betriebsgruppe (IRB) erweitert ständig die Kapazitäten des Netzes und so werden auch die im Moment noch mit 10Mbit/s angeschlossenen Rechner in der Zukunft auf 100Mbit-Ports gelegt, wenn die kleineren Switche getauscht werden.

Die Bezeichnung der kleinen Switche weisst auf deren Typ hin:

ER = Ethernet Repeater (nur 10 Mbit Ports)
FH = Fast Ethernet Hub (2 Stränge, davon 1 Strang 10 Mbit Ports und 1 Strang 100 Mbit Ports)
FR = Fast Ethernet Repeater (100 Mbit Ports)
FS = Fast Ethernet Switch (100 Mbit Ports)

Alle Uplinkports der Gerätetypen FH, FR und FS sind direkt mit dem Catalyst 5500 verbunden.

Sind bis zu diesem Punkt keine Fehler aufgetreten, so kann mit der eigentlichen Messung der für die Leistungsbewertung nötigen Größen begonnen werden. Am ersten Tag soll dazu die normale Netzlast gemessen werden. Dazu sind Messungen während eines ganzen Arbeitstages im gesamten 16er Netz durchzuführen und entsprechend zu protokollieren. (z.B. in Logfiles). Screenshots sind nicht nötig, da der Monitor Logfiledaten auch nachträglich graphisch als Animation darstellen kann. Eventuell werden hier schon Engpässe oder besondere Vorkommnisse wie plötzlich stark ansteigende, temporäre Netzlasten erkannt. An einem weiteren Tag wird dann in Zusammenarbeit mit Dipl. Inform. Markus Fischer, der die verteilten Berechnungen großer Matrizen auf den Suns startet, erneut gemessen. Die Berechnungen werden mehrmals wiederholt, jeweils mit einer unterschiedlichen Zahl an beteiligter Maschinen. Es ist zu untersuchen, welchen Einfluß die Anzahl der miteinander kommunizierenden Maschinen in Relation zu der beschleunigten Berechnung der Matrizen hat. Auch hier werden die Logfiles aller Meßdurchgänge zur späteren (graphischen) Auswertung gespeichert.

Am Ende der auf 3 Tage angesetzten Messungen sollten die Sun SNMP Agenten deaktiviert werden und der Zugriff auf die SNMP Ports des Catalyst deaktiviert werden und der Community String wieder auf den alten Wert zurückgesetzt werden.

Die Messungen am Lehrstuhl 4 sind zugleich auch letzte Tests für den Einsatz von Speedtrap im Netz der Kreispolizeibehörde Unna. Während am LS4 Fehler im Sourcecode noch schnell im PG-Pool behoben werden können, ist dies in Unna auf Grund der Abschottung des Polizeinetzwerkes nicht so leicht möglich, weil kein Zugriff auf das CVS möglich ist.

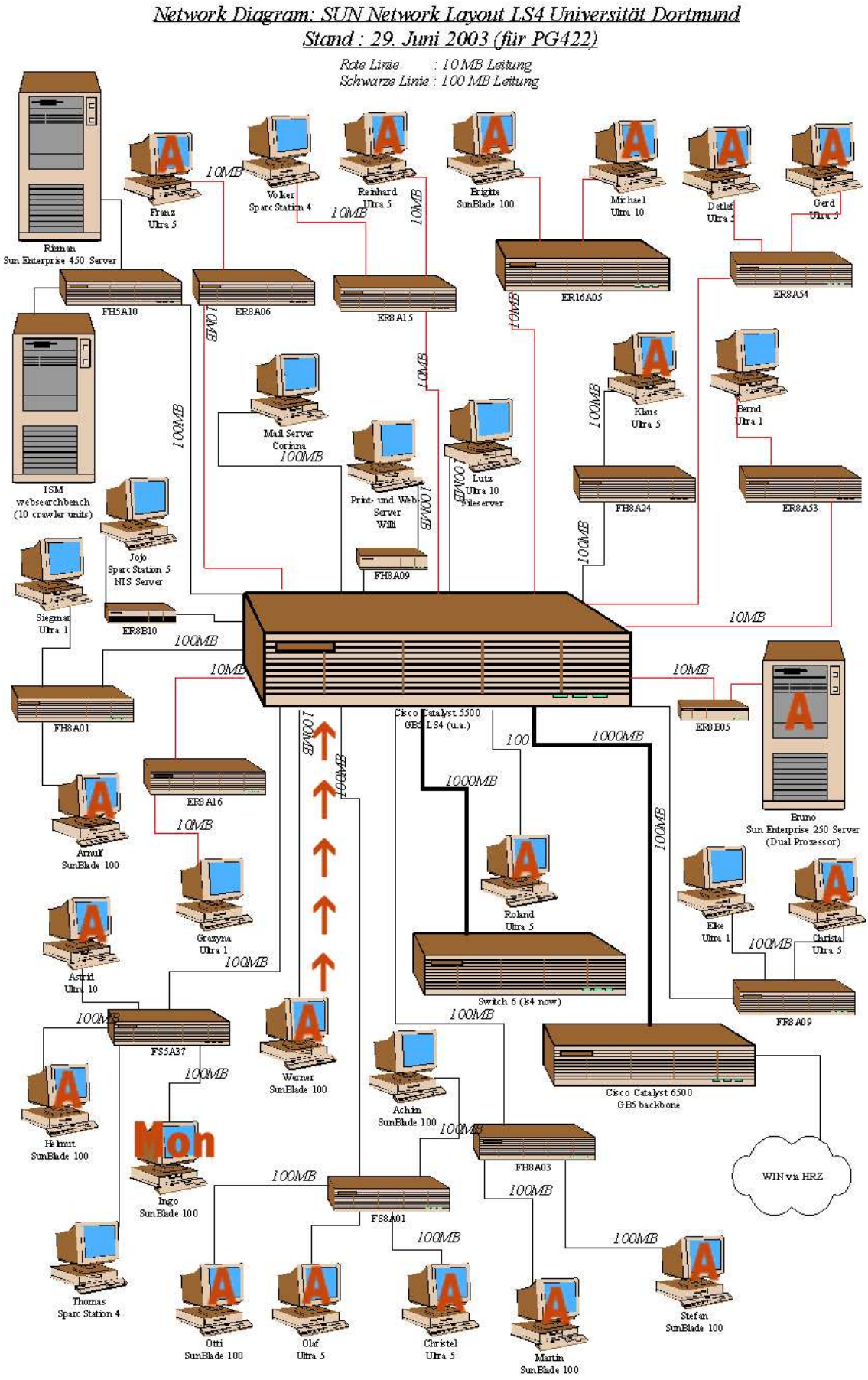



Abbildung 4.3: Das Sun-Netzwerk des LS4 mit Agenten und Monitor

#### IRB vs. PG422

Nachdem abzusehen war, dass Speedtrap einsatzbereit war, wurden die oben genannten zwei Punkte (Freischaltung des Cisco 5500 und Installation von SNMP Agenten auf den Sun) angefragt. Während es kein Problem war, die Zusage zu erhalten, vollen SNMP- Zugriff auf den Cisco 5500 zu bekommen gab es leider bei den SNMP-Agenten von Seiten der IRB keine Bereitschaft zur Zusammenarbeit. Mit dem Hinweis auf Sicherheitsrisiken wurden alle benötigten Hilfen verweigert. Aus dem gleichen Grunde musste im Übrigen das NIC Backend komplett bei einigen PG Mitgliedern zu Hause entwickelt werden.

#### Alternativer Meßaufbau 1

Die studentischen PG Mitglieder und Jürgen einigten sich daher darauf, eine kleine Testumgebung im Raum 438 aufzubauen basierend auf dem Cisco 2900 Switch und den folgenden Rechnern, die alle direkt an diesen angeschlossen wurden:

Typ Sun Ultra1 : bernd, elke
Typ Sun Ultra5 : christa, reinhard, roland
Typ Sun Blade 100 : achim, ingo, martin
2 Laptops : ls4int05, ls4int07

Insgesamt 10 Rechner.

Der Rechner Achim wurde dazu aus Raum 433 geholt.

Allerdings sind die Sun Ultra 1 / Ultra 5 nicht für die verteilte Berechnung geeignet, da sie über zu wenig Hauptspeicher verfügen. (Benötigt werden umgefähr 900 MB RAM Speicher.) Diese Voraussetzung erfüllen am LS4 nur die Sunblade 100 Maschinen und der Enterprise 250 Server, die über jeweils 2GB Ram verfügen. Laut Markus Fischer sind die älteren Maschinen hauptsächlich damit beschäftigt, Daten auszulagern (Swapping) als wirklich produktiv an der Berechnung teilzunehmen. Da der Lehrstuhl allerdings im Jahr 2002 nicht über 10 SunBlades verfügt hat, wurden wohl doch einige der älteren Maschinen eingesetzt (Abbildung 4.1).

#### Alternativer Meßaufbau 2

Von daher wurde der Versuchsaufbau ein weiteres Mal modifiziert. (Abbildung 4.4) An den Switch wurden nur noch die folgenden Rechner gehängt:

- 3 SunBlade 100, die an der verteilten Berechnung teilnehmen sowie zusätzlich die SunBlade 100 "werner", die über den Uplink zum Lehrstuhlnetz eingebunden ist.
- 2 Laptops, von denen der ls4int07 den Monitor beherbergt und der ls4int05 einen Lastgenerator
- 2 Sun Ultra 1, die als Lastgeneratoren zur Verfügung stehen.

Am 11.07.2003 wurden von 8 - 17 Uhr die verschiedenen Messungen durchgeführt.

## 4.2 Messauftrag LS4

Im folgendem wird die Durchführung einer Messung am LS4 beschrieben. Im Anschluss folgt die Auswertung der Messergebnisse.

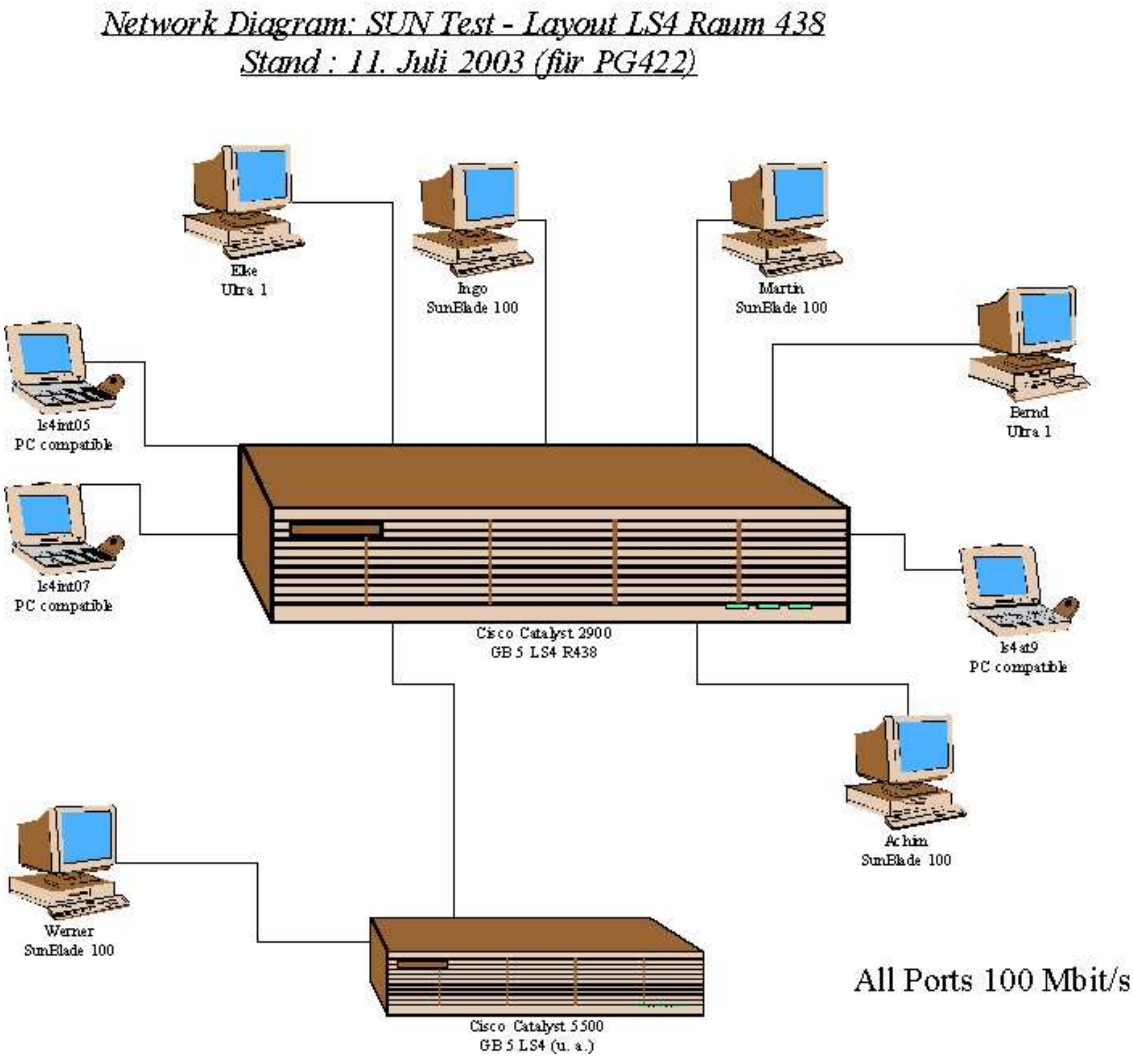



Abbildung 4.4: Der Testaufbau Raum 438 LS4

**Testablauf am LS4**

Zur Vorbereitung der Messug wird das Netzwerk im Raum 438 GB5 aufgebaut. Dieses Netzwerk besteht aus einem Switch (Cisco Catalyst 2900 mit 16 Port, 2 Uplinkports), fünf Sun-Rechnern (”bernd”, ”martin”, ”achim”, ”elke” und ”ingo”), und drei Notebooks. Zwei von denen gehören zum LS4 und einer wird von einem PG Mitglied mitgebracht.
Der den Switch abfragende Agent wird auf dem Rechner ”martin” installiert. Der Monitor wird auf dem Notebook ”ls4int07” gestartet. Die automatische Netzplan-Erkennung wird gestartet. Der Netzplan wird fehlerfrei erkannt und graphisch angezeigt (Abbildung 4.5).

Die Messung besteht aus einem 30 minütigem Messintervall. Zeitgleich wird das Programm für die „Verteilte Berechnung von Markov Ketten” mit den besagten 4 Rechnern gestartet. Dessen Berechnung ist zwar nach einer halben Stunde noch nicht beendet, doch zeichnet Speedtrap die characteristischen Datentransfers in dieser Zeit schon auf.

Da wir nur noch mit einer isolierten Laborlösung anstelle des gesamten LS4-Netzes arbeiten können, wird eine Messung für ausreichend erachtet. Nach Beenden des Messauftrages zeigt sich,

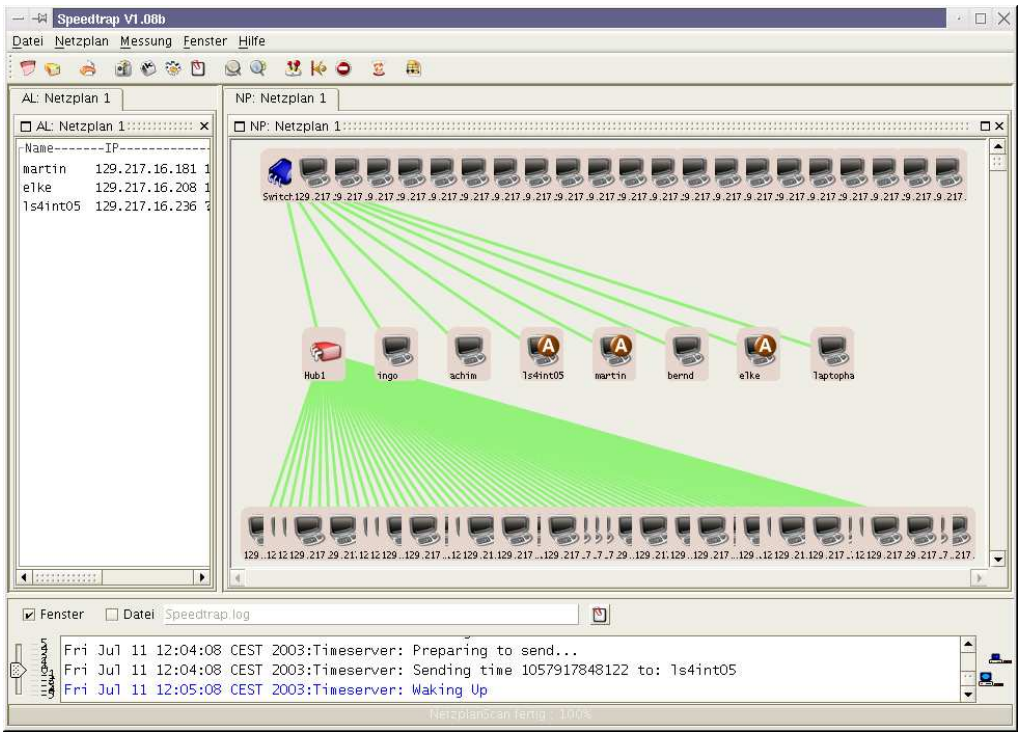



Abbildung 4.5: Ergebnis der automatischen Netzplanerkennung

dass alle Komponenten von Speedtrap einwandfrei funktioniert haben, es gibt keine Abstürze oder Java-Exceptions. Die Messergebnisse werden in zwei Formaten abgespeichert: Einmal in Textformat(.txt) zur automatischen Konvertierung in eine Exceltabelle und einmal im monitoreigenem Format(.nmp). Daten, die im .nmp-Format vorliegen, lassen sich zur graphischen Analyse vom Monitor nocheinmal abspielen, genau so, wie sie aufgenommen wurden. Der genaue Verlauf der Messung kann sich so nocheinmal angesehen werden. (Abbildung 4.6)

Der Cisco Catalyst 2900 hat 16 100 Mbit Full Duplex Ports und 2 Glasfaser Uplink Ports mit je 1 Gbit.

| Port | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Rechner | - | uplink* | - | ingo | - | achim | - | ls4int05 | ls4int07 |
| Port | 10 | 11 | 12 | 13 | 14 | 15 | 16 | - | - |
| Rechner | martin | - | - | bernd | - | elke | ls4at9* | - | - |

Tabelle 4.1: Portbelegung am LS-4 Testswitch

uplink* = Uplink an der Dose 438-05
ls4at9* = Hayats Laptop

Tabelle 4.1 zeigt die Portbelegung für den Test am LS4. Wie zu erkennen, wurden nicht die üblichen Uplink Ports verwendet, da kein Glasfaser Anschluss zur Verfügung steht. Am Port 2 ensteht Traffic, wenn die an der Berechnung in Raum 438 beteiligten SunBlades mit der Sun Blade "Werner" Daten austauschen, sowie wenn die Suns mit dem NIS Server "jojo" kommunizieren.

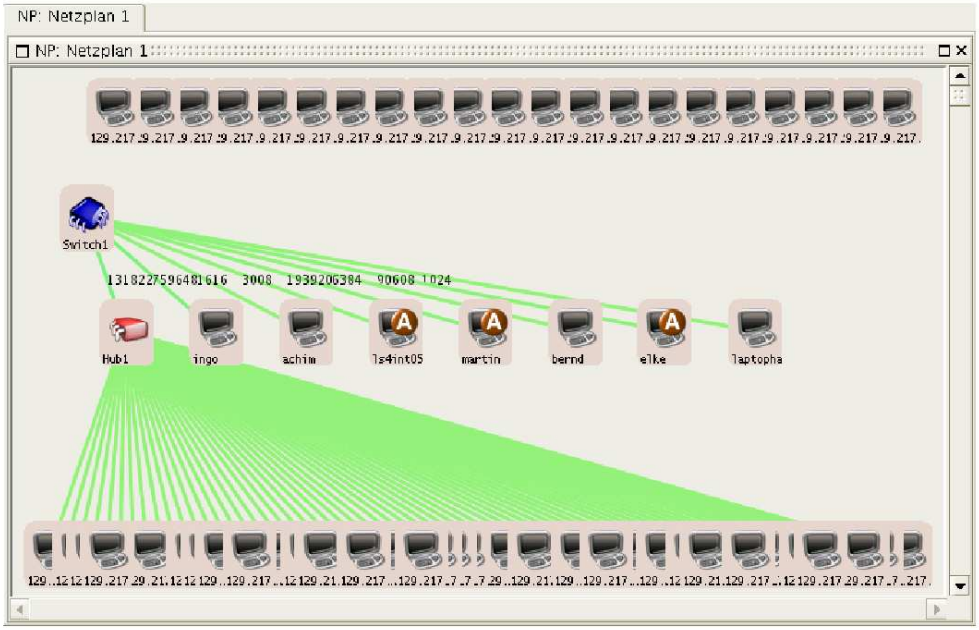



Abbildung 4.6: Messung mit dynamischer Anzeige der Pakete

Ausser dem wird hier natürlich aller sonstiger Netzverkehr (http, ssh, rdesktop) durchgeleitet.

Die MTU, die von Speedtrap, erkannt wurde beträgt 1500 Bytes. Die Verkabelung war einwandfrei, denn es wurden keine Fragmente (beschädigte Pakete) gezählt. Häufige Ursache für Fragmente sind defekte Netzwerkkomponenten, insbesondere NICs und Netzwerkkabel.

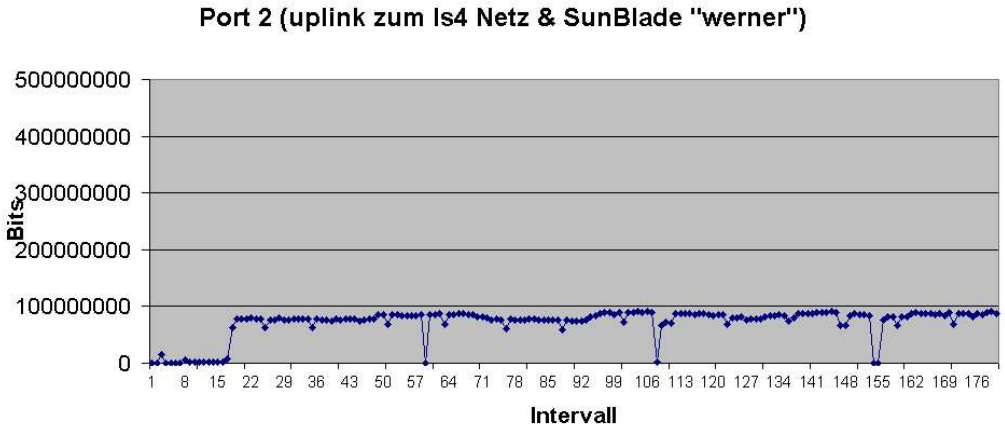



Abbildung 4.7: Datendurchsatz am Uplinkport zum ls4 bzw. zum Rechner "werner"

Die Analyse der Messdaten ergibt zuerst einmal, dass die Dose 438-05, die den Uplink für unser System darstellt, nur mit 10 Mbit beschaltet ist (Abbildung 4.7). Dies deckt sich mit dem Übersichtsplan des LS4 vom 29. Juni 2003 (Abbildung 4.2). An dieser Dose hängt normalerweise der Rechner "bernd" allein. Er ist verbunden mit dem Ethernet Repeater ER8A54, der nur ein

10 Mbit/s Gerät ist. Die Kurve verdeutlicht einwandfrei, dass hier über nahezu den gesamten Zeitraum das Maximum einer 10 Mbit-Verbindung genutzt wird. Schon hier kann man feststellen, besonders wenn man die vertikalen Skalenbereiche vergleicht, dass für besagtes verteiltes Berechnen 10 Mbit-Anbindungen zu klein dimensioniert sind. Der Rechner "werner" verhungert quasi und kann nicht das volle Potential seiner CPU ausspielen. Ausserdem werden auch die Rechner mit schneller Anbindung aufgehalten, weil diese auf die Teilergebnisse der langsam angebundenen Rechner warten müssen. Wenn man nun den LS4 Netzplan betrachtet (Abbildung 4.2), stellt man fest, dass unter Anderem der Rechner "bruno", der über die höchste CPU- Leistung verfügt (Dualprozessormaschine), auch nur über 10Mbit angeschlossen ist. Dieser Sun Enterprise 250 Server wird seine Rechenleistung nie ausspielen können. Das Netz ist hier eindeutig der Flaschenhals. Je mehr Rechner sich im Netz befinden und je mehr davon nur mit 10 Mbit/s angebunden sind, um so länger muss jede einzelne Maschine warten, bis sie mit neuen Zwischenergebnissen von den anderen Maschinen versorgt wird.

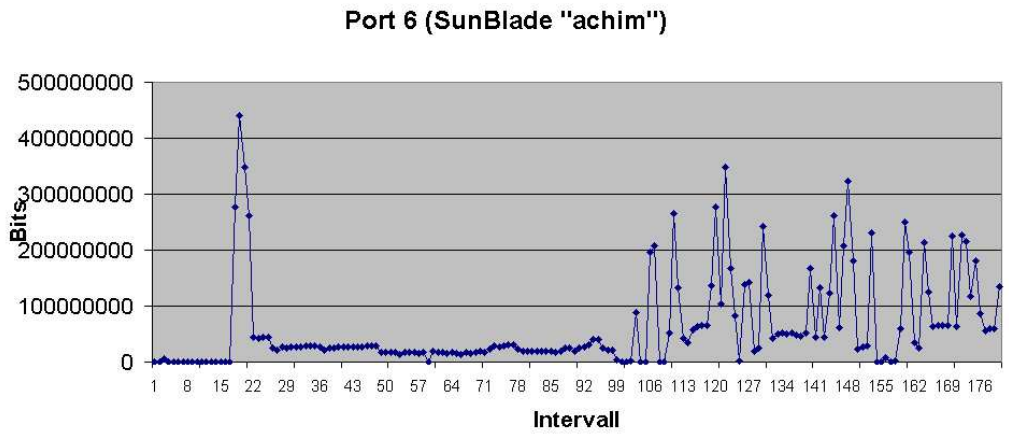



Abbildung 4.8: Datendurchsatz am Port zum Rechner "achim"

Bei den Rechnern "martin", "achim" (Abbildung 4.8) und "ingo" erkennt man, dass alle 3 zu Anfang eine ganze Menge an Daten erhalten. Danach sind sie für eine ganze Weile beschäftigt, ohne Daten auszutauschen. Zum Ende des Messzeitraumes scheinen alle SunBlades kräftig Daten auszutauschen. Sie erreichen dabei aber nie mehr als 40% der theoretisch verfügbaren Kapazität von 100Mb/s. Dies spricht dafür, dass 100Mbit/s als Netzkapazität für die gestellte Aufgabe absolut ausreichend sind. Der Rechner "ls4int05" hatte im Verlauf der Messung keine Aufgabe, da er nur als Lastgenerator dient. Dementsprechend entstand an seinem Port nur vernachlässigbar wenig Traffic. (Abbildung 4.9)

Ls4int07 beherbergt den Monitor. Allerdings wird hier nur unwesentlich mehr Traffic gemessen als am ls4int05. Dies spricht für Speedtrap und sein Konzept, alle Messdaten über UDP Pakete mit dem geringsten Overhead im Netz zu transportieren. (Abbildung 4.10) Die beiden Sun Ultra 1 "bernd" und "elke" fallen auch nicht weiter ins Gewicht, da ihre Lastgeneratoren für diese Messreihe abgeschaltet waren. Das Laptop "ls4at09" produzierte nur zu einem Zeitpunkt Last, als durch Fehlbedienung ein Download aus dem Internet gestartet wurde. Allerdings zeigt die Grafik, dass dabei nicht so viele Daten übertragen wurden pro Intervall, wie wenn die SunBlades sich ihre Teilergebnisse untereinander mitteilen. (Abbildung 4.11)

Darauf hin wurde analysiert, ob der Datenaustausch nach einem bestimmten Muster abläuft. Dann könnte man das Programm zur Berechnung der Markov Ketten so anpassen, dass durch effizientes Nutzen der 10Mbit-Anbindungen diese nicht mehr den Flaschenhals im System darstellen. Allerdings zeigt Abbildung 4.12, dass keinerlei Regelmäßigkeiten zu erkennen sind. (In Grafik 4.12

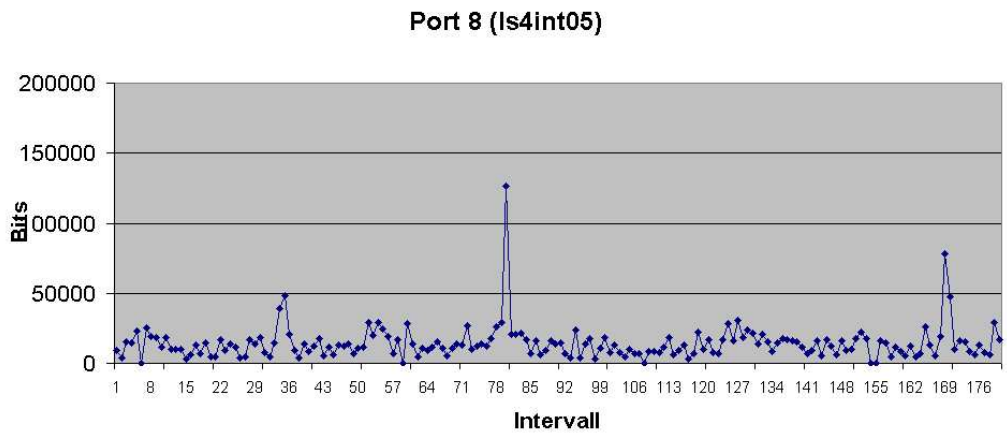



Abbildung 4.9: Datendurchsatz am Port zum Rechner "ls4int05"

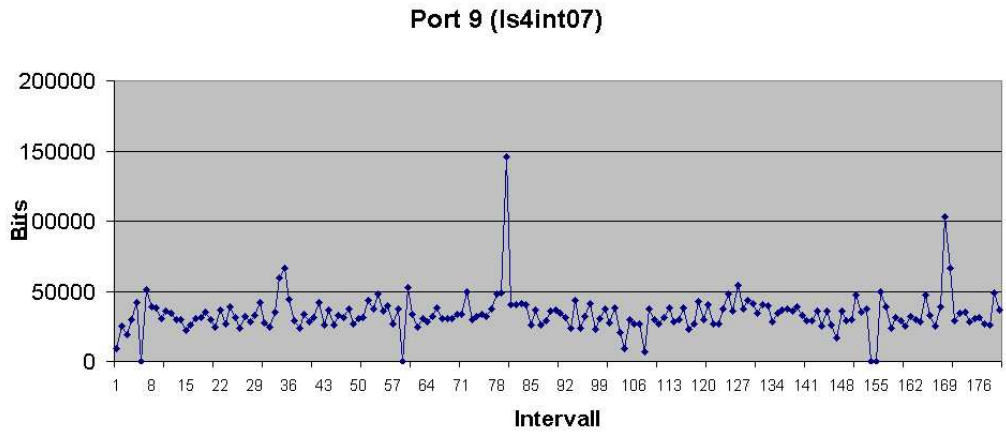



Abbildung 4.10: Datendurchsatz am Port zum Rechner "ls4int07" (Monitor)

wurden nur die Gesamtbits der an der Berechnung beteiligten SunBlades in einem Diagramm zusammengefasst.) Hier ist auch deutlich zu sehen, dass 10Mbit viel zu wenig ist. Da 100Mbit-Anbindungen heute Stand der Technik ist und 1 GBit auch schon technisch möglich ist, macht es natürlich auch keinen Sinn, die Software, die Markus Fischer einsetzt, auf veralterte Technik anzupassen.

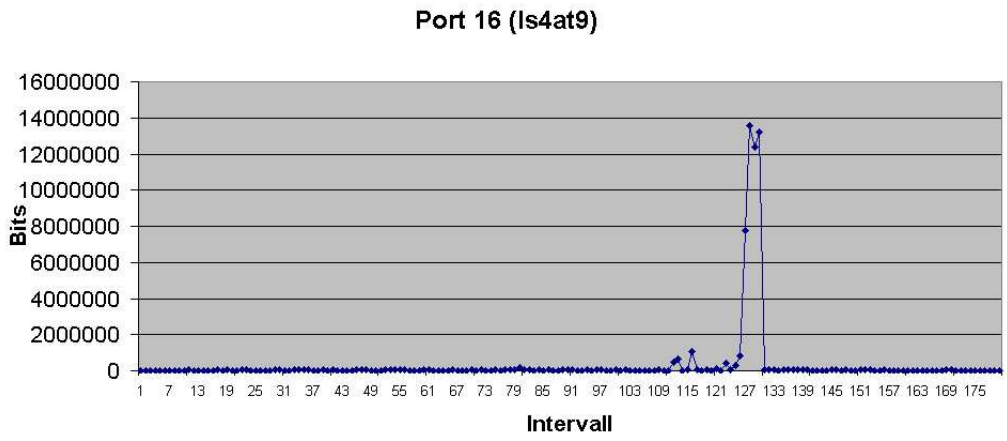



Abbildung 4.11: Datendurchsatz am Port zum Rechner "ls4at9"

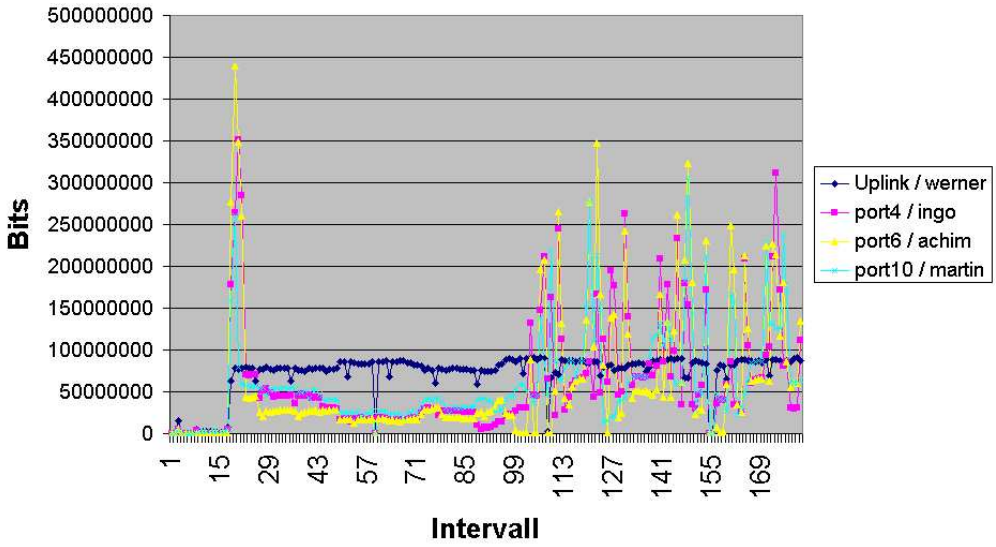



Abbildung 4.12: Datendurchsatz aller 4 SunBlades zusammen

## 4.3 Verifikation

SimulaNT ist ein Softwarewerkzeug, um eine Leistungsbewertung eines Ethernetzes durchzuführen. Benutzer können mit Hilfe einer graphischen Oberfläche die verschiedenen Parameter eines Netzwerks festlegen und nach einer Simulation durch das Programm eine Aussage über die Performanz des Netzes treffen. SimulaNT ist am Lehrstuhl IV des Fachbereichs Informatik der Universität Dortmund im Rahmen der Arbeit der Projektgruppe PG358 POL/NT [P358EB] entwickelt worden. Um Speedtrap zu verifizieren, wird also das Tool der Vorgänger-PG benutzt.

SimulaNT liefert nicht, wie Speedtrap, Werte zu Intervallen. Am Ende der Simulation werden lediglich die Werte für den gesamten simulierten Zeitraum geliefert. Die Stärke von SimulaNT liegt darin, dass Übertragungszeiten genaustens ermittelt werden. So lässt sich beispielsweise leicht ermitteln, um wieviel schlechter die Leistung eines Netzwerks ist, wenn man an einer bestimmten Stelle einen Hub anstelle eines Switches einsetzt, oder wie sich längere Kabel in einem Segment auswirken.

Für die Verifikation wird die Version 1.1 von SimulaNT gewählt. Im Rahmen einer Diplomarbeit wurde zwar eine neuere Version 2.0 entwickelt, aber die Neuerungen sind für die Verifikation von Speedtrap nicht relevant. Version 2.0 ist in der Lage eine Leistungsbewertung von Lastkontrollverfahren durchzuführen. Die Lastgeneratoren von Speedtrap versenden aber lediglich einfache UDP-Pakete.

### 4.3.1 Messungen

Aufgrund der Tatsache, dass SimulaNT für die Simulation konkrete Lastbeschreibungen erwartet und wir diese für die bisherigen Messungen nicht kennen, wird entschieden, nicht die Messung zu verifizieren sondern das Tool Speedtrap selbst.

SimulaNT braucht für die zu simulierenden Lasten Start- und Zieladressen, sowie Verteilungsfunktionen über die Zeitpunkte, zu denen sie auftreten. Diese sind nicht bekannt für die bisherigen Messungen und Speedtrap kann die Start- und Zieladressen nicht messen. Es wäre aber auch unsinnig, gemessene Daten als Eingabe für die Simulation zu nutzen, da diese bereits falsch sein könnten.

Für die Verifikation wird eine Meßreihe gestartet in der kontrollierte Lasten erzeugt werden, dadurch sind die auftretenden Lasten bekannt. Hierzu werden die Lastgeneratoren der Agenten von Speedtrap benutzt. Die Meßreihe besteht aus drei Durchgängen und wird protokolliert:

**1. Messung:**

Dauer: 1 Minute
Last: Der Rechner "ingo" schickt für die Dauer von 60 Sekunden an den Rechner "elke" jede Sekunde 10 Pakete der Größe 64KB.
(10 pro Sekunde, 1 Minute lang) insgesamt 600 Pakete, 38400 KB entsprechend 37,5 MB

Die Messergebnisse am Rechner "ingo" zeigt Tabelle 4.2:

Bewertung zum Rechner "ingo":
Die Diagramme zeigen, dass die Netzinfrastruktur problemlos mit der Datenmenge, die über den Switch versendet wurde, zurechtkommt. Dafür spricht auch der nahezu lineare Verlauf der Lastkurve in den Abbildungen 4.13 und 4.14. Darüber hinaus zeigt Abbildung 4.15, dass der Zielrechner fast keinen Traffic während der Messung erzeugt hat, die Lastgenerierung also einwandfrei als Einwegekommunikation erfolgt. Allerdings erreicht der Lastrechner "ingo" nicht die gewünschte Datenlast von 640KB pro Sekunde. Wenn man die Meßergebnisse etwas umformt ergibt sich folgendes:

| Intervall | Bits gesendet | Bits empfangen | Bits insgesamt |
|---|---|---|---|
| 0 | 26268800 | 59240 | 26330240 |
| 1 | 33104832 | 85144 | 33192754 |
| 2 | 33124584 | 97824 | 33225202 |
| 3 | 33098688 | 83664 | 33185124 |
| 3 | 33108496 | 85832 | 33197110 |
| 4 | 32591824 | 93600 | 32688168 |

Tabelle 4.2: Messwerte am Rechner "ingo"

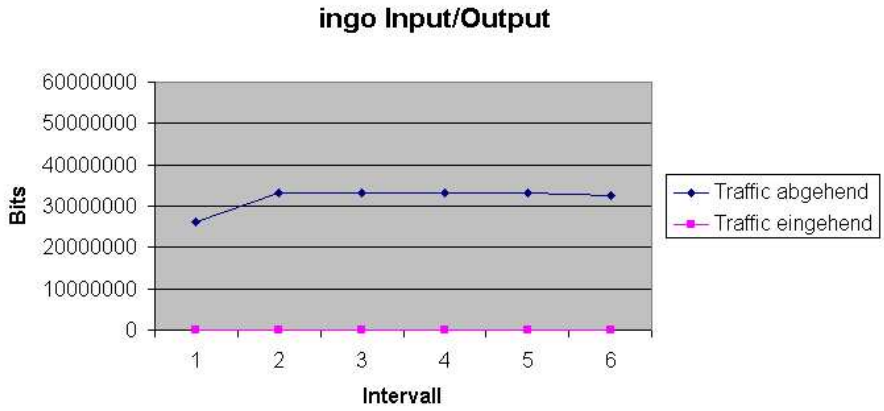



Abbildung 4.13: Datendurchsatz am Port 4 zum Rechner "ingo"

$33.104.832 Bits/10 Sekunden$ (Intervallänge)
$\Leftrightarrow 3.310.483 Bits/s$
$\Leftrightarrow 413810 Bytes/s$
$\Leftrightarrow 404 KBytes/s$

Das entspricht lediglich rund 63% der gewünschten 640 KB/s (10 Pakete der Größe 64 KB). Es stellt sich daher die Frage, ob die Java Virtual Machine hier nicht leistungsfähig genug war oder die eingebaute Netzwerkkarte der Sun Workstation?

Erstaunlich ist im Übrigen auch der Peak im Graph für den Zielrechner Elke (Abbildung 4.16). Vermutlich wurde von ausserhalb auf den Rechner zugegriffen oder der Switch hat gerade im Moment der Messung seinen Bufferinhalt über die Leitung zu "elke" übertragen. Leider ist es wegen der Sun Philosophie ("Netzwerkcomputer") nicht möglich, eine isolierte Struktur zu erschaffem, weil die Suns dann nicht mehr mit dem NIS-Server in Verbindung treten können und auch keine Programmdateien mehr laden oder speichern können. Lediglich die /tmp-Verzeichnisse halten die Sun Workstations lokal auf der eingebauten Festplatte.

Das Ziel der versendeten Pakete ist der Rechner "elke". Die dazugehörigen Graphen dazu sind in den Abbildungen 4.15 und 4.16 .

Tabelle 4.3 zeigt die Messwerte am Rechner "elke".

Bewertung zum Rechner "elke":
Im Mittel stimmen die Anzahl der empfangenen Bit mit denen der von "ingo" gesendeten Bits überein. Die Daten sind daher als konsistent anzusehen.

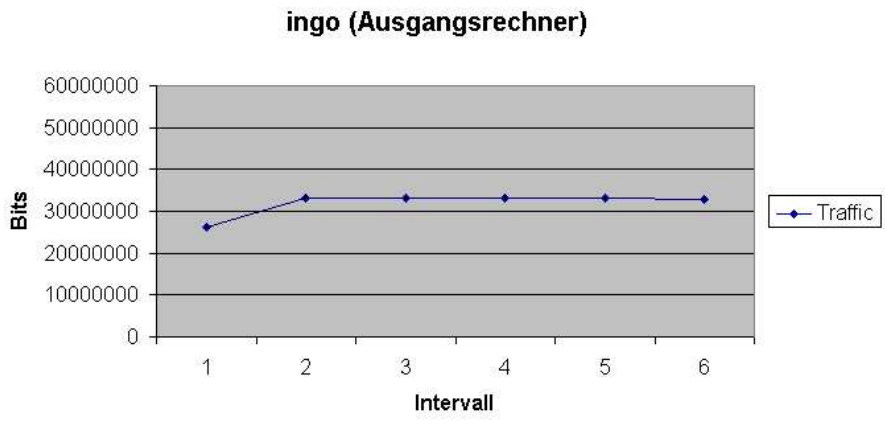



Abbildung 4.14: Gesamtdatendurchsatz am Port 4 zum Rechner "ingo"

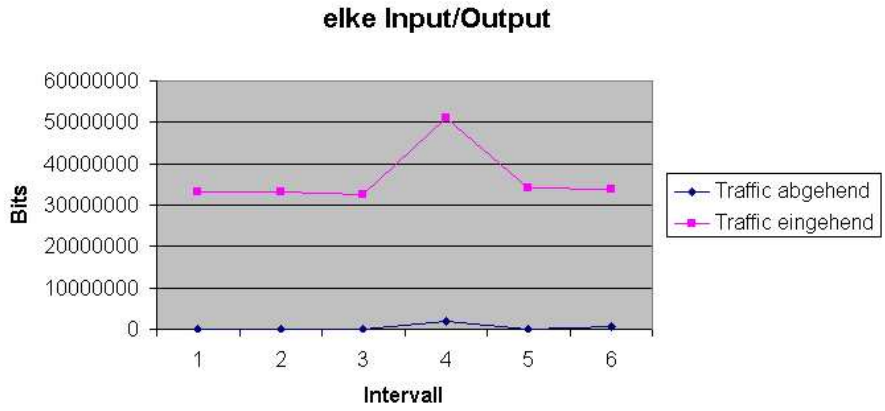



Abbildung 4.15: Datendurchsatz am Port 4 zum Rechner "elke"

**2. Messung:**

Dauer: 2 Minuten
Last: Der Rechner "ingo" schickt für die Dauer von 120 Sekunden an den Rechner "elke" jede Sekunde 1 Paket der Größe 64KB. (1 pro Sekunde, 2 Minuten lang) insgesamt 120 Pakete, 7680 KB entsprechend 7,50 MB
Last: Der Rechner "bernd" schickt für die Dauer von 120 Sekunden an den Rechner "elke" jede Sekunde 2 Pakete der Größe 32KB. (2 pro Sekunde, 2 Minuten lang) insgesamt 240 Pakete, 7680 KB entsprechend 7,50 MB

Die Messergebnisse am Rechner "ingo" zeigt Tabelle 4.4 . Die dazugehörigen Graphen dazu sind in den Abbildungen 4.17 und 4.18 .

Bewertung zum Rechner "ingo":
Da die generierte Arbeitslast pro Lastgenerator um den Faktor 10 geringer ist als in der 1. Messung gibt es diesmal keine Performance Probleme von Java oder der Hardware.

| Intervall | Bits empfangen | Bits gesendet | Bits insgesamt |
|---|---|---|---|
| 0 | 33111648 | 69088 | 33180803 |
| 1 | 33110016 | 65520 | 33175599 |
| 2 | 32588112 | 75024 | 32663207 |
| 3 | 51059720 | 1965520 | 53027039 |
| 4 | 34020072 | 98352 | 34118530 |
| 5 | 33972720 | 743824 | 34717076 |

Tabelle 4.3: Messwerte am Rechner "elke"

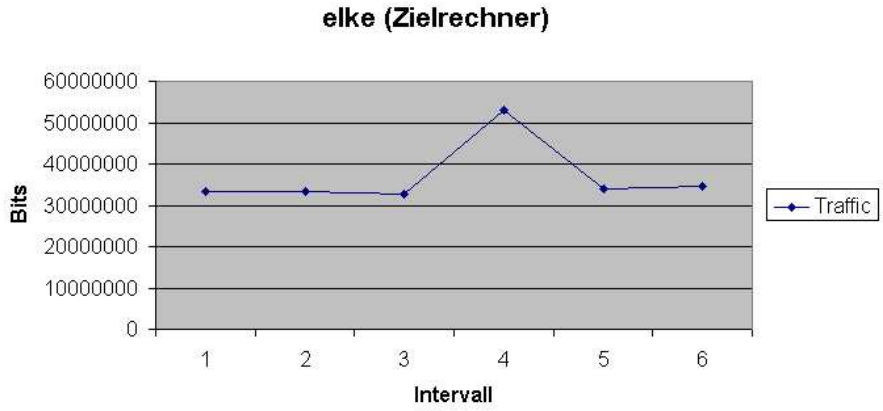



Abbildung 4.16: Gesamtdatendurchsatz am Port 4 zum Rechner "elke"

| Intervall | Bits gesendet | Bits empfangen | Bits gesamt |
|---|---|---|---|
| 0 | 3161056 | 22736 | 3184064 |
| 1 | 3684432 | 18808 | 3703555 |
| 2 | 4756576 | 57560 | 4814555 |
| 3 | 4734336 | 318246 | 4766562 |
| 4 | 4728384 | 15416 | 4744196 |
| 5 | 4755088 | 57416 | 4812923 |
| 6 | 4728384 | 24056 | 4752836 |
| 7 | 4729424 | 25968 | 4755789 |
| 8 | 4746872 | 57152 | 4804435 |
| 9 | 4734432 | 37600 | 4772434 |
| 10 | 4728384 | 23744 | 4752524 |
| 11 | 4227928 | 58040 | 4286341 |

Tabelle 4.4: Messwerte am Rechner "ingo"

Gemessen wurden rund 4.756.576 Bits pro Intervall.

Gesendet werden 64KBytes/s bzw. 2 * 32KBytes/s.
$\Leftrightarrow 65.536 Bytes/s$
$\Leftrightarrow 524.288 Bits/s$
$\Leftrightarrow 5.242.880 Bits/Intervall$ (Intervallänge 10 Sekunden)

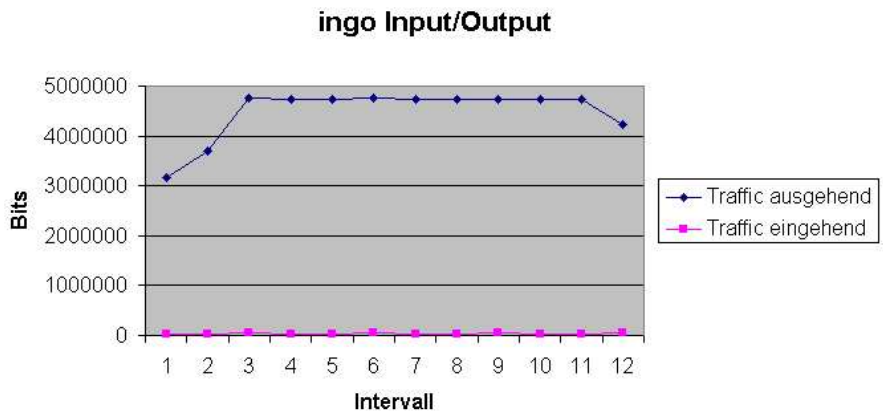



Abbildung 4.17: Datendurchsatz am Port 4 zum Rechner "ingo"

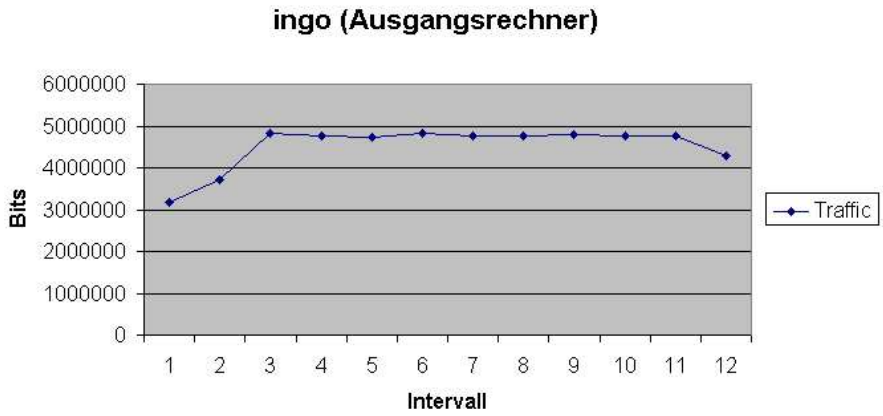



Abbildung 4.18: Gesamtdatendurchsatz am Port 4 zum Rechner "ingo"

Vergleicht man diese Zahl nun mit den gemessenen Werten von rund 4.756576 b/Intervall, so kommt man auf eine Abweichung von rund 9%.

Für den Rechner "bernd" wird der graphische Verlauf der Messung in den Abbildungen 4.19 und 4.20 dargestellt.

Bewertung:
Auch beim Rechner "bernd" ergibt sich eine durchschnittliche Abweichung von 9%, ähnlich wie beim Rechner "ingo".

Für den Rechner "elke" wird der graphische Verlauf der Messung in den Abbildungen 4.21 und 4.22 dargestellt.

Bewertung:
Die Messergebnisse für den Rechner "elke" sind schlüssig, denn die Anzahl empfangener Bits entsprechend weitestgehend der Summe der von den beiden Lastgeneratoren erzeugten Anzahl bits. Eine gewisse Differenz läßt sich nicht vermeiden, weil die Art der Netzcomputer nicht ausschließt, dass andere Prozesse während der Messungen auch Daten erzeugen.

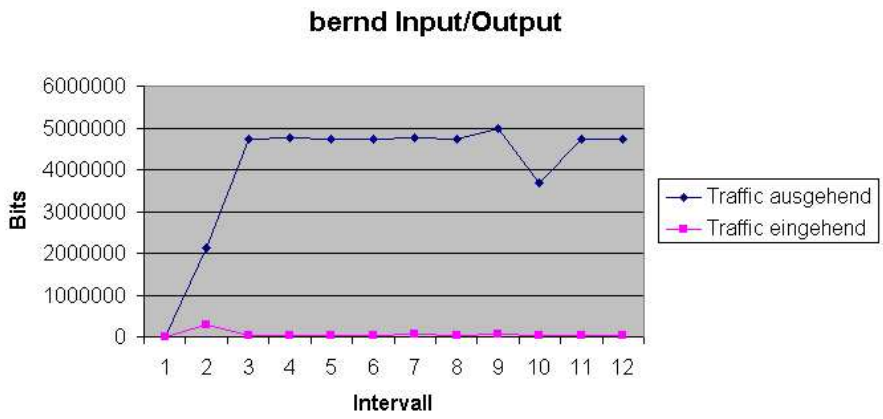



Abbildung 4.19: Datendurchsatz am Port 4 zum Rechner "bernd"

| Intervall | Bits gesendet | Bits empfangen | Bits gesamt |
|---|---|---|---|
| 0 | 11360 | 14456 | 25825 |
| 1 | 2113680 | 298360 | 2412230 |
| 2 | 4728384 | 41784 | 4770564 |
| 3 | 4749808 | 46376 | 4796599 |
| 4 | 4728384 | 26824 | 4755604 |
| 5 | 4728384 | 37384 | 4766164 |
| 6 | 4752464 | 58576 | 4811456 |
| 7 | 4730768 | 40992 | 4772159 |
| 8 | 4991072 | 48712 | 5040202 |
| 9 | 3685472 | 38216 | 3724002 |
| 10 | 4741504 | 45536 | 4787448 |
| 11 | 4728384 | 45176 | 4773956 |

Tabelle 4.5: Messwerte am Rechner "bernd"

**3. Messung:**

Dauer: 5 Minuten
Last: Der Rechner "ingo" schickt für die Dauer von 300 Sekunden an den Rechner "elke" jede Sekunde 1 Paket der Größe 64KB. (1 pro Sekunde, 5 Minuten lang) insgesamt 300 Pakete, 19200 KB entsprechend 18,75 MB
Last: Der Rechner "bernd" schickt für die Dauer von 300 Sekunden an den Rechner "elke" jede Sekunde 2 Pakete der Größe 32KB. (2 pro Sekunde, 5 Minuten lang) insgesamt 600 Pakete, 19200 KB entsprechend 18,75 MB

Die Messergebnisse für den Rechner "ingo" zeigen die Abbildungen 4.23 und 4.24 .

Bewertung zu den Graphen:
Auffällig sind die Einbrüche bei der Übertragung der Bits in den Meßintervallen 6, 16 und 24. Sie treten bei beiden Lastgeneratoren zeitgleich auf. Eventuell hat hier der Switch ein Performance Problem. Hier ist zu hinterfragen, in wieweit die eingesetzten Switche überhaupt eine lineare Leistung garantieren.

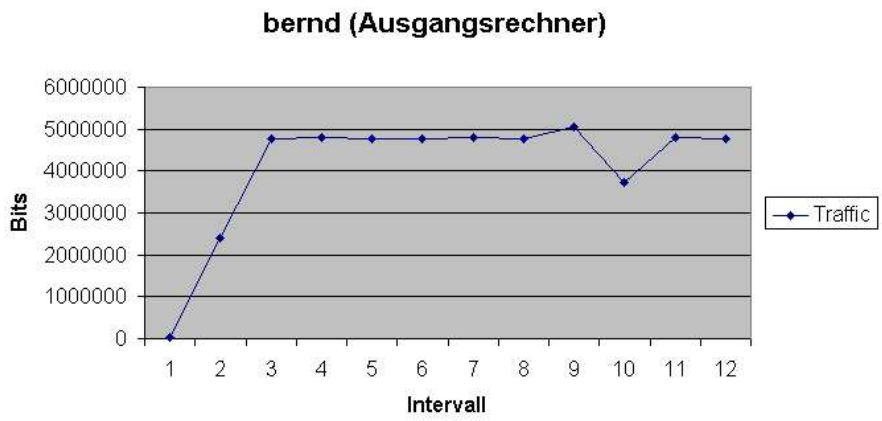



Abbildung 4.20: Gesamtdatendurchsatz am Port 4 zum Rechner "bernd"

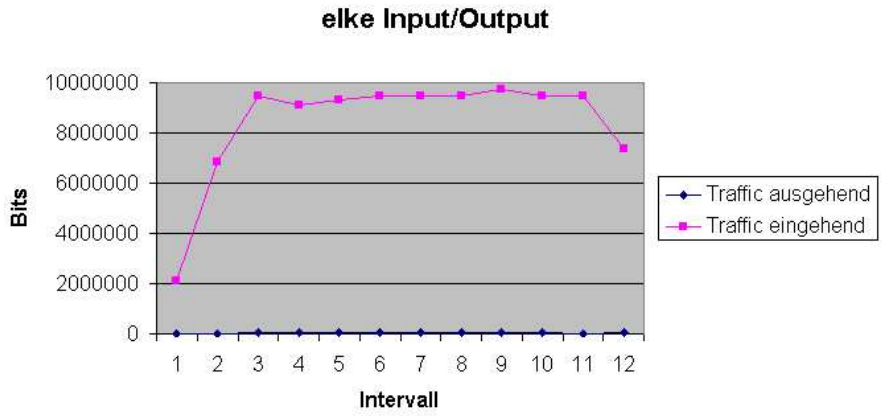



Abbildung 4.21: Datendurchsatz am Port 4 zum Rechner "elke"

Die Messergebnisse zum Rechner "bernd" zeigen die Abbildungen 4.25 und 4.26 .

Tabelle 4.8 zeigt die Bits insgesamt pro Intervall für Port 13 (bernd).

Bewertung:
Siehe Performanceeinbrüche bei den Intervallen 6, 16 und 24, die zur gleichen Zeit auch am Rechner "ingo" auftreten.

Die Messergebnisse zum Rechner "elke" zeigen die Abbildungen 4.26 und 4.27 .

Tabelle 4.9 zeigt die Bits insgesamt pro Intervall für Port 15 (elke).

Bewertung:
Messung 3 entspricht von den Werten (Lastgröße, Lastpfade) genau denen, die auch bei Messung 2 eingestellt wurden, nur die Dauer des Meßprozesses ist nun fünf Minuten und nicht nur zwei. Die gemessenen Werte unterscheiden sich nur marginal von denen aus Messung 2. Von daher kann man hier von gesicherten Werten ausgehen, die in SimulaNT weiterverarbeitet bzw. verifiziert werden

| Intervall | Bits empfangen | Bits gesendet | Bits insgesamt |
|---|---|---|---|
| 0 | 2106760 | 7104 | 2113871 |
| 1 | 6838296 | 20032 | 6858348 |
| 2 | 9495800 | 43568 | 9539407 |
| 3 | 9086344 | 27552 | 9113923 |
| 4 | 9320792 | 29024 | 9349844 |
| 5 | 9495864 | 47840 | 9543746 |
| 6 | 9474704 | 28848 | 9503580 |
| 7 | 9478512 | 29920 | 9508461 |
| 8 | 9760312 | 41632 | 9801982 |
| 9 | 9480360 | 28080 | 9508467 |
| 10 | 9471664 | 22880 | 9494571 |
| 11 | 7394264 | 46800 | 7441100 |

Tabelle 4.6: Messwerte am Rechner "elke"

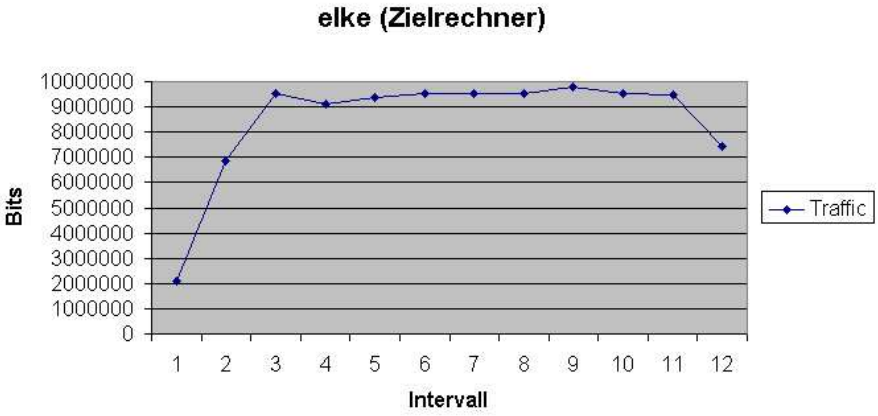



Abbildung 4.22: Gesamtdatendurchsatz am Port 4 zum Rechner "elke"

sollten.

Abschließende Anmerkung: In allen Graphen wurden einige Unicast- und sonstige Pakete nicht berücksichtigt. Die Maximum Transmission Unit (MTU) im Testnetz ist 1500 Bytes groß.

Die Verifikation dieser Meßreihen mit SimulaNT beschreibt das folgende Kapitel.

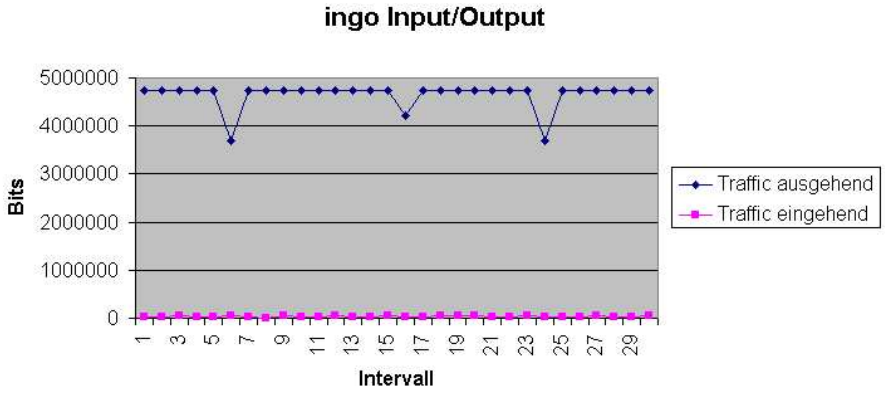



Abbildung 4.23: Datendurchsatz am Port 4 zum Rechner "ingo"

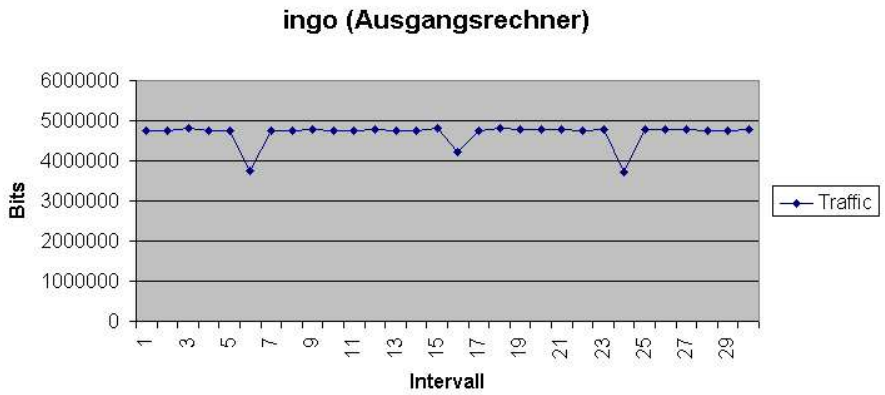



Abbildung 4.24: Gesamtdatendurchsatz am Port 4 zum Rechner "ingo"

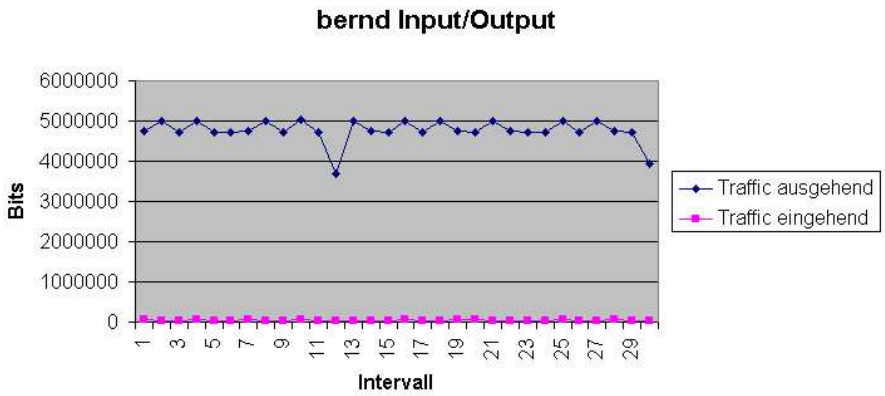



Abbildung 4.25: Datendurchsatz am Port 4 zum Rechner "bernd"

| Intervall | Bits gesendet | Bits empfangen | Bits gesamt |
|---|---|---|---|
| 0 | 4728384 | 24184 | 4752964 |
| 1 | 4728384 | 14760 | 4743540 |
| 2 | 4749528 | 47120 | 4797061 |
| 3 | 4728384 | 21448 | 4750228 |
| 4 | 4728384 | 15672 | 4744452 |
| 5 | 3697448 | 40792 | 3738564 |
| 6 | 4730520 | 15456 | 4746374 |
| 7 | 4728384 | 13016 | 4741796 |
| 8 | 4748296 | 44272 | 4792980 |
| 9 | 4728384 | 29280 | 4758060 |
| 10 | 4728384 | 18448 | 4747228 |
| 11 | 4750464 | 44392 | 4795270 |
| 12 | 4730520 | 19776 | 4750694 |
| 13 | 4728384 | 14640 | 4743420 |
| 14 | 4749528 | 51472 | 4801413 |
| 15 | 4203008 | 27280 | 4230640 |
| 16 | 4728384 | 36680 | 4765460 |
| 17 | 4748200 | 53016 | 4801628 |
| 18 | 4730520 | 44776 | 4775694 |
| 19 | 4735344 | 44816 | 4780563 |
| 20 | 4744736 | 29504 | 4774649 |
| 21 | 4734432 | 25504 | 4760338 |
| 22 | 4735344 | 40640 | 4776387 |
| 23 | 3696216 | 34840 | 3731475 |
| 24 | 4736600 | 33440 | 4770348 |
| 25 | 4743984 | 30168 | 4774563 |
| 26 | 4751240 | 41248 | 4792902 |
| 27 | 4734528 | 28936 | 4763866 |
| 28 | 4734336 | 22608 | 4757346 |
| 29 | 4745640 | 40584 | 4786634 |

Tabelle 4.7: Messwerte für Rechner ”ingo”

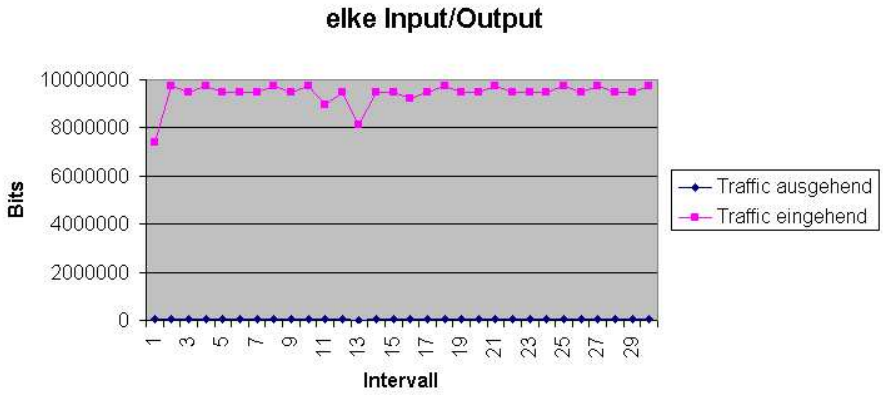



Abbildung 4.26: Datendurchsatz am Port 4 zum Rechner ”elke”

| Intervall | Bits gesendet | Bits empfangen | Bits gesamt |
|---|---|---|---|
| 0 | 4744704 | 49512 | 4794625 |
| 1 | 4993456 | 27528 | 5021405 |
| 2 | 4728384 | 35904 | 4764684 |
| 3 | 5011760 | 52800 | 5064994 |
| 4 | 4728384 | 25032 | 4753812 |
| 5 | 4728384 | 33112 | 4761892 |
| 6 | 4756912 | 49928 | 4807261 |
| 7 | 4993456 | 24848 | 5018725 |
| 8 | 4728384 | 34960 | 4763740 |
| 9 | 5015840 | 67416 | 5083694 |
| 10 | 4728384 | 22976 | 4751756 |
| 11 | 3677632 | 26328 | 3704268 |
| 12 | 4997456 | 34288 | 5032167 |
| 13 | 4740192 | 36384 | 4776982 |
| 14 | 4728384 | 40256 | 4769036 |
| 15 | 5011344 | 60184 | 5071961 |
| 16 | 4728944 | 46040 | 4775381 |
| 17 | 4991072 | 44232 | 5035722 |
| 18 | 4748784 | 71240 | 4820437 |
| 19 | 4730768 | 48496 | 4779663 |
| 20 | 4991072 | 23504 | 5014994 |
| 21 | 4746080 | 44072 | 4790562 |
| 22 | 4729200 | 42640 | 4772237 |
| 23 | 4728384 | 27528 | 4756308 |
| 24 | 5012240 | 55832 | 506850 |
| 25 | 4730768 | 25856 | 4757023 |
| 26 | 4991072 | 25040 | 5016530 |
| 27 | 4749504 | 53696 | 4803612 |
| 28 | 4728384 | 27112 | 4755892 |
| 29 | 3940320 | 30704 | 3971442 |

Tabelle 4.8: Messwerte für Rechner "bernd"

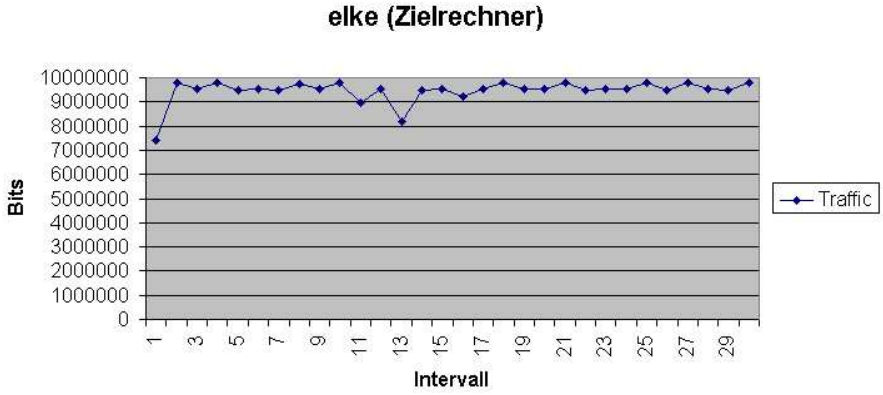



Abbildung 4.27: Gesamtdatendurchsatz am Port 4 zum Rechner "elke"

| Intervall | Bits gesendet | Bits empfangen | Bits gesamt |
|---|---|---|---|
| 0 | 7370920 | 27984 | 7398931 |
| 1 | 9734024 | 37008 | 9771068 |
| 2 | 9492896 | 46416 | 9539353 |
| 3 | 9750160 | 46096 | 9796301 |
| 4 | 9465640 | 31008 | 9496678 |
| 5 | 9488616 | 44992 | 9533648 |
| 6 | 9469160 | 34896 | 9504090 |
| 7 | 9724016 | 30960 | 9755006 |
| 8 | 9491952 | 46416 | 9538409 |
| 9 | 9743704 | 29120 | 9772852 |
| 10 | 8935648 | 27040 | 8962714 |
| 11 | 9488672 | 44992 | 9533704 |
| 12 | 8149224 | 23920 | 8173172 |
| 13 | 9463888 | 29920 | 9493837 |
| 14 | 9497248 | 46416 | 9543700 |
| 15 | 9213048 | 28080 | 9241155 |
| 16 | 9484088 | 28080 | 9512195 |
| 17 | 9765016 | 51648 | 9816711 |
| 18 | 9490344 | 28080 | 9518451 |
| 19 | 9486016 | 29920 | 9515965 |
| 20 | 9738080 | 44480 | 9782600 |
| 21 | 9465752 | 28080 | 9493859 |
| 22 | 9480688 | 28896 | 9509612 |
| 23 | 9483032 | 44992 | 9528064 |
| 24 | 9739000 | 31792 | 9770823 |
| 25 | 9463371 | 29920 | 9493325 |
| 26 | 9743680 | 47456 | 9791178 |
| 27 | 9475264 | 32848 | 9508144 |
| 28 | 9465160 | 28080 | 9493267 |
| 29 | 9758992 | 53520 | 9812559 |

Tabelle 4.9: Messwerte für Rechner ”elke”

### 4.3.2 Verifikation der Messungen mittels SimulaNT

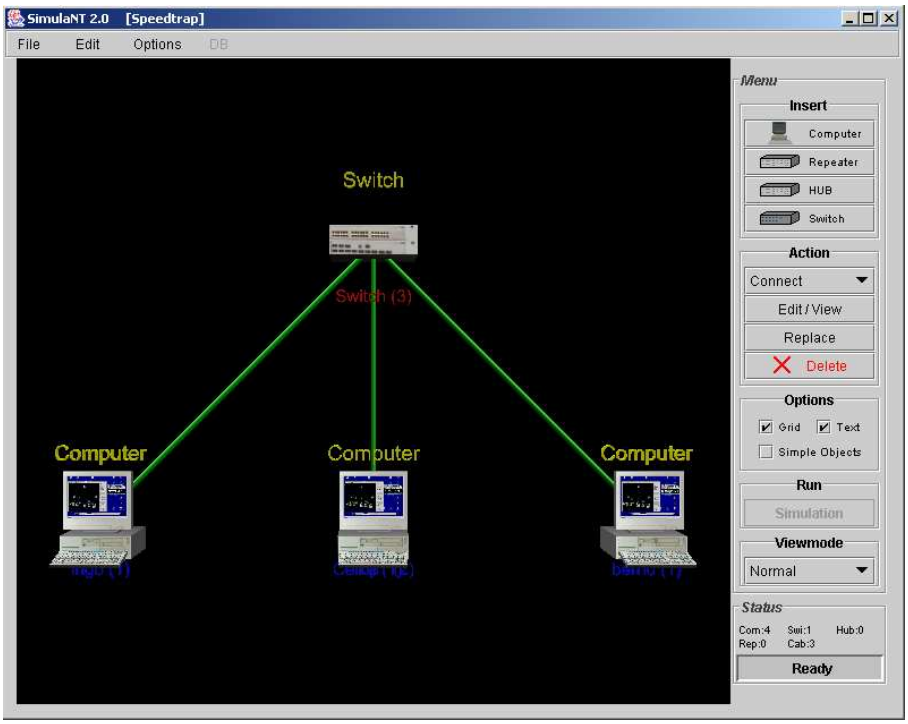



Abbildung 4.28: SimulaNT

Abbildung 4.28 zeigt das modellierte Netz. Es ist minimal und besteht nur aus einem Switch und den drei an der Messung beteiligten Rechnern. Die Lasten werden für die Simulation der drei Messungen genau wie bei den Messungen (beschrieben in Kapitel 4.3.1) gewählt.

| | Ingo |
|---|---|
| Durchsatz | 38,16 MB |
| Goodput | 38,16 MB |
| Badput | 0 |
| Good Frames | 26400 |
| Packetloss | 0 |
| Mean Delay | 121,24 $\mu$s |
| Max Delay | 122,08 $\mu$s |
| Utilization | 1,07% |
| Min. Framesize | 1515,45 Byte |

Tabelle 4.10: Ergebnisse der Simulation der 1. Messung

Die Ergebnisse entsprechen weitestgehend den Erwartungen. Die Tabellen 4.10 bis 4.12 zeigen die Ergebnisse der Simulationen. Relevant ist nur der Wert für „Durchsatz". Dieser Wert, fälschlicherweise von Simulant als Durchsatz bezeichnet, beschreibt die übertragene Datenmenge. In Tabelle 4.13 sind deshalb nochmal die übertragenen Datenmengen der Messungen mit denen der Simulationen und dem theoretischen Wert gegenübergestellt.

Es fällt auf, dass die simulativen Werte immer leicht über den theoretischen liegen. Dies lässt sich leicht damit erklären, das die UDP-Pakete mit den Grössen 32kB und 64kB in viele kleinere Pakete

| | Ingo | Bernd |
|---|---|---|
| Durchsatz | 7,6 MB | 7,6 MB |
| Goodput | 7,6 MB | 7,6 MB |
| Badput | 0 | 0 |
| Good Frames | 5258 | 5236 |
| Packetloss | 0 | 0 |
| Mean Delay | 121,24 $\mu$s | 121,24 $\mu$s |
| Max Delay | 122,08 $\mu$s | 122,08 $\mu$s |
| Utilization | 0,53 % | 0,53 % |
| Min. Framesize | 1515,45 Byte | 1515,45 Byte |

Tabelle 4.11: Ergebnisse der Simulation der 2. Messung

| | Ingo | Bernd |
|---|---|---|
| Durchsatz | 19,02 MB | 19,05 MB |
| Goodput | 19,02 MB | 19,05 MB |
| Badput | 0 | 0 |
| Good Frames | 13178 | 13156 |
| Packetloss | 0 | 0 |
| Mean Delay | 121,24 $\mu$s | 121,24 $\mu$s |
| Max Delay | 122,08 $\mu$s | 122,08 $\mu$s |
| Utilization | 0,53 % | 0,53 % |
| Min. Framesize | 1515,45 Byte | 1515,45 Byte |

Tabelle 4.12: Ergebnisse der Simulation der 3. Messung

| | gemessen | simuliert | berechnet |
|---|---|---|---|
| 1. Messung Ingo | 22,8 MB | 38,16 MB | 37,5 MB |
| 2. Messung Ingo | 6,4 MB | 7,6 MB | 7,5 MB |
| 2. Messung Bernd | 5,8 MB | 7,6 MB | 7,5 MB |
| 3. Messung Ingo | 16,6 MB | 19,02 MB | 18,75 MB |
| 3. Messung Bernd | 17,0 MB | 19,05 MB | 18,75 MB |

Tabelle 4.13: Zusammenfassung des Durchsatz der drei Simulationen

kleiner als 1500B zerlegt wurden. Für jedes der kleinen Pakete kamen noch Header-Informationen dazu, so dass die tatsächlich übertragen Bits mehr waren als die 64kB pro UDP-Paket.

Weiterhin fällt an den gemessenen Werten auf, dass die tatsächlich erzeugte Last kleiner war als die gewünschte Last. Hier ist naheliegend, dass die Rechner (Sun Ultra 1) nicht so schnell waren, wie es für die Java-Maschine nötig gewesen wäre, um die gewünschten Pakete zu erstellen. Man sieht, dass die Lasten aus der ersten Messung (die das zehnfache der beiden anderen Messungen betragen) deutlich schlechter erzeugt wurden. Hier wurden nur knapp 60% der gewünschten Pakete erzeugt. In den beiden weiteren ist die Differenz nicht mehr so hoch, es wurden aber auch niedrigere Anforderungen gestellt. Es wurden im Mittel 86% der Pakete erzeugt.

Abschliessend lässt sich aber sagen, dass die Verifikation erfolgreich war, die Messergebnisse sind ausgefallen wir erwartet bzw. gewünscht.

# Kapitel 5

# Einsatz

Nach dem erfolgreichen Testeinsatz der Anwendung am Lehrstuhl Informatik 4, kann nun der Einsatz bei der Kreispolizeibehörde in Unna erfolgen. Vorbereitend wurde dafür ein Einsatzplan erstellt, der in Kapitel 5.1 erläutert wird. Im Kapitel 5.2 wird dann die Durchführung des Einsatzes beschrieben und eine Auswertung der erlangten Messdaten vorgenommen.

## 5.1 Einsatzplan Unna

**Vorgehensmodell**

Im folgendem wird ein Vorgehensmodell für den Einsatz des Java-basierten Netzwerk-Analysetools beschrieben:

- Ein Gruppenmitglied konfiguriert die Switche in Unna. Dabei sollte die SNMP-Funktionalität auf allen Switchen aktiviert und die Passwörter gesetzt werden.
- Gruppen, bestehend aus 1-2 Personen, installieren Agenten und den Monitor in Unna und anderen Standorten. (Da die Standorte der Switche und die Rechner bekannt sind, muss der Monitor nicht vorher gestartet werden, um den Netzplan zu erkennen.
- Falls Fehler auftreten, sollte der Debug-Modus eingeschaltet und das Problem ausfindig gemacht werden. Je nach Art der Schwierigkeit das Problem beiseitigen zu können, können mehrere Tage dafür draufgehen.
- Mehrfache Messungen sollten durchgeführt werden, um auf Plausibilität zu überprüfen. Dazu könnte auch der Lastgenerator zum Einsatz kommen. Es können die Testdaten verwendet werden, die auch beim Einsatz am LS4 zum Einsatz gekommen sind.
- Falls bis hierhin alles gut verlaufen ist, kann dann mit den eigentlichen Messungen begonnen werden. Es kann dann auf die Probleme des Polizeinetzwerks eingegangen werden. D.h. es gibt Effizienzprobleme bei der Übertragung der NT-Useraccounts, falls sich mehrere Benutzer zur gleichen Zeit an die Windowsdomäne anmelden. Darüber hinaus auch bei den HTML Masken, wo ganze Worddokumente übers Netz übermittelt werden.
- Abschließend sollte aus Sicherheitsgründen SNMP auf allen Switchen wieder deaktiviert werden.

Bei den oben genannten Punkten sollten die (Mess-) Daten gesammelt werden, um sie später zur Beurteilung heranziehen zu können.

**Die Messung**

Die Polizei hat Performance-Schwächen bei der Übertragung der unter Umständen sehr großen User-Domäne-Informationen. Dieser Fall tritt auch nur auf, wenn mehrere Benutzer sich zur gleichen Zeit an die Domäne anmelden. Eine Vermutung ist, dass die Verbindung zum Domäneserver ausgelastet ist. Wenn die Leitung nur 100 MBit zulässt, kann man sich leicht ausrechnen, was passiert, wenn 20 Benutzer eine Domäne-Information von ca. 5 MB zur gleichen Zeit anfordern.

Um diese Problematik genau analysieren zu können, wäre eine Messung speziell dieser Leitung notwendig. Dabei könnte man den Monitor über Nacht laufen lassen, so dass er morgens Aufschluss über den Netzverkehr während der morgigen "Massen"-Einloggphase geben kann. Dies sollte aber nur für eine alternative Angabe gelten, die das Messergebnis unterstützen soll, da es nicht sicher ist, dass auch wirklich viele User sich gleichzeitig anmelden. Besser ist es, gezielt Rechner dafür einzusetzen, um dieses Szenario nachzustellen. Da wir nicht die Polizeibeamten bei ihrer wichtigen Arbeit stören wollen, wäre es sinnvoll, dies während der Mittagspause zu vollziehen, auch um etwaige Störfaktoren auszuschließen, die während der Benutzung der Computer durch die Beamten auftreten können.

Wichtige Informationen, die Aufschluss über den Flaschenhals geben, sind zum einen die Anzahl der gesendeten und empfangenen Bits, als auch die Größe der User-Domäne-Informationen, die sich auf den jeweiligen Rechnern befinden und auch auf den Domäneserver.

Ein weiteres Effizienzproblem bei der Polizei ist, dass sie jeweils eine 128 KBit Leitung zu den beiden Nebenstandorten besitzen, wobei eine Leitung davon, laut einem Beamten, angeblich schneller sein soll. Um dieses genau zu untersuchen, sollten unter anderem die Ports des Switches, an dem der Router der 128 KBit Leitungen hängt, über einen gewissen Zeitraum abgefragt werden. Es müssten dann auch gezielt Datenpakete über diese Leitungen verschickt werden.

Zu guter letzt gibt es auch die Feststellung, dass HTML-basierte Dokumente, übers Netz fließen, wobei die Zeit zwischen Anfrage und erfolgter Übermittelung der Dateien unzumutbar hoch seien. Auch bei dieser Analyse, sollten die Verbindungen kontrolliert werden, während spezielle Dokumente angefordert werden. Dateigrößen und Anzahl der Rechner sollten mitprotokolliert werden.

## 5.2 Messbericht Unna

Dieser Bericht über den Messeinsatz in Unna ist in drei Teile gegliedert. Als erstes wird die Vorbereitung und die Einsatzumgebung beschrieben. In zweiten Teil wird über den Messeinsatz berichtet und im dritten Teil abschließend eine Auswertung der Ergebnisse und eine Schlussfolgerung daraus gegeben.

### 5.2.1 Einsatzvorbereitung

Bei dem Netz in Unna handelt es sich um das IP-Netz 8.3.4.0. Es besitzt zwei Kernkomponenten, um die das Netz aufgebaut ist. Zum einen den sogenannten Kopf-Router, der die Außenstellen an Unna anbindet und den Zugang zum CNPOL-Netz der Polizei in NRW ermöglicht, über das auch ein beschränkter Zugriff auf das Internet möglich ist. Die 128 kBit Leitungen zu den Außenstellen Bergkamen, Schwerte und Werne sind direkt mit dem Router verbunden, die Anbindung nach Kamen wird durch die Hicom Telefonanlage realisiert.

Bei dem Kopf-Router handelt es sich um einen Cisco-3600/ Cisco-RPS Router (IP-Adresse 8.3.4.254). Abbildung 5.1 zeigt das gesamt Netzwerk. Die andere zentrale Komponente ist ein 3COM Switch (Superstack 3300, IP-Adresse 8.3.4.221) an den direkt alle Server und sechs weitere Switche (3COM Superstack 1100, IP-Adressen .222-.226+.228) über 100 MBit CAT-5 Verkabelung angeschlossen sind. Außerdem ist auch der sogenannte „KDVZ-Server" (IP-Adresse 8.3.4.99 zum Unna-Netz und

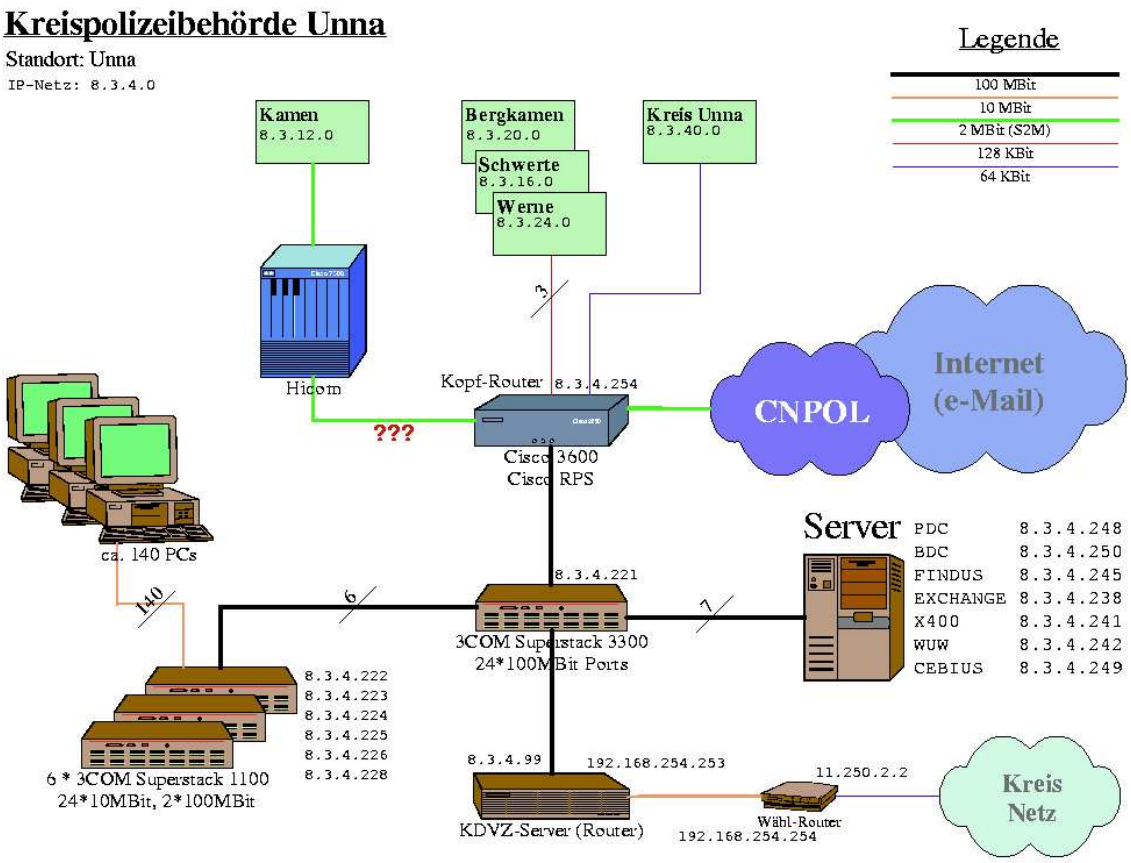



Abbildung 5.1: Netzplan der Kreispolizei Unna

192.168.254.253 zum Wählrouter), der eine Wählverbindung zum Netz der Kreisverwaltung über einen Wählrouter (IP-Adresse 192.168.254.254 zum KDVZ-Server und 11.250.2.2 zum Kreisverwaltungsnetz) realisiert, an diesen Switch angeschlossen.

Die sechs weiteren Switche dienen dem Anschluss der 140 Arbeitsplatz-PCs in den Büros der Kreispolizeibehörde Unna.

Als Besonderheit sei erwähnt, dass die Büros mit Lichtwellenleitern (LWL) verkabelt sind und deshalb die CAT-5 Verkabelung der Switche über Mediakonverter auf LWL umgesetzt wird. Die Server sind direkt über CAT-5 an den zentralen Switch angeschlossen.

Es muss hier bemerkt werden, dass im Gegensatz zum Einsatz im LS-4 Netzwerk der Uni-Dortmund, der Netzplan der Kreispolizeibehörde Unna mit ca. 200 Komponenten (Rechner, Switches, Hubs usw.) wesentlich umfangreicher ist. Daher werden wir hier nicht jede Komponente einzeln erwähnen. Nennenswerter dagegen sind die Rechner, auf denen die Agenten laufen und die Switches, die sie überwachen. Im Netzwerk der Kreispolizeibhörde Unna sind sieben Switches vorhanden und entsprechend sind höchstens sieben Agenten notwendig für die Messung des gesamten Netzwerks. Die Tabelle 5.1 zeigt die Switches und ihre Agenten. Die Agenten sind als Dienste installiert. Die Switches und Agenten wurden als IP-Addresse angegeben wobei alle IP-Addressen mit dem Netz 8.3.4.* beginnen; daher ist in dieser Tabelle nur die Host-Adresse angegeben.

Die Rechner auf denen die Agenten installiert wurden, wurden so gewählt, dass jeder Agent direkt an dem Switch hängt, den er überwacht.

Der Einsatz in Unna hat insgesamt fünf Tage von jeweils ca. 7 Stunden in Anspruch genommen.

| Switch | Agent |
|---|---|
| 221 | 63 |
| 222 | 112 |
| 223 | 87 |
| 224 | 37 |
| 225 | 77 |
| 226 | 1 |
| 228 | 125 |

Tabelle 5.1: Agent/Switch Zuordnung

Als erstes standen die erfolgreiche automatische Erkennung des Netzes und die Installation von Agenten im Vordergrund. Dabei sind ein paar unvorhersehbare Probleme aufgetreten, die teilweise vor Ort und teilweise nicht sofort beseitigt werden konnten. Für die Fehlerbehebung wurden zwei Tage in Anspruch genommen. Nachdem die automatische Netzplanerkennung erfolgreich funktionierte, konnte mit dem ersten Messversuch begonnen werden, der jedoch fehl schlug, da am Monitor einige Fehler beseitigt werden mussten. Die Korrekturen und Tests die damit verbunden waren haben zwei weitere Tage gedauert. Damit waren dann alle bekannten Probleme aus dem Weg geräumt. Am fünften und letzten Tag konnte die Messungen dann einwandfrei durchgeführt werden.

### 5.2.2 Messeinsatz

Am Mittwoch den 23.07.2003 wurden die Tests in Unna durchgeführt. Es sind insgesamt sechs Messungen mit verschiedenen Zielen durchgeführt worden. Alle Messungen bis auf die letzte (Anmeldung) die mit GUI-Intervall 30 Sekunden wurde mit eine GUI-Interval von 10 Sekunden durchgeführt. Abbildung 5.2 zeigt Speedtrap während eine Messung.

**Allgemeine Funktionalität von Speedtrap :** Als erstes wurde eine halbstündige Messung mit 7 Agenten (da es sieben Switches gibt) durchgeführt. Die Ziele dieser Messung sind zum einen, der Test der allgemeinen Funktionalität von *Speedtrap* und zweitens die allgemeine Messung um einen Überblick über die Aktivitäten im produktiven Polizei-Netzwerk zu bekommen. Die Messung ist erfolgreich zwischen 11:31 Uhr und 12:01 Uhr gelaufen.

**Messung der Leitung nach Schwerte:** Die Motivation für diese Messung liegt darin, dass das Verschieben von Daten zwichen Unna und Schwerte meistens zu lange dauert. Uns ist auch bekannt das die Leitung bis zum Router eine 100Mbit Leitung ist. Vom Router bis nach Schwerte hat die Leitung jedoch nur noch eine Kapazität von 128kBit. Unsere Vermutung ist daher, dass wegen der 128kBit der Datenverkehr ab dem Router zu stark verlangsamt wird. Unser Ziel mit dieser Messung ist es, diese Vermutung zu untermauern. Dazu wurde mit einem Agenten direkt am Hauptswitch 221 gemessen. Dabei interessierte uns hauptsächlich der Port 24, da dies der Ausgangsport zum Router ist. Eine 10 minütige Messung wurde durchgeführt. Während dieser Messung wurde ein Mitarbeiter der Polizeibehörde aufgefordert, eine Datei von Unna nach Schwerte zu kopieren.

**Datenbank Server:** Die Mitarbeiter beschweren sich, dass es bei Abrufen oder Recherchen in Datenbanken zu lange dauert, bis die Informationen zurückgeliefert werden. Die Vermutung wäre, dass die Datenmengen zu groß sind und vermutlich die Auslastung der Leitungen die ganze Zeit voll ausgelastet sein müssten. Das Ziel dieser Messung ist es, diese Theorie zu unterstützen oder zu verwerfen. Agent 63 kommt dabei zum Einsatz, da der Datenbankserver mit IP-Adresse 250 direkt mit dem Switch 221 auf Port 19 verbunden ist (siehe Tabelle 5.1). Ein Mitarbeiter der Polizei wurde aufgefordert, Anfragen an die Datenbank während der

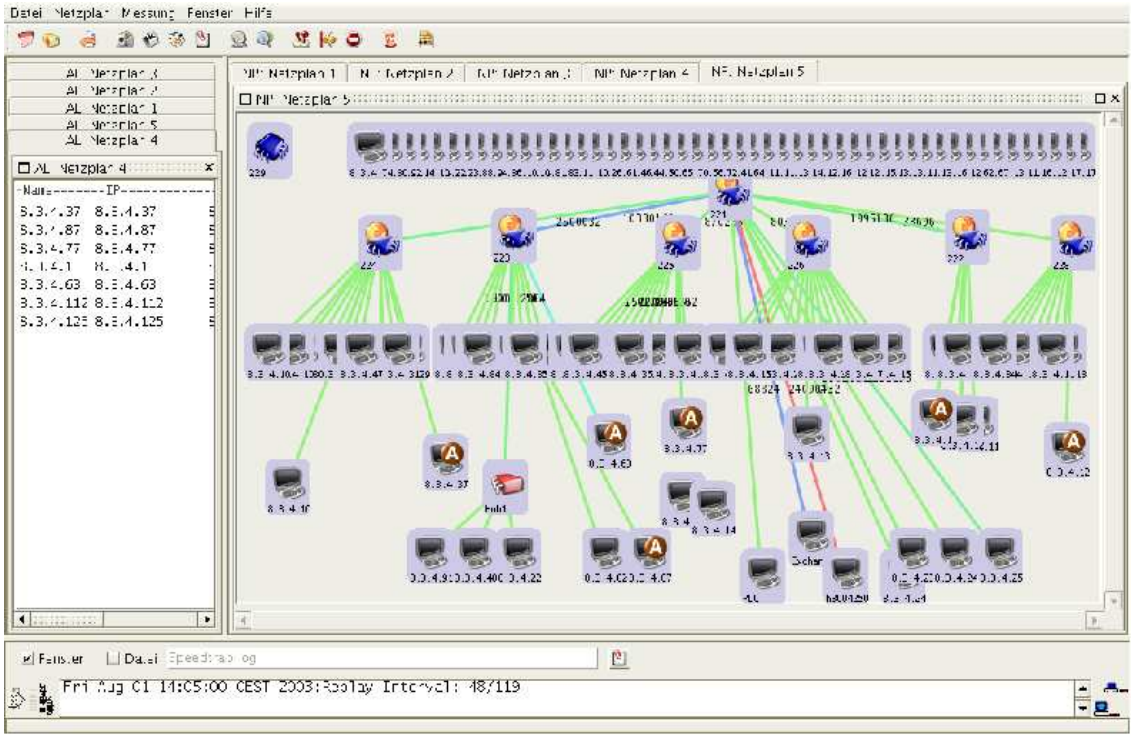


Abbildung 5.2: Speedtrap: Netzplan Unna

Messung zu tätigen. Diese geschah vom Rechner mit der IP-Adresse 65 aus. Die Messung wurde über einen Zeitraum vom 25 Minuten durchgeführt.

**Mail Server:** Der Mailserver wurde genauer unter die Lupe genommen. Ein Mitarbeiter wurde aufgefordert, eine möglichst große e-Mail an sich selbst zu schicken, um zu erfahren, ob und wie der Mailserver belastet wird. Es wurde eine 21 minütige Messung durchgeführt. Dabei wurde ein Agent eingesetzt, der direkt am Switch 221 misst, wo der Mailserver auf Port 18 angeschlossen ist.

**Ausloggen der Mitarbeiter:** Die Mitarbeiter der Kreispolizeibehörde Unna haben einen zentralen Domain-Server, auf dem die Profile der Nutzer gespeichert werden. Dabei wird berichtet, dass es während des Geschäftschlusses zwischen 15:45 Uhr und 16:15 Uhr zu erheblichen Verzögerungen im Netz kommt. Die Vermutung ist, dass manche Mitarbeiter Profile, die bis zu 5MByte groß sein können, haben. Es kommt zu erheblichem Datenverkehr, wenn sie alle auf einmal ausloggen und die Profile wieder auf den Server verschoben werden müssen. Das Ziel hierbei ist es, die Leitung zu diesem Server in diesem Zeitraum zu beobachten, um die Auslastung festzustellen. Der Domain-Server hängt am Switch 221 auf Port 5. Daher ist diesee Port von besonderem Interesse bei dieser Messung.

**Einloggen der Mitarbeiter:** Die Motivation für diese Messung ist analog zu der Motivation beim Ausloggen der Mitarbeiter, nur dass es eben um den umgekehrten Fall des einloggen geht. Man erwartet eigentlich keinen Unterschied im Verhalten, im Vergleich zum Ausloggen. Der Völlständigkeit halber, wurde die Messung jedoch für wichtig erachtet. Zur Erfassung der Daten bei der Anmeldung, wurde eine Messung gegen 16.00 Uhr am 23.07.2003 für die Zeit zwischen 07:15 Uhr und 08:15 Uhr am 24.07.2003 gestartet, da diese Zeit die Hochbetriebzeit darstellt, zu der die Mitarbeiter ihre Tagesarbeit aufnehmen und sich ins System einloggen.

### 5.2.3 Auswertung und Fazit

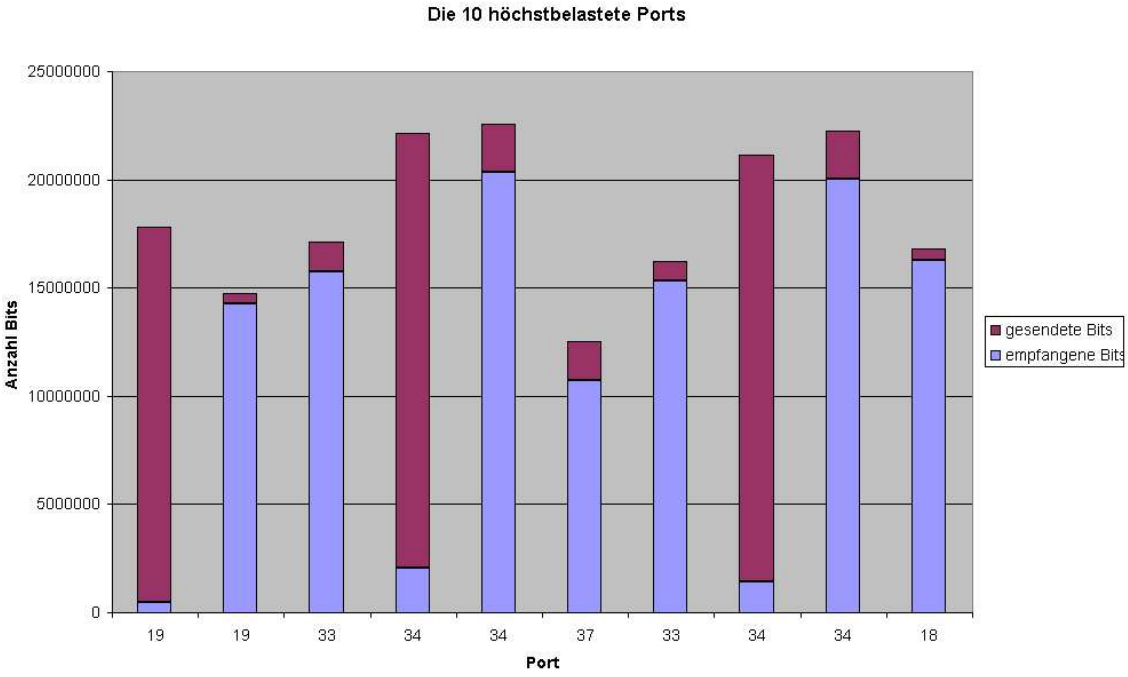



Abbildung 5.3: Die 10 meist belasteten Leitungen

**Auswertung der allgemeinen Funktionalität von Speedtrap** Bei der ersten Betrachtung der Messdaten ist uns sofort aufgefallen dass die gemessenen Datenmengen relativ klein sind. Die Werte schienen alle weit unter den Kapazitäten der Leitungen von 100Mbit zu liegen. Als erstes wurden die Messdaten gefiltert und alle Ports, die nicht besetzt waren, wurden ausgeblendet. Danach wurde die gesamte Anzahl von Bits auf jedem Port berechnet. Diese setzte sich aus den *gesendeten Bits* und den *empfangenen Bits* für jeden Port zusammen. Dadurch war auch der gesamte Durchsatz in jedem Port pro Intervall ersichtlich. Die Informationen, die wir aus diesem Schritt gewinnen konnten, erhärtet weiter den Verdacht, dass die Anzahl der Bits die durch die Leitungen fließen relativ gering sind. Trotzdem konnte man immer noch keine Aussage angesichts der Datenmenge treffen. Um unseren Verdacht zu untermauern, wurden letztendlich die 10 meist belasteten Leitungen hervorgehoben, wie in Abbildung 5.3 zu sehen ist. Wie man sehen kann, liegt die höchste Belastung bei 22.592.304 Bits pro Intervall(10 Sekunden).

Dieser Wert entspricht im „Worst-Case" Fall eine Auslastung von ungefähr 25%. Dieser Wert ist realistisch und kann in keinerlei Weise eine Verzögerung in einem Netzwerk hervorrufen. Aus diesem doch etwas überraschenden Ergebnis lässt sich schon vermuten, dass die Verzögerung im Netzwerk doch eine andere Ursache haben könnte als die Belastung der Leitungen im Netzwerk. Ob die anderen Messungen dieses Erkenntnis bestätigen, sehen wir unten bei der Auswertung der anderen Messungen.

**Auswertung der Messung der Leitung nach Schwerte** Die Auswerung dieser Messung verläuft analog zur der Auswertung oben. Jedoch da lediglich Port 24 von Interesse ist, wurde nach der Berechnung der Gesamtbits die Messdaten von allen übrigen Ports ausgelassen. Dadurch entstand eine Tabelle mit so vielen Zeilen, wie die Messung Intervalle in der Messung gab. Eine Darstellung des Messergebnisses als Diagramm verrät uns, dass der Trend

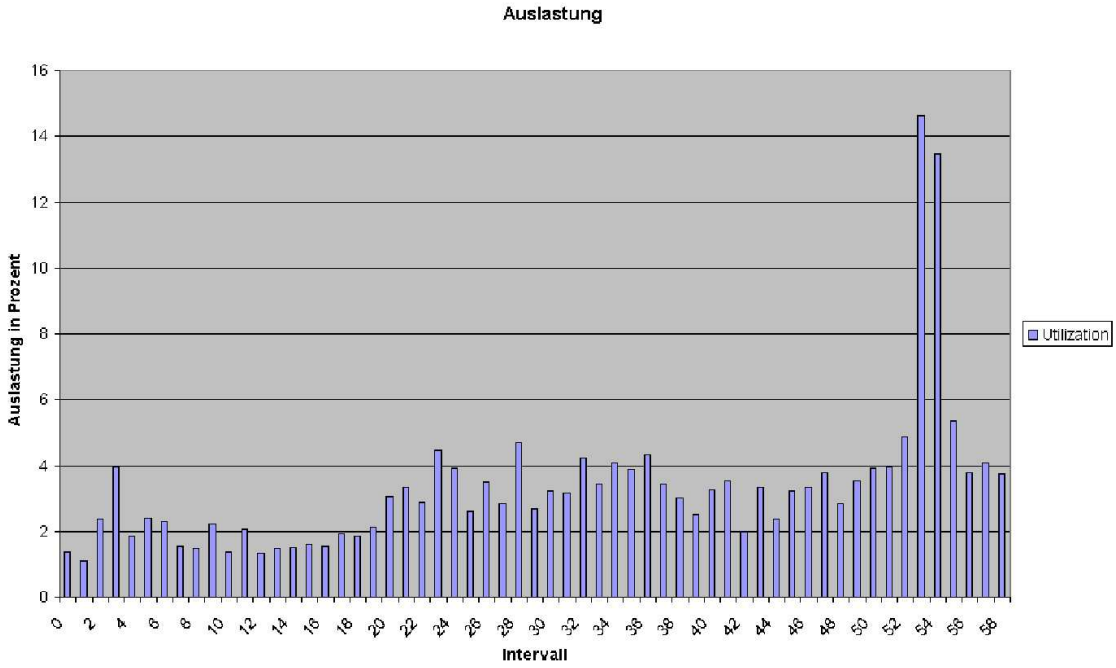



Abbildung 5.4: Auslastung der Leitung nach Schwerte

weiter geht. Die grösste Belastung der Leitung liegt bei diesem Port (Port 24) bei 1.4615.624 Bits. Abbildung 5.4 zeigt das Ergebnis. Mit einer einmaligen Auslastung von ca 14.6% kann von Überlastung keine Rede sein. *Anmerkung:* Die Leitung zum Router bedient ausser der Leitung nach Schwerte auch die Verbindungen nach Kamen, Bergkamen und ins CNPOL-Netz. Daher ist die Datenmenge, die auf der Leitung gemessen wurde nicht ausschließlich zu Schwerte zuzuordnen. Außerdem könnte die 128kBit Leitung vom Router nach Schwerte noch einen Engpaß darstellen. Diese lässt sich aber nicht durch Speedtrap oder die Messdaten die aus der aktuellen Konfiguration erlangt wurden nicht eindeutig nachprüfen.

**Datenbank Server** Nach der Auswertung der Messdaten des Datenbankserver bestätigt den Trend der vorherigen Auswertung. Das Vorgehen ist bereits aus den vorigen Messungen bekannt. Der für diese Messung wichtige Port 19 wurde herausgefiltert und als Kurve in Abbildung 5.5 dargestellt. Es fällt wieder auf, dass sich die Anzahl der Gesamtbits weiter im Rahmen hält.

**Mail Server** Über den Mailserver wurde während der Messung eine ungewöhnlich große e-Mail versendet. Diese hat sich eindeutig im Messergebnis wiedergespiegelt. Es ist aber wegen einer Begrenzung der e-Mail Größen nicht möglich, während des normalem Betriebs so eine große e-Mail zu versenden. Trotzdem war diese aussergewöhnlich große Email nicht in der Lage die ganze Kapazität der Leitung zu erschöpfen. Abbildung 5.6 zeigt dass die Leitung weit davon entfernt ist, überlastet zu sein.

**Ausloggen der Mitarbeiter** Bei dieser Messung ist die Erwartung, dass mindestens 80% der Mitabeiter sich innerhalb dieses Zeitrahmens, wie gewöhnlich vom System abmelden und dass die Auslastung oder Überlastung zu spüren sein wird. Die Auswertung dieser Messung ist daher sehr wichtig um die Frage zu beantworten, ob die Verzögerungen tatsächlich auf die Auslastung zurückzuführen sind.

Das Ergebnis zeigt in Abbildung 5.7 einen Anstieg der Auslastung, jedoch liegen sie alle unter

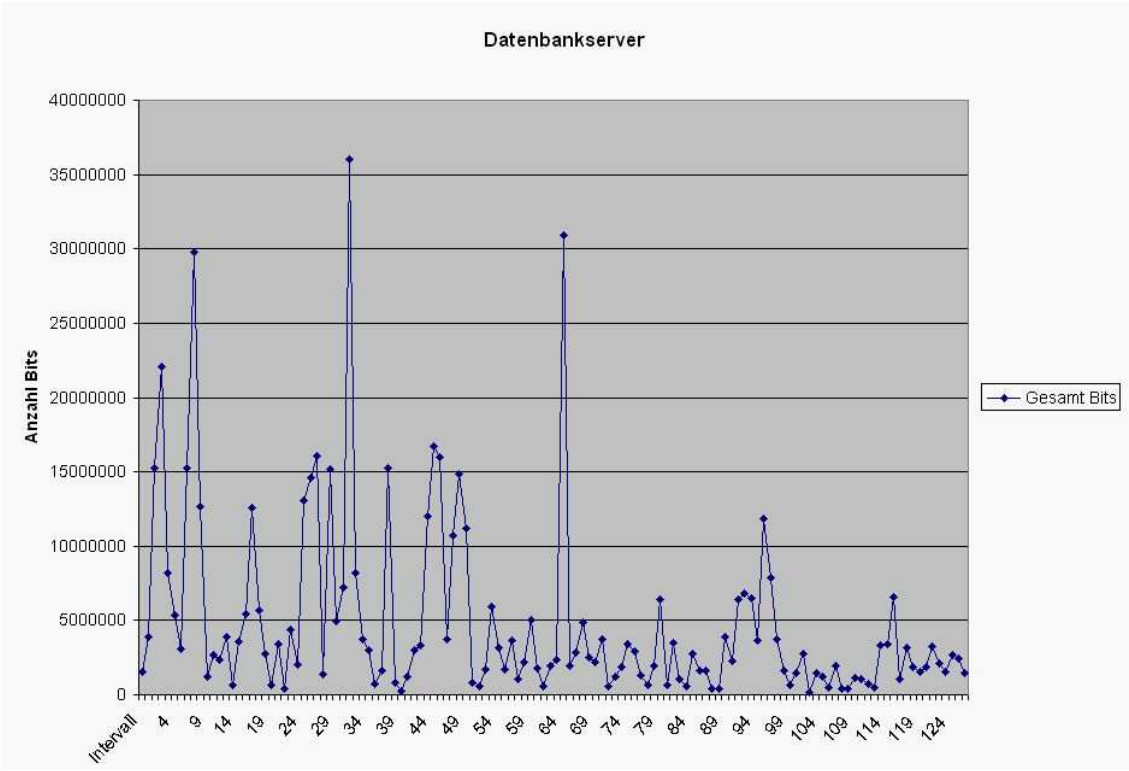



Abbildung 5.5: Durchsatz pro Intervall bei der Datenbank

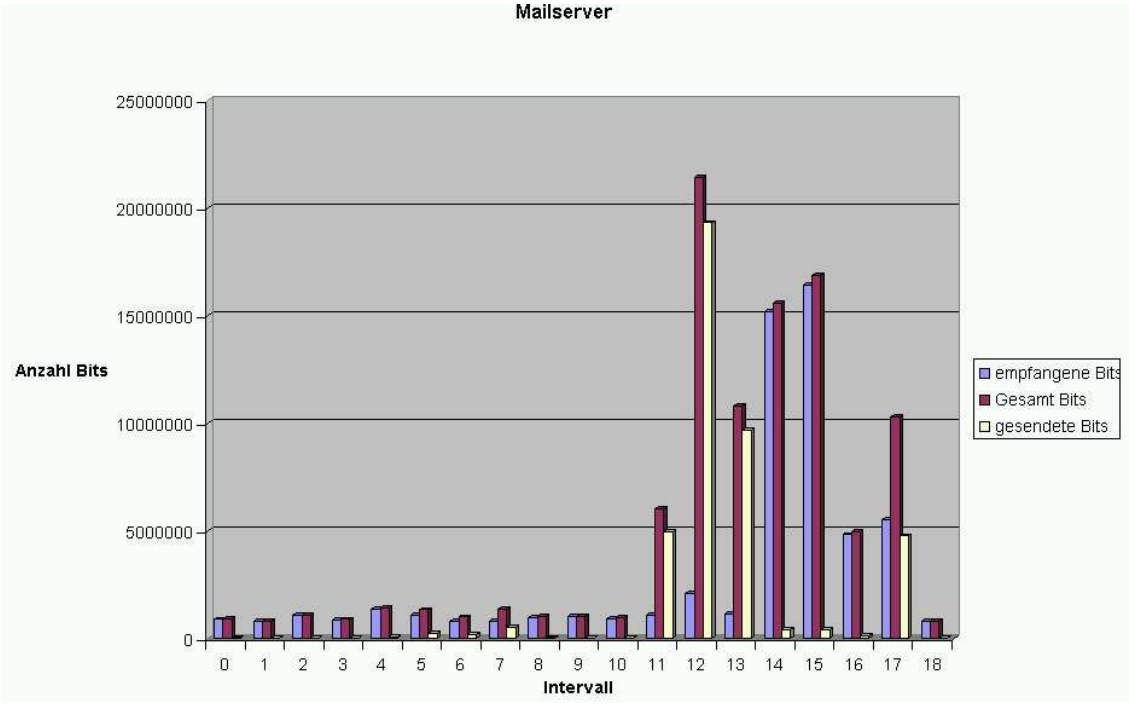



Abbildung 5.6: Durchsatz pro Intervall beim Mailserver

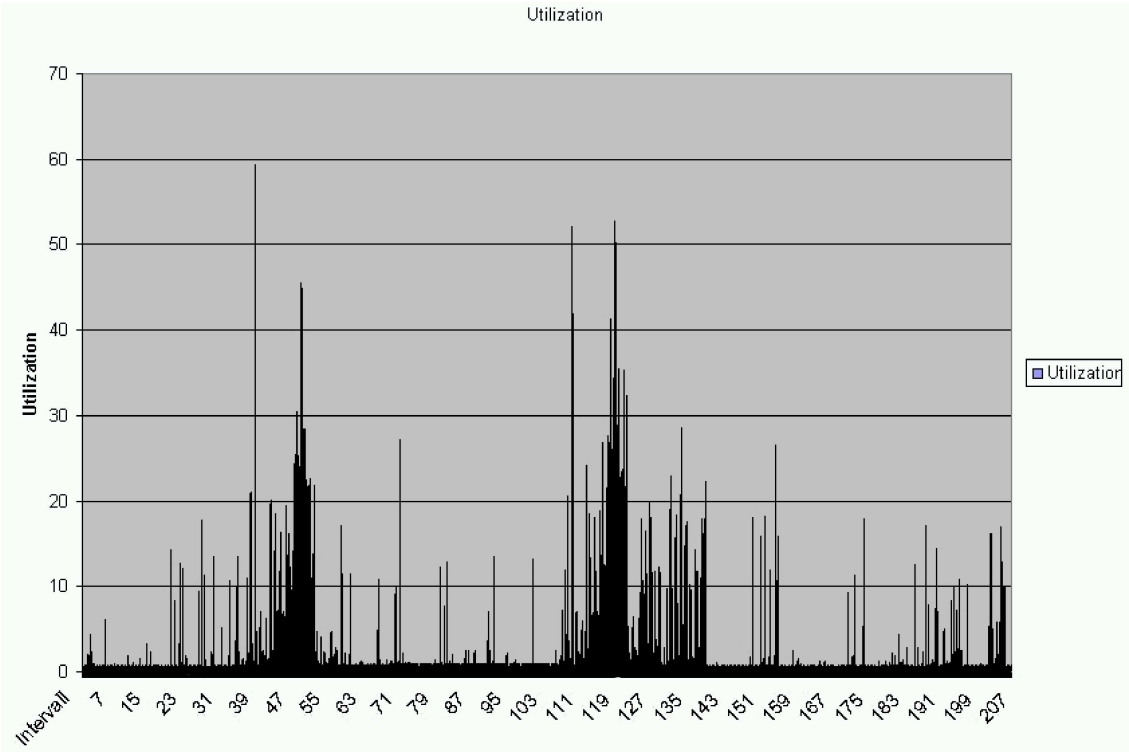



Abbildung 5.7: Auslastung beim Ausloggen der Mitarbeiter

der maximalen Kapazität von 100MBit. Die Durchschnittauslastung, die auf der Leitung zum Domain-Server gemessen wurde, liegt gerade bei 21% der maximalen Kapazität.

Bei der Betrachtung des gesamten Netzes sieht man eine erhebliche Belastung zwischen den Switchen 221 und 223 (Abbildung 5.8. Aus den höchsten Gesamtbits von 54Mbits/Intervall ergibt sich gerade mal eine Auslastung von 5,4%). Das Ergebnis hat aber die bisherigen Erkenntnisse insgesamt gefestigt, dass die Leitungen zu keiner Zeit überlastet waren. Die Auslastungen der Verbindungen zwischen dem Hauptswitch 221 und den anderen Switchen sind auch nicht so hoch.

**Einloggen der Mitarbeiter** Die Anmeldung zu Dienstbeginn ist einigermaßen analog zum Abmelden bei Dienstschluß. Daher wird ein Ergebnis ähnlich dem beim Ausloggen erwartet. Die Auswertung in Abbildung 5.9 hat diese Erwartung auch bestätigt. Die Daten sind Tabellerisch von Speedtrap in Abbildung 5.10 dargestellt. Dabei ist in die Spalte Ütilization"(Auslastung) die höchste Auslastung von 54,3% an der BDC(IP-Addresse: 8.3.4.250) zu erkennen

Dass die Auslastung zu Dienstbeginn der meisten Zeit sehr gering war, hat noch eine andere Dimension in der Suche nach der Ursache vorgebracht, nachdem es sicher ist, dass die Leitungen nicht überlastet sind. Obwohl die Verzögerungen sich sowohl zu Dienstbeginn als auch beim Dienstschluss bemerkbar macht, könnte man vermuten, dass der Server wahrscheinlich nicht in der Lage ist, auf soviele Anfragen auf einmal zu antworten. Auch sind die Auslastungen im Netz in diesen zwei Fällen unterschiedlich. Dies führt und jetzt zu der Annahme, dass der Server überlastet sein könnte.

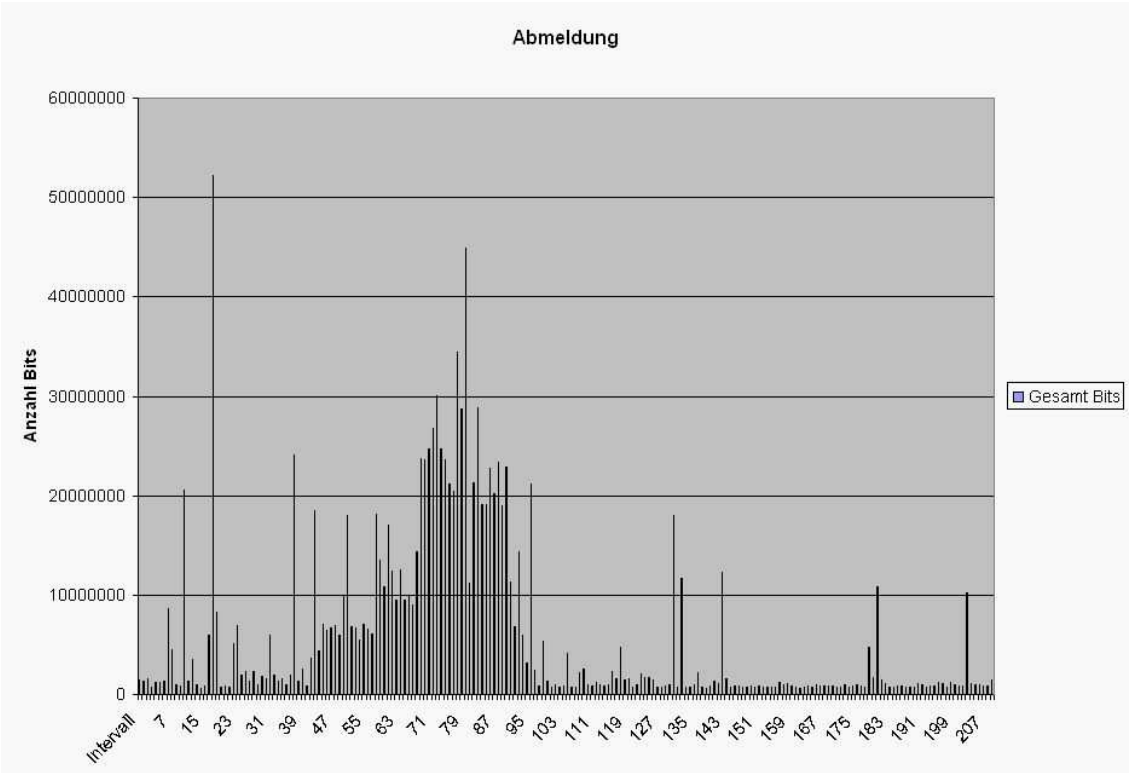



Abbildung 5.8: Durchsatz pro Intervall der Verbindung zwischen Switche 221 + 223

**Weitere Erkenntnisse aus der Auswertung der gemessenen Daten**

**Die Verbindungsstrecken zwischen den Switchen** Als besonders belastet hat sich die Verbindung zwischen den Switchen 221/223 und 221/224 herausgestellt. Insbesondere die Leitung zwischen 221/223 hat eine Auslastung die bis zu 5,4% liegt. Bei der Verbindung von Switch 221 mit Switch 224 haben wir recht unterschiedliche Werte gemessen. Während bei der abendlichen Messung eine für Switchverbindungen durchschnittliche Last vorlag, waren bei der Anmeldung Durchschnittswerte von 3,3% Auslastung. Die dritte Verbindung zwischen Switchen, die sich leicht abhebt ist die Verbingung 221/222. Alle Werte liegen daher weit unter 30% Auslastung die die Hersteller als Grenze empfehlen bevor eine Netzwerkverbindungserweiterung nachgedacht wird.

**221/PrimaryDomainController(248)** Der PDC ist bei unseren Messungen vom Netzwerkverkehr extrem wenig belastet worden. Dies ist für uns recht überraschend, da wir die Verzögerung bei der Anmeldung auf eine Überlastung des Netzes zurückgeführt haben. Die gemessene Auslastung bei der Anmeldung beträgt machmal 1%, ist damit absolut im Toleranzbereich. Bei der Abmeldung ist auch eine Auslastung von 2% keine Seltenheit. Lange Wartezeiten haben also nichts mit einer Überlastung dieser Verbindung zu tun.

**Switch(221)/BackupDomainController(250)** Auffallend ist, das der BDC wesentlich mehr Last beim Abmeldevorgang trägt, als der PDC. Die Daten bei der Anmeldung liegen bei einer Auslastung von 5,4%. Bei der abendlichen Vorfeierabendmessung, bei der der Abmeldevorgang gemessen wurde, erreicht der BDC eine durchschnittliche Utilisation von 4,8% . Die Begründung liegt darin, das die BDCs aufgrund der geringeren Belastung oft schneller reagieren als die PDCs, die einen höheren Verwaltungsauffand haben.

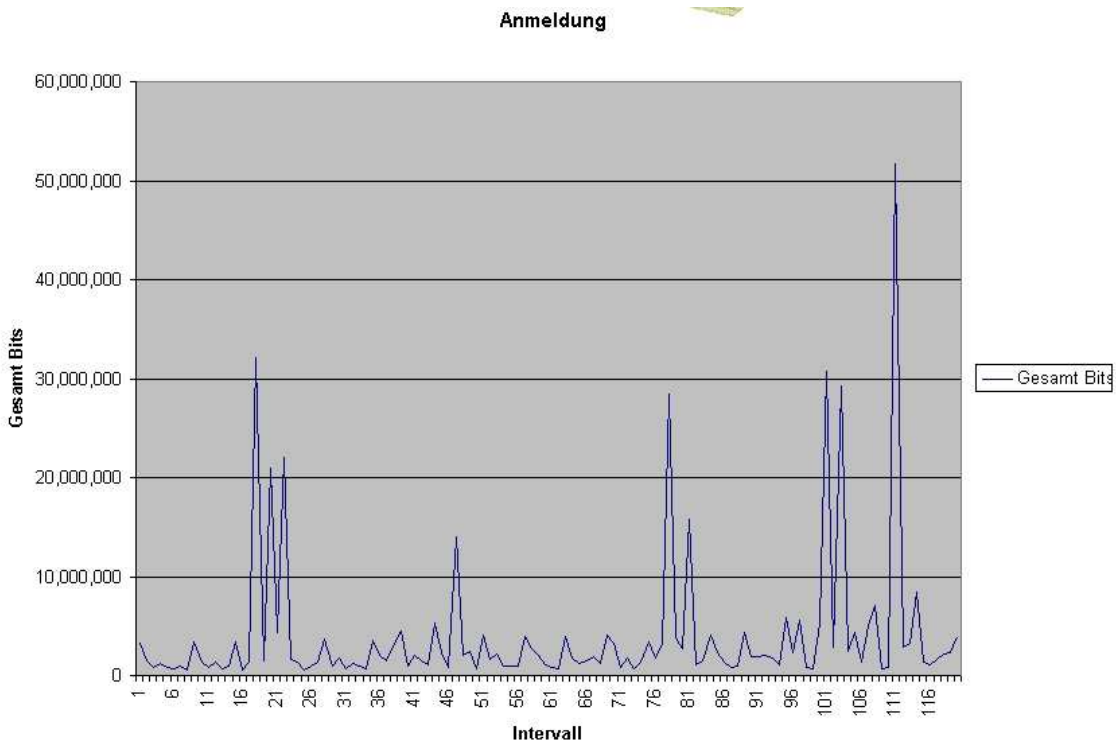



Abbildung 5.9: Durchsatz pro Intervall bei der Anmeldung der Mitarbeiter

**Switch(221)/Exchange(238)** Der Exchange Server ist der am stärksten frequentierte Server ausser den BDCs und PDC. Die Auslastung liegt durchschnittlich zwischen 2,4% und 4,5% je nach Messung. Den Grund hierfür finden wir wohl in den gerne mal verschickten Powerpoint und höchstwahrscheinlich anderen Emails, die große Datenmengen transportieren.

**Switch(221)/andere Server (249, 242, 241, 245)** Der Cebius Server (249), der WUW Server (242) und der Findus Server(245) erzeugen in der morgentlichen Messung wenig Verkehr. Der X400 Server(241) taucht in dieser Messung gar nicht auf. In denen er zu finden ist, ist auch hier keine grosse Last festzustellen. In den restlichen Messungen sind auch nur Werte der Auslastung zu finden, die unter der 8% Grenze liegen. Das deckt sich mit unseren Berichten der Polizei, die über Antwortzeiten dieser Server nicht geklagt haben.

**Client 63** Client 63 ist der Spitzenreiter unter den Clients. Hervorzuheben ist, das ein Agent auf diesem Rechner installiert war. Allerdings haben die anderen Agentenrechner wenig Auslastung, teilweise weniger als 1%. Das bestätigt die Arbeit unserer Gruppe, die stets darauf bedacht war, durch eigenen Datentransfer die Messung so wenig wie möglich zu beeingrächtigen.

**Sonstiges** Ein weiteres Fehlerpotential, das bisher nicht zur Sprache kam sind Überlastungen des Switches selbst, was sich in Form von internen Kollisionen bemerkbar macht und fehlerhafte Kabel, Brücken auf Glassfaser und Switche. Hier konnten wir aber fast keine Fehler feststellen. In allen Messungen wurde fast kein einziges fehlerhaftes Paket gezählt, ebensowenig wie Kollisionen in den Switchen. Lediglich Switch 221 an Port 23 zu Rechner 249 hat bisweilen Fehlerhafte Pakete.

Insgesamt muss festgestellt werden, das die Belastung des Netzwerks in der Zeit der Anmeldung nicht wesentlich höher ist als bei den anderen Messungen, die getätigt wurden.

Parameter der gegenwärtigen Messung

| IPAdresse | Utilisation | |
|---|---|---|
| 8.3.4.91 | 0.3663918 | 228597992 |
| 8.3.4.40 | 0.3663918 | 228597992 |
| 8.3.4.227 | 0.3663918 | 228597992 |
| 8.3.4.175 | 0.025119714 | 12772816 |
| 8.3.4.120 | 0.16900928 | 64944960 |
| 8.3.4.84 | 0.014718419 | 5547624 |
| 8.3.4.140 | 0.014624888 | 5347408 |
| 8.3.4.63 | 0.25552353 | 13180139 |
| 8.3.4.42 | 0.015382801 | 5854976 |
| 8.3.4.85 | 0.1291809 | 50093976 |
| 8.3.4.82 | 0.08572225 | 38552392 |
| 8.3.4.87 | 0.08572225 | 38552392 |
| 8.3.4.221/8.3.4.223 | 0.3241731 | 1833473552 |
| 8.3.4.242 | 0.013199152 | 89127248 |
| 8.3.4.38 | 0.16342811 | 66728936 |
| 8.3.4.45 | 0.10614299 | 48067176 |
| 8.3.4.35 | 0.15325657 | 62640288 |
| 8.3.4.169 | 0.01607787 | 6376048 |
| 8.3.4.159 | 0.019386308 | 7737008 |
| 8.3.4.36 | 0.025726665 | 13168776 |
| 8.3.4.173 | 0.06542109 | 34670832 |
| 8.3.4.174 | 0.014542827 | 5311024 |
| 8.3.4.77 | 0.22990817 | 149964872 |
| 8.3.4.241 | 0.021502713 | 9369968 |
| 8.3.4.99 | 0.01680689 | 8051056 |
| 8.3.4.221/8.3.4.225 | 0.095881425 | 481847792 |
| 8.3.4.226/8.3.4.221 | 0.10283301 | 430760504 |
| 8.3.4.222/8.3.4.221 | 0.09587462 | 424413624 |
| 8.3.4.223/8.3.4.221 | 0.32615405 | 1839344368 |
| 8.3.4.224/8.3.4.221 | 0.22000757 | 1038967272 |
| 8.3.4.225/8.3.4.221 | 0.09588201 | 481834896 |
| 8.3.4.248/PDC | 0.010454313 | 55702072 |
| 8.3.4.228/8.3.4.221 | 0.09721235 | 413327368 |
| 8.3.4.238/Exchange | 0.23975578 | 936572664 |
| 8.3.4.250/BDC | 0.5437746 | 2503453781 |
| 8.3.4.245 | 0.010597204 | 75852648 |
| 8.3.4.237 | 0.11872293 | 660618336 |
| 8.3.4.249 | 0.004129131 | 26191944 |
| 8.3.4.254 | 0.06676776 | 395976832 |

Abbildung 5.10: Auslastung bei der Anmeldung, Screenshot „Speedtrap"

### Vorschläge zur Performance-Steigerung des Netzwerks der KPB Unna

Der Einschätzung der meisten Hersteller zufolge, sollte darüber nachgedacht werden eine Netzwerkverbindung zu erweitern, wenn die Auslastung über 30% ist. Dringend empfohlen wird eine Lösung wenn die Auslastung bis zu 50% beträgt. Ein Wert der Auslastung über 50% führt zu einer Überlastung die sich unmittelbar auf die Nutzung des Netzwerks bzw. des Segments aller Beteiligter auswirkt. Sämtlich Angaben zur Auslastung sind auch Durchschnittswerte, das heißt, wenn der Wert der Auslastung bei 50% liegt, sind immer auch Stoßzeiten dabei, die bis zu 90% Utilisation gehen.

Wenn man sich an dieser Aussage orientiert und die gemessenen Werte (Abb. 5.10) berücksichtigt, muss man sich keine Gedanken über die Netzwerkverbindung machen. Es is naheliegend, dass der PDC und der BDC und andere Server nicht mit dem hohen Maß an Anfragen umgehen können. Der BDC könnte auf folgende Art und Weise entlastet werden. Zum einen könnte der PDC durch einen stärkeren Server ersetzt werden. Dadurch könnten auch auf dem PDC Anmeldevorgänge ausgeführt werden und der BDC wird entlastet. Diese Variante setzt allerdings den Erwerb teurer Hardware und einen nicht unerheblichen administrativen Auffwand voraus. Leichter wäre es, einen weiteren BDC in das Netz zu hängen. Die Anforderung an diese Maschine wären auch nicht sehr hoch, daher könnte man vielleicht von alten Beständen einen Rechner wählen. Der Exchange Server könnte ebenfalls nahe an der Kapazitätsgrenze sein. Eine Möglichkeit, die Bandbreite effektiver zu nutzen ist zu schulen, wie sinnvoll mit den begrenzten Ressourcen umgegangen werden kann. Statt große Datenmengen zu verschicken, die möglicherweise sogar an mehrere Empfänger geht kann beispielsweise einfach nur ein Link auf eine Freigabe geschickt werden. Das spart bei einer

1MB großen Powerpoint, die an 10 Benutzer geschickt nicht nur 10MB Netzlast, sondern auch Platz auf den Festplatten und den Exchange Postfächern der Nutzer.

Zuletzt kann noch darüber nachgedacht werden, ob Port 23 auf Switch 8.3.4.221 auf einen anderen Port des selben Switch umgepatcht wird. Von unseren relativ kurzen Messungen kann aber nicht genau betimmt werden, ob die fehlerhaften Pakete ein zufälliger Fehler war, oder ob er ständig auftaucht.

# Kapitel 6

# Handbuch

Es existieren zwei Handbücher zu Speedtrap, die sich an unterschiedliche Benutzerkreise richten. Das *Benutzungshandbuch* (Kapitel 6.1) richtet sich an die Anwender von Speedtrap, also die Personen, die Messungen mit Speedtrap durchführen wollen. Speedtrap gliedert sich in die zwei Komponenten *Monitor* und *Agent*. Der *Agent* ist die Komponente, die im Hintergrund, verteilt auf viele Rechner arbeitet und dabei die Messdaten sammelt. Der *Monitor* stellt die Benutzerschnittstelle dar, mit der der Benutzer Messungen konfigurieren und auswerten kann. Mit der Handhabung dieser Komponente beschäftigt sich hauptsächlich das *Benutzungshandbuch*. Es erklärt die Funktionen des Monitors und gibt eine Einweisung, wie Messungen mit Speedtrap durchgeführt werden. Das *Administrationshandbuch* (Kapitel 6.2) beschäftigt sich mit der Installation der beiden Speedtrap-Komponenten und der Kommunikation dieser beiden untereinander. Es richtet sich daher hauptsächlich an Administratoren, die die Software installieren und näher verstehen möchten.

## 6.1 Benutzungshandbuch

Das folgende Kapitel stellt das Benutzungshandbuch zu dem von der PG 422 entwickelten Softwaretool Speedtrap dar. Es gibt einen kurzen Abriss über Sinn und Zweck des Programms, weist grob in Funktionsweise und Benutzungsziele des Programms ein und erklärt schließlich ausführlich den Umgang im täglichen Gebrauch.

Auf die weitere Motivation zur Erstellung von Speedtrap im Rahmen der Projektarbeit und die Entwicklungsgeschichte mit den einzelnen Konzeptionsphasen wird in diesem Handbuch nicht weiter eingegangen.

Für ein besseres Verständnis der Materie und den evtl. autretenden Fragen, warum bestimmte Features auf eine bestimmte Art und Weise enstanden oder eben nicht enstanden sind, wird daher auf den Zwischenbericht [P422ZB], sowie die Kapitel 1 bis 5 dieses Endberichts der PG 422 verwiesen.

Weiterhin wird darauf verwiesen, daß Speedtrap inklusive der Programmcode Dokumentationen komplett im Quellcode verfügbar ist und die Internals vom interessierten Anwender so eingesehen werden können.

### 6.1.1 Einleitung

#### Idee und Ziel von Speedtrap

Heutige Rechnernetzwerke wachsen schnell mit den an sie gestellten Anforderungen. Um die Übersicht über die Auslastung und den zukünftigen Leistungsbedarf zu behalten, reicht es in den heutzutage mehr und mehr geswitchten Netzwerken nicht mehr aus, die Leistungsgrößen an einem Knoten zu ermitteln, viel mehr wird eine Überwachung der Switche im Netz notwendig. Zu diesem Zweck wurde in der Vergangenheit bereits das Simple Network Managemant Protocol eingeführt, mit dessen Hilfe SNMP-fähige Netzwerkkomponenten abgefragt und geregelt werden können.

Im Rahmen von zwei an der Uni Dortmund veranstalteten Projektgruppen wurden als Grundlage bereits Tools zur Planung, Erfassung und Last- bzw. Normalbetriebssimulation eines Beispielnetz bei der Kreispolizeibehörde Unna entwickelt und testweise sowohl am die Projektgruppen ausschreibenden Lehrstuhl 4 als auch bei der Kreispolizeibehörde eingesetzt. Die dabei gewonnenen Erkentnisse flossen dabei teilweise in die Weiterentwicklung des Netzwerkes bei der Kreispolizeibehörde ein.

Die grundsätzliche Idee des hier vorgestellten Softwaretools Speedtrap ist es nun, in Folge der beiden angesprochenen, eher planerisch ausgerichteten Softwareprodukte ein Programm zur Hand zu haben, mit dessen Hilfe ein gegebenes Computernetzwerk erfasst und ver- bzw. gemessen werden kann. Dazu bedient sich Speedtrap vor allem der o.g. SNMP fähigen Hardwarekomponenten, also vor allem Switches. Von diesen Switches werden durch auf herkömmlichen Rechnern als Services installierten „Agenten" Daten gesammelt und weiterverarbeitet zu einem zentralen Software Monitor gesendet, der diese anfordert.

Ziel ist es, eine komplette Übersicht über das zu erfassende und messende Netzwerk und die darin auftretenden Vorgänge zu erhalten. Zu dem Zweck wird eine automatische Netzplanerkennung angeboten. Weiterhin ist es möglich, die Vorgänge im Netzwerk in Echtzeit an Hand einer grafischen Darstellung des Netzwerkes zu verfolgen. Zu den eigentlichen Programmfeatures gehören vor allem:

- automatische Strukturerkennung eines gegebenen Netzwerkes inklusive vorhandener Agenten, Switches, Hubs und Router
- semiautomatische Vorschläge zur Installation von Agenten und komfortable Konfiguration der installierten Agenten per Softwaremonitor
- Konfiguration von Messszenarien am zentralen Monitor unter Auswahl einer Vielzahl von möglichen Messdaten
- Verfolgung einer Messung in Echtzeit am Monitor, sowie Replay und Speichermöglichkeiten von Szenarien.
- geteilte Lastgenerierung durch die Agenten zur maximalen Kapazitätsbestimmung

Die Funktionsweise bzw. Bedienung des Monitors ist Gegenstand dieses Handbuchs. Soweit dies für den Anwender notwendig ist, wird auch auf die Installation der angesprochenen Agenten eingegangen.

**Anwendungsbeispiel**

*Zum besseren Verständnis werden am Ende jedes Kapitels, die besprochenen Funktionen anhand eines Anwendungsbeispiels erläutert. Als Beispiel wird dafür die „2. Messung" von Seite 61 dieses Endberichts gewählt. Das abschließende Anwendungsbeispiel ist, wie hier gezeigt, vom restlichen Text abgehoben.*

### Programmübersicht und Arbeitsweise von Speedtrap

Die eigentliche Funktionsweise und Bedienung von Speedtrap soll im folgenden an Hand einer Programmübersicht vorgestellt werden. Die weiteren Abschnitte spielen dann eine komplette Anwendung von der Installation über die Netzplanerkennung hin zur Messungskonfiguration und Auswertung durch.

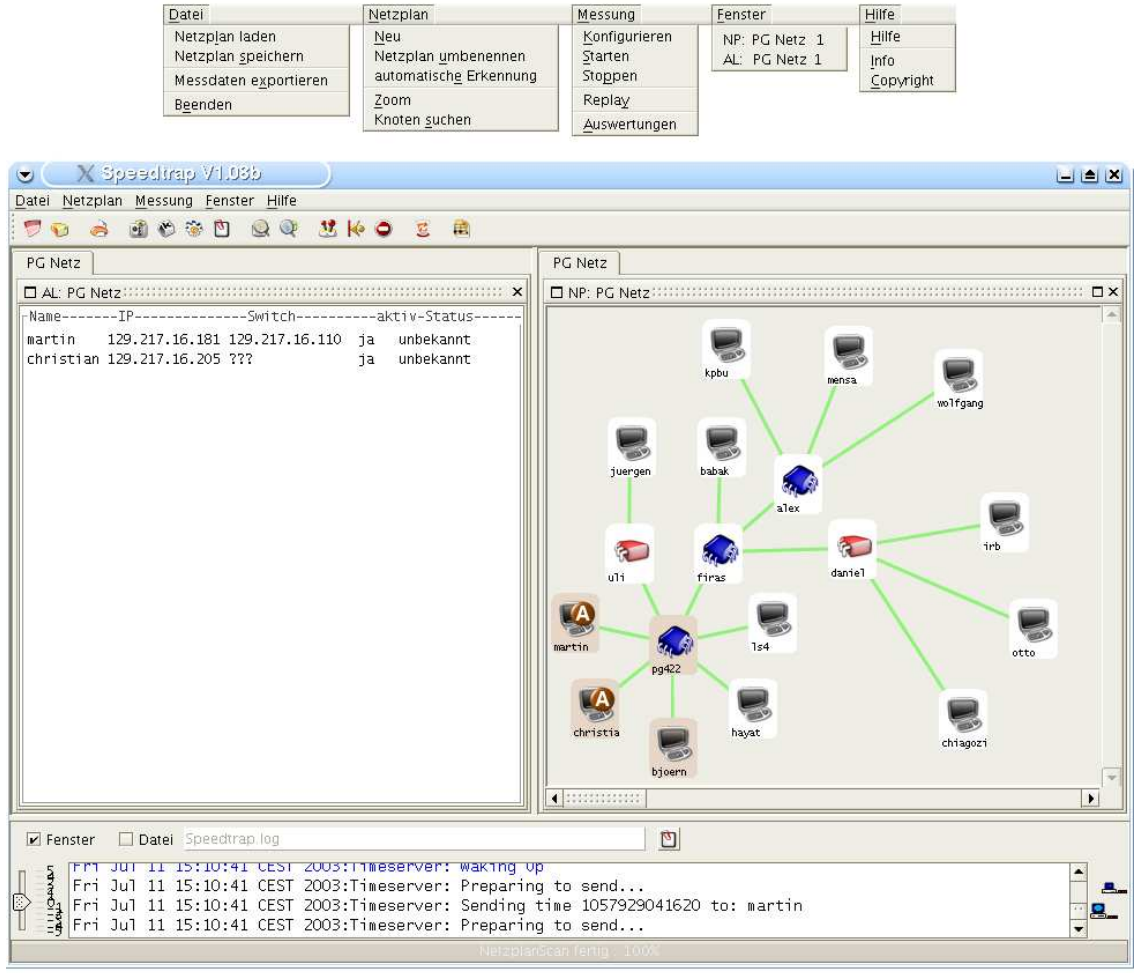



Abbildung 6.1: Benutzerschnittstelle im Überblick

Das Programm Speedtrap ist in Abbildung 6.1 im Einsatz mit gesondert dargestellter Menüstruktur gezeigt, auf die einzelnen Bedienelemente und Menükomponenten wird im folgenden näher eingegangen. Hier soll lediglich eine grobe Übersicht gegeben werden, die exakte Beschreibung der Vorgehensweise bei der Bedienung folgt wie angesprochen in den folgenden Absschnitten.

In der Mitte auf der rechten Seite befindet sich das Netzplanfenster. Man sieht hier einen fiktiven Netzplan mit Hubs, Switches und Rechnern, wie er von Hand angelegt oder automatisch erkannt sein kann. Netzelemente aus bekannten IP Subnetzen werden dabei farblich mit der vom Programm zum Subnetz ausgewählten Farbe hinterlegt. Per Rechtsklick auf die einzelnen Elemente kann man deren Eigenschaften wie IP Adresse oder Verbindungen untereinander ändern. Weiterhin ist es per dem Rechtsklick folgendem Kontextmenü möglich Agenten einen Switch zuzuweisen. Näheres dazu in den folgenden Abschnitten.

Die Verbindungen zwischen den Elementen sind hier grün dargestellt, verändern während einer laufenden Messung aber ihre Farbe gemäß den gemessenen Werten und werden mit Zahlenwerten versehen, näheres dazu in den folgenden Abschnitten.

Den Netzplan selber kann man per Rechtsklick auf eine freie Fläche sowohl zoomen, als auch mit zwei verscheidenen Anordnungsverfahren - radial und hierarchisch - neu darstellen.

Auf der linken Seite befindet sich die zum Netzplan gehörende Agentenliste. Sie listet Informationen zu den Agenten auf und per Doppelklick auf einen Listeneintrag, können die Einstellungen eines Agenten bearbeitet werden.

Unterhalb der beiden Hauptfenster befindet sich die Logausgabe, hier kann man Informationen des Monitors bzw. Fehlermeldungen von den Agenten und Debugmeldungen mitlesen. Über den Schieber auf der linken Seite lässt sich das Debuglevel einstellen, je niedriger das Level, desto mehr (evtl. für den normalen Anwender unnütze) Ausgaben zum Programmablauf erhält man. Oberhalb des Logfensters besteht auch die Möglichkeit, die Logausgabe zusätzlich in einer Datei mitschreiben zu lassen.

Oberhalb der beiden Hauptfenster befindet sich die Toolbar, über die die häufig verwendete Funktionen direkt aufgerufen werden können (wird hier nicht näher beschrieben) und darüber die Menüleiste die oberhalb gesondert dargestellt ist.

Im Datei Menü hat man die Möglichkeit, den dargestellten Netzplan zu speichern und zu laden, sowie evtl. gemessene, in dem Netzplan vorhandene Daten - dazu später mehr - in einer Textdatei zu exportieren.

Im Netzplan Menü kann man einen neuen Netzplan anlegen, den aktuellen Netzplan umbenennen, die automatische Erkennung eines neuen Netzplans starten, sowie den Netzplan nach Elementen durchsuchen. Auch dies wird im folgenden detaillierter erklärt.

Das Messung Menü ist geeignet um eine Messung auf dem aktuellen Netzplan zu konfigurieren, durch die Agenten zu starten und evtl. auch abzubrechen. Zusätzlich kann man sich über dieses Menü ein Replay der Messung im Schnelldurchlauf anzeigen lassen und Messdaten in der Auswertung betrachten.

Die beiden restlichen Menüs schließlich lassen zwischen den Fenstern - hier ist allerdings jeweils nur ein Netzplan und Agentenfenster dargestellt - wechseln und bieten Zugang zur Hilfe und den obligatorischen Copyright und Impressumsinformationen.

### 6.1.2 Installation und Start der Anwendungen

Die Installation der beiden Speedtrap Komponenten, *Monitor* und *Agenten* wird ausführlich im Administrator-Handbuch in Kapitel 6.2 ab Seite 103 beschrieben. Nachdem die Komponenten installiert wurden, können sie wie folgt gestartet werden:

#### Start der Agenten unter Unix/ Linux

Vorausgesetzt, dass die Installation korrekt abgeschlossen wurde, kann Speedtrap wie folgt gestartet werden:

- Monitor: `{Java-Pfad}/java -jar {Speedtrap-Pfad}/Speedtrap.jar`
- Agent: `{Java-Pfad}/java -jar {Speedtrap-Pfad}/Agent.jar`

#### Start der Agenten unter Windows

Sofern die JVM korrekt unter Windows installiert wurde, genügt ein Doppelklick auf die entsprechende JAR-Datei, um den Monitor oder Agenten zu starten. Ansonsten erfolgt der Aufruf ähnlich zu dem unter Unix bzw. Linux:

- Monitor: `{Java-Pfad}\java -jar {Speedtrap-Pfad}\Speedtrap.jar`
- Agent: `{Java-Pfad}\java -jar {Speedtrap-Pfad}\Agent.jar`

Der Agent kann alternativ auch als Dienst installiert werden, was ebenfalls im Administratorhandbuch detailiert beschrieben wird.

**Anwendungsbeispiel**

*An unserem Anwendungsbeispiel sind vier Rechner beteiligt. Auf dem Rechner* ls4int07 *wird der Monitor installiert und die Rechner* ingo, bernd *und* ls4int05 *erhalten je einen Agenten, der entweder als Lastgenerator dient, oder den Switch* switch1 *abfragt. Die jeweiligen* .JAR*-Dateien sind im* home*-Verzeichnis der Rechner verfügbar. Da auf den genannten Rechnern die Java-Runtime so installiert ist, dass sie über die Pfadvariablen gefunden werden kann, genügen die folgenden Aufrufe im* home*-Verzeichnis der Rechner, um die jeweilige Anwendung zu starten:*

```
java -jar Speedtrap.jar
java -jar Agent.jar
```

*Die Anwendungen sind nun auf den entsprechenden Rechnern vorbereitet.*

### 6.1.3 Netzwerkerkennung

Über eine automatische Netzwerkerkennung werden Rechner, Switches, Router und bereits installierte Agenten erkannt. Hubs werden eingefügt, wo Kollisionsdomänen gefunden werden.

Nach einer Netzwerkerkennung kann der erkannte Netzplan manuell angepasst werden. Ein Netzplan kann auch komplett manuell erstellt werden. Die folgenden Unterkapitel zeigen wie dies geht.

Unser Anwendungsbeispiel beschreibt die Arbeitsschritte, die nötig waren, um die Messungen für die Verifikation in Kapitel 4 durchzuführen. Abbildung 4.5 zeigt den Netzplan der dafür erstellt wurde.

#### Die automatische Netzwerkerkennung

Nach dem Start von Speedtrap muss zuerst ein neuer Netzplan erstellt werden. Alternativ könnte man auch einen gespeicherten Netzplan laden, das wird in einem späteren Kapitel beschrieben.

Der Eintrag *Neu* im Menü *Netzplan* erzeugt einen neuen leeren Netzlpan. Wer eine automatische Erkennung durchführen will, kann auch direkt *automatische Erkennung* wählen. Wenn noch kein Netzplan erzeugt wurde, ist es in jedem Fall sinnvoller über die *automatische Erkennung* einen neuen Netzplan zu erzeugen.

Nach einem Klick auf *automatische Erkennung* erscheint der Dialog Adressbereiche wählen (Abb. 6.2). In diesem wird der Adressbereich festgelegt, in dem nach Komponenten gesucht werden soll. In den Feldern links oben wird ein Adressbereich in der Form *Basisadresse / Netzmaske* eingegeben. Die Netzmaske legt dabei fest, wieviele Bits (von links) der Basisadresse unverändert bleiben. Eine

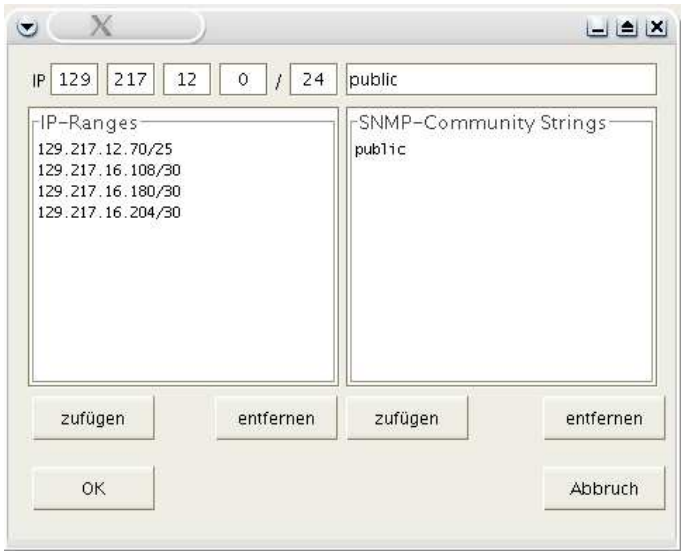



Abbildung 6.2: Dialog „Adressbereiche wählen"

Netzmaske von 24 ergibt also ein Class C Netz, Eine Netzmaske von 8 ein Class A Netz. Beispiel: 192.168.99.0/24 ergibt einen Adressbereich von 192.168.99.0 bis 192.168.99.255.

Mit dem Button *zufügen* unter dem Feld *IP-Ranges* wird der Adressbereich in dieses Feld hinzugefügt. In dem Feld werden alle zu scannenden Bereiche aufgelistet. So kann man, wenn man will, einen Adressbereich aus mehreren Bereichen zusammenstellen. Markierte Bereiche können mit dem *entfernen*-Button wieder gelöscht werden.

Auf der rechten Seite des Dialogs können SNMP-Community-Strings für die Switche angegeben werden. Die Funktionsweise des *zufügen*- und *entfernen*-Button ist genauso wie bei den Adressbereichen. Die Community-Strings sind die Passwörter für die Switches. Man kann mehrere eintragen. Sie werden nacheinander für jeden Switch durchprobiert. Wenn der Switch antwortet wird der passende String dem Switch zugeordnet und gespeichert.

Mit einem Klick auf *OK* wird die Erkennung gestartet.

**Anwendungsbeispiel**

*In unserem Anwendungsbeispiel starten wir zunächst die automatische Netzwerkerkennung. Als Parameter tragen wir 129.217.16.0/24, weil unsere Rechner alle innerhalb des Bereichs zwischen 129.217.16.1 und 129.217.16.255 liegen. Als SMNP-Community-String dient „speedtrap". Der geliehene Switch ist entsprechend konfiguriert.*
*Das Ergebniss dieses Scans stimmt schon fast mit Abbildung 4.5 überein. Lediglich der Rechner* ls4int05 *fehlt noch, weil dieser zum Zeitpunkt des Scans noch nicht angeschlossen ist. Dieser Rechner wird manuell hinzugefügt. Das nächste Kapitel beschreibt, wie das geht.*

### Korrektur des automatischen Netzwerkplans

Nach einer Erkennung kann der erkannte Netzplan weiter angepasst werden. Genauso kann auch ein Netzplan komplett von Hand erstellt werden. Alle Aktionen laufen über das Kontext-Menü.

Zunächst kann der Netzplan umbenannt werden. Standardmäßig wird ein neuer Netzplan „Netzplan 1" genannt. Jeder weitere erstellte Plan bekommt eine um jeweils 1 erhöhte Nummer. Unter

dem Eintrag *Netzplan umbenennen* kann dieser Name gegen einen Aussagekräftigeren ersetzt werden.

Das Kontext-Menü des Hintergundes des Netzplans enthält an erster Stelle den Eintrag *Neu*. Mit diesem können Rechner, Switches, Hubs und Router erzeugt werden. Es entsteht eine neue Komponente. Bei Rechnern, Switches und Routern muss anschliessend die Anzahl der Interfaces (entspricht bei Switches den Ports) eingegeben werden.

Die Eigenschaften der neu erzeugten Komponente können über das Kontext-Menü der Komponente unter dem Eintrag *Eigenschaften* bearbeitet werden. Der Eigenschaften-Dialog gibt die Möglichkeit, den Name, einen Kommentar, die IP-Adresse und die MAC-Adresse festzulegen. Bei Switches kann zusätzlich unter *SNMP-Auth* der SNMP-Community-String eingetragen werden.

Ein Agent kann auf einem Rechner installiert werden. Dazu gibt es im Kontext-Menü eines Rechners den Eintrag *Agent installieren*. Dadurch wird der Rechner mit einem „A" markiert und der Eigenschaften-Dialog dieser Komponente wird erweitert um die Agent-Konfigurations-Felder. Mit dem Eintrag *Agent löschen* kann der Agent von dem Rechner wieder entfernt werden.

Verbindungen zwischen Komponenten werden ebenfalls über das Kontext-Menü der jeweiligen Komponenten bearbeitet. Jede Komponente hat den Eintrag *Verbinden*, *Verbindung ändern* und *Verbindung löschen*. Nach einem Klick auf *Verbinden* muss zunächst in einem Dialog der Port (bzw. das Interface) gewählt werden das verbunden werden soll. Anschliessend muss die Komponente angeklickt werden, mit der verbunden werden soll, und auch der Port dieser Komponente ausgewählt werden.

Verbindungen können geändert werden, in dem man *Verbindung ändern* auf der Komponente wählt, die von einer anderen Komponente getrennt und an eine neue Komponente angehängt werden soll. Anschliessend wählt man die Komponente aus, von der die Komponente getrennt werden soll. Die Verbindung wird gelöscht. Abschliessend muss man die Komponente wählen, mit der die zuerst gewählte Komponente verbunden werden soll und wieder den passenden Port wählen.

Verbindungen werden gelöscht, indem man auf einem der Endpunkte der Verbindung *Verbindung löschen* auswählt und anschliessend auf die zweite Komponente der Verbindung klickt.

**Anwendungsbeispiel**

*Um den Rechner* ls4int05 *hinzuzufügen, wählt man also aus dem Kontext-Menü des Netzplans den Punkt* Neu *und dann* Rechner. *Die Anzahl der Interfaces ist in diesem Fall 1 für genau eine Netzwerkkarte. Im Eigenschaften-Dialog (erhältlich durch Wahl von* Eigenschaften *im Kontextmenü des Rechners) wird jetzt noch der Name, die IP-Adresse und die MAC-Adresse eingetragen.*
*Der Rechner wird jetzt noch mit dem Switch verbunden. Dazu wählt man* Verbinden *im Kontext-Menü des Rechners und wählt Port 0 (das einzige Interface des Rechners). Anschliessend klickt man auf den Switch und wählt den richtigen Port, in diesem Fall Port 7.*

### Speichern/ Aufrufen des Netzplans

Der fertige Netzplan kann in eine Datei gespeichert werden. Unter dem Menüpunkt *Datei* gibt es dazu den Eintrag *Netzplan speichern*. Wählt man *Netzplan speichern* muss man anschliessend nur noch in einem Standard-Dateiauswahl-Dialog eine Datei auswählen. In die gewählte Datei wird der Netzplan gespeichert.

Auf gleiche Weise kann ein gespeicherter Netzplan aus einer Datei geladen werden. Der Eintrag *Netzplan laden* im Menü *Datei* führt wieder zu einem Dateiauswahl-Dialog. Der Netzplan wird aus der gewählten Datei geladen und auf dem Bildschirm dargestellt.

### Verteilung und die Zuweisung der Aufgaben der Agenten

Die Agenten werden (wie bereits erwähnt) über den Eigenschaften-Dialog konfiguriert.

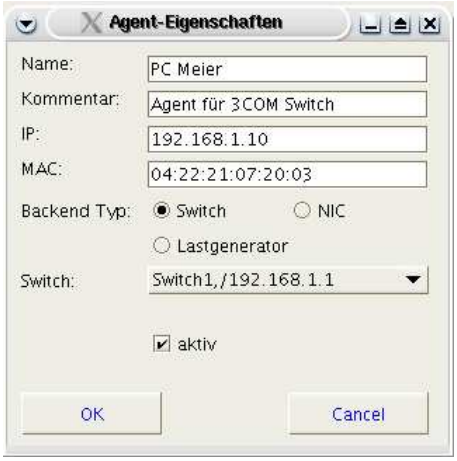



Abbildung 6.3: Eigenschaften-Dialog eines Agenten

Abbildung 6.3 zeigt diesen Dialog. Zusätzlich zu den bekannten Feldern Name, Kommentar, IP und MAC gibt es die Felder „Backend Typ" und „aktiv". Bei „Backend Typ" hat man die Wahl zwischen Switch, NIC und Lastgenerator.

Switch bedeutet dabei, das ein Switch per SNMP abgefagt werden soll. Zusätzlich muss dann noch der Switch aus der Liste gewählt werden, der überwacht werden soll. Diese Liste erscheint nur, wenn Backend Typ "Switch" gewählt wurde.

NIC bedeutet, dass das Segment, an dem der Agent angeschlossen ist, an der Netzwerkkarte überwacht werden soll. Ist der Agent als Lastgenerator konfiguriert, schickt er Pakete an einen anderen Rechner um die Netzlast zu erhöhen. Für einen Lastgenerator erscheinen zwei weitere Felder in der die Zieladresse und die Grösse der Pakete eingestellt werden muss.

Ein Häkchen im Feld „aktiv" sorgt dafür, das bei der nächsten Messung, der Agent aktiviert wird. Will man den Agenten nicht verwenden, kann man ihn durch Wegnehmen des Häkchens deaktivieren.

### Suche nach Netzwerkkomponenten

Eine Möglichkeit, um bestimmte Komponenten in grösseren Netzplänen zu finden, ist die Funktion *Rechner suchen*. Sie ist verfügbar über das Kontext-Menü des Hintergrunds des Netzplans. Zusätzlich kann sie auch im *Netzplan*-Menü als *Knoten suchen* gewählt werden.

Es erscheint ein Dialog in dem der Name, oder ein Teil davon eingetragen werden kann. Wird eine Komponente gefunden, die die eingegebene Zeichenkette im Namen oder der IP-Adresse enthält, blinkt diese für 5 Sekunden.

Falls man also die Komponente sucht, dessen IP-Adresse mit 127 endet, kann man nach „.127" suchen und wird sie so leicht finden.

## 6.1.4 Konfigurieren der Messung

Nach einer erfolgreichen Darstellung des Netzplans ist das Konfigurieren des/der Agent(en) und das Konfiguerieren einer Messung erforderlich.

### Agenten konfigurieren

Für jeden Switch im Netz, muss ein eigener Agent installiert werden. Bereits installierte Agenten werden bei der automatischen Netzplanerkennung gefunden und entsprechend im Netzplan angezeigt. Die Rechner für die Installation der Agenten sollten so gewählt werden, dass sie unmittelbar an dem zu überwachenden Switch hängen. Wählt man im Kontextmenü eines Switches (Rechtsklick auf den Switch) den Punkt „Agenten vorschlagen", so blinken für ca. 10 Sekunden bis zu fünf Rechner, die sich als Basis für die Installation eines Agenten eignen. In Netzwerken, in denen kein (SNMP fähiger) Switch vorhanden ist, kann ein einzelnes Hub-Segment über die Netzwerkkarte eines Agentenrechners gemessen werden. Hierbei muss der Agentenrechner zwingend direkt an dem zu überwachenden Hub angeschlossen werden und wie unten beschrieben das NIC-Backend gewählt werden. Die Agenten, die während einer automatischen Netzplanerkennung oder durch manuelle Installation durch den Benutzer im Netzplan vorhanden sind, sind nicht automatisch konfiguriert, d.h. sie haben noch keine Zuweisung, welche Aufgabe sie zu erfüllen haben. Daher muss jeder Agent vor einer Messung manuell konfiguriert werden. Dabei kann ein Agent bei einer Messung nur eine von drei möglichen Aufgaben übernehmen. Ein Agent kann entweder einen Switch (bzw. ein SNMP-fähiges Gerät) überwachen, direkt an einer Netzwerkkarte messen oder als Lastgenerator dienen. Zur Konfiguration eines Agenten muss man den Menüeintrag „Eigenschaften" aus dem Kontextmenü des Agenten wählen, oder den Agenten in der *Agentenliste* mit einem Doppelklick auswählen. Das Kontextmenü öffnet man mit einem Klick mit der rechten Maustaste auf den entsprechenden Agent. Im Eigenschaftsfenster (Abbildung 6.4) des Agenten können dann die Einstellungen vorgenommen werden.

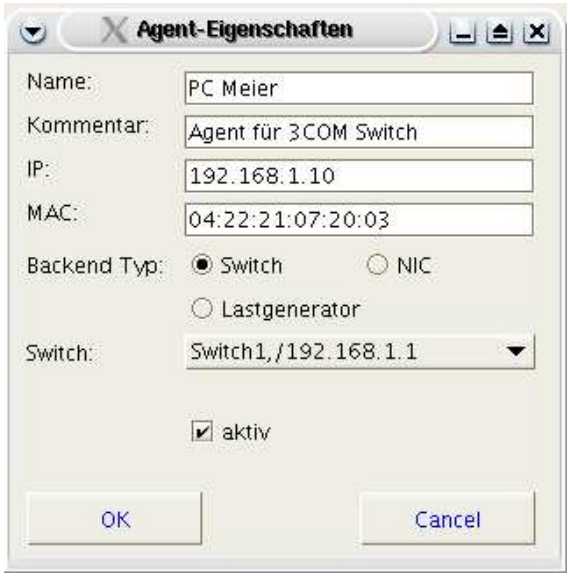



Abbildung 6.4: Dialog „Agent-Eigenschaften (Switch Backend)"

Sofern dies nicht bereits bei der automatischen oder manuellen Erstellung des Netzplans geschehen ist, kann dem Agenten in diesem Dialog ein Name und Kommentar, sowie seine IP- und MAC-Adressen zugeordnet werden. Die IP-Adresse ist in jedem Fall für die Kommunikation zwischen Monitor und Agent notwendig. Die übrigen Angaben sind optional.

Anschließend muss eine der bereits oben erwähnten Funktionen für den Agenten ausgewählt werden. Je nach Wahl sind dann ggf. zusätzliche Angaben zu machen:

**Switch-Backend:** Bei Auswahl des Switch-Backend muss dem Agenten mitgeteilt werden, welchen Switch er überwachen soll. Dieser wird aus einer Liste ausgewählt, die alle im aktuellen

Netzplan vorhandenen Switche enthält (Abbildung 6.4).

**NIC-Backend:** Bei dieser Konfiguration misst der Agent an seiner lokalen Netzwerkkarte. Eine weitere Konfiguration ist nicht notwendig (Abbildung 6.5).

**Lastgenerator:** Bei dieser Wahl muss eine IP-Adresse des Zielknotens für die Lastgenerierung angegeben werden. Außerdem wird die zu verwendende Paketgröße eingestellt (Abbildung 6.6).


Abbildung 6.5: Dialog „Agent-Eigenschaften (NIC Backend)"

Das Häkchen "aktiv", aktiviert den Agenten für die nächste Messung. Agenten bei denen dieses Häkchen nicht gesetzt ist, nehmen an der nächsten Messung nicht teil. Mit einem Klick auf „OK" werden die Eingaben übernommen und gespeichert. Jeder Agent muss wie oben beschrieben konfiguriert werden.

*Tipp:* Wenn die Konfiguration der Agenten nach dem Schließen des Netzplans erhalten bleiben soll, muss der Netzplan nach dem konfigurieren gespeichert werden. Beim nächsten Laden des Netzplans ist dann die Konfiguration wieder verfügbar und muss nicht erneut eingegeben werden.

**Anwendungsbeispiel**

*In unserem Anwendungsbeispiel müssen nun die Agenten festgelegt und konfiguriert werden. Durch die Auswahl „Agenten vorschlagen" des Kontextmenüs von* switch1*, werden durch blinken fünf Rechner signalisiert, die sich für die Aufnahme eines Agenten eignen. In unserem Beispiel wählen wir die drei Rechner* ingo, bernd *und* ls4int05 *als Agenten. Mit dem Kontextmenüeintrag „Agent installieren", installieren wir auf den entsprechenden Geräten die Agenten. Die IP-Adresse und der Rechnername wurden bereits durch die automatische Netzwerkerkennung, oder manuell beim Erstellen des Netzplans vergeben, so dass dem Agenten nur noch seine Funktion während der Messung zugewiesen werden muss. Der Rechner* ls4int05 *soll die eigentliche Messung übernehmen, indem er den Switch* switch1 *überwacht. Aus diesem Grund wird für ihn das Switch-Backend gewählt und als Switch der* switch1 *gewählt und der Agent über das Häkchen „aktiv" für die nächste Messung aktiviert. Die Rechner* ingo *und* bernd *sollen während der Messung als Lastgeneratoren dienen. Daher wird für beide Rechner das Lastgenerator-Backend gewählt und als Gegen-IP die*

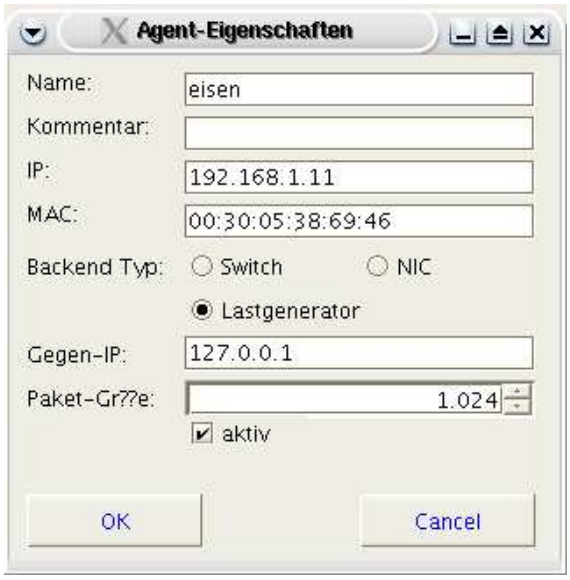



Abbildung 6.6: Dialog „Agent-Eigenschaften (Last Generator)"

*IP-Adresse des Rechners* elke *angegeben, der die Lastpakete empfangen soll. Die IP-Adresse des Rechners* elke *kann über dessen Eigenschaften-Dialog in Erfahrung gebracht werden. Die Paketgröße am Rechner* ingo *wird auf 64kB, bei* bernd *auf 32kB festgelegt. Die Agenten sind damit für die nächste Messung vorbereitet.*
*Vor dem Fortfahren sollte nun sichergestellt werden, dass die Agenten auf den entsprechenden Rechnern gestartet wurden (s. Kapitel 6.1.2).*

### Messkonfiguration

Über den Menüpunkt *Konfigurieren* im Menü *Messung* wird ein Dialog geöffnet, in dem die Einstellungen für eine Messung vorgenommen werden (Abbildung 6.7). Der Dialog wird auch geöffnet, wenn der Punkt *Starten* im Menü *Messung* ausgewählt wird, und zuvor noch keine Messung konfiguriert wurde. Im Messkonfigurations-Dialog wird zunächst die Start und Endzeit der Messung eingestellt. Anschließend werden die zu messenden Daten bestimmt. Dabei kann aus den in Tabelle 6.1 dargestellten Daten gewählt werden. Die jeweiligen Messdaten beziehen sich wie in der Tabelle angegeben, entweder auf Kanten im Netzwerk bzw. die Ports der jeweiligen Geräte, oder auf das Gerät (den Knoten) selbst. Es können einzelne oder alle (*Strg+a*) Messdaten selektiert werden. Schließlich muss noch das Intervall in Sekunden bestimmt werden, in dem die Agenten ihre Messdaten sammeln. Da Switche relativ lange brauchen, um auf SNMP-Anfragen zu antworten, macht hier ein Wert unter 10 Sekunden keinen Sinn. Damit ist nun festgelegt, welche Werte von den Agenten, in welchem Rhythmus gemessen werden sollen.

### Konfiguration des Echtzeitverhaltens

Im nächsten Teil des Konfigurationsdialogs wird das Verhalten von Speedtrap während einer Messung konfiguriert. Über das Häkchen *GUI* wird festgelegt, dass Speedtrap während einer Messung eines der Messdaten im aktuellen Netzplan anzeigen soll. Dieses Datum kann über die Combo-Box *Echtzeit Datum* gewählt werden. Hier stehen die gleichen Messwerte wie bei der generellen

| Messdatum | gemessen an |
|---|---|
| empfangene Bits | Kanten/Ports |
| gesendete Bits | Kanten/Ports |
| Unicast Pakete | Kanten/Ports |
| non Unicast Pakete | Kanten/Ports |
| fehlerhafte Pakete | Kanten/Ports |
| maximale Bandbreite | Kanten/Ports |
| MTU | Kanten/Ports |
| operational Status | Kanten/Ports |
| Kollisionen | Kanten/Ports |
| Interface Beschreibung | Kanten/Ports |
| Interface Typ | Kanten/Ports |
| in Utilization | Kanten/Ports |
| out Utilization | Kanten/Ports |
| Anzahl IP Fragmentation | Knoten |
| gesendete TCP Pakete | Knoten |
| empfangene TCP Pakete | Knoten |
| gesendete UDP Pakete | Knoten |
| empfangene UDP Pakete | Knoten |
| gesendete IP Pakete | Knoten |
| empfangene IP Pakete | Knoten |

Tabelle 6.1: Messdaten

Konfiguration der Messung zur Verfügung (Tabelle 6.1). Die Einstellung des Refreshintervalls (*Refreshint.*) gibt an, wie häufig die Netzplanansicht mit den gemessenen Daten aktualisiert werden soll. Es ist darauf zu achten, dass für die Darstellung der Messdaten, der Monitor während der Messung mit den Agenten kommunizieren muss, um die Messdaten zwischenzeitlich abzurufen. Es entsteht also zusätzlicher Netzverkehr, der die Messung beeinflussen kann.

Die Messung ist nun vollständig konfiguriert. Mit einem Klick auf *OK* werden die Einträge bestätigt und gespeichert. Um die Messung zu starten ist im folgenden auf den Eintrag *Starten* in Menü *Messung* zu klicken.

*Tipp*: Die Konfiguration und das Starten von Messungen können auch über die Symbolleisten vorgenommen werden. Wenn man den Mauszeiger über ein Symbol platziert wird die Funktion dieses Symbols eingeblendet.

**Anwendungsbeispiel**

*Um die Messung für unser Anwendungsbeispiel zu konfigurieren, wird das Messkonfigurationsmenü aufgerufen. Die Start- und Endzeit werden so festgelegt, dass ein zukünftiger Zeitraum von zwei Minuten beschrieben wird. Durch drücken der Tastenkombination* Strg+a *in der Auswahl der Messdaten, werden alle Messdaten selektiert. Das Messintervall wird bei 10 Sekunden belassen und durch Auswahl des Häkchens „GUI", die Echtzeitdarstellung während der Messung aktiviert. Für das Refreshintervall wird ebenfalls ein Wert von 10 Sekunden gewählt und schließlich eines der verfügbaren Messdaten aus der Liste für das Echtzeit Datum gewählt und die Eingaben durch einen Klick auf „OK" gespeichert. Die Messung kann nun über das Menü „Messung starten", bzw. das äquivalente Symbol in der Symbolleiste gestartet werden.*

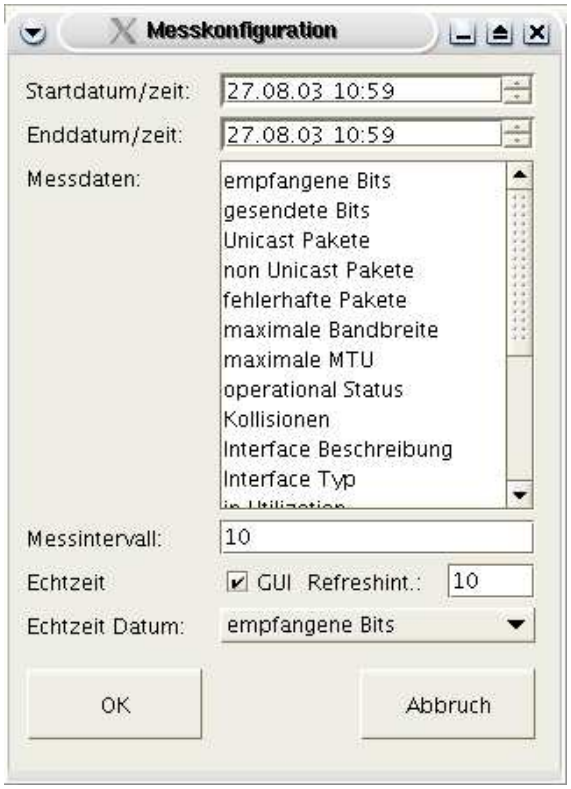



Abbildung 6.7: Dialog „Messkonfiguration"

### 6.1.5 Auswertung

Speedtrap bietet verschiedene Möglichkeiten die erhobenen Messdaten darzustellen. So können während einer Messung die jeweils aktuellen Daten im Netzplan dargestellt werden und nach der Messung nochmals über die Replay Funktion wiedergegeben werden. Außerdem kann nach der Messung eine Zusammenfassung der Messdaten angezeigt werden und die Daten können in einem für Tabellenkalkulationsprogramme geeignetem Format exportiert werden, was eine grafische Aufbereitung durch externe Software möglich macht.

#### Betrachten der Real-Time Daten

Wurde bei der Konfiguration der Messung die Realtime Darstellung über das Häkchen „Echtzeit" in der Messungskonfiguration aktiviert, so wird das ausgewählte Echtzeitdatum während der Messung im Netzplan dargestellt. Die Einstellung des „GUI Refreshintervalls" gibt dabei an, wie häufig (in Sekunden) die Ansicht aktualisiert werden soll. Die Aktualisierung der Ansicht hat zur Folge, dass der Monitor die Messdaten zwischenzeitlich von den Agenten abrufen muss. Dies tut er in genau diesem eingestellten Refreshintervall. Es ist zu beachten, dass dadurch zusätzlich Verkehr auf dem Netz – nämlich zwischen den Agenten und dem Monitor – erzeugt wird, der eventuell Einfluss auf die Messung hat. Außerdem ist ein Refreshintervall von weniger als 10 Sekunden nicht sinnvoll, da sonst für die Switche nicht genügend Zeit zum Antworten bleibt.

Die Darstellung der Messwerte erfolgt numerisch und farblich an den Kanten und Knoten des Netzplans. Die Farbe des Objekts wird dabei zwischen 0 und dem bis zu dem aktuellen Zeitpunkt maximalen Messwert der aktiven Messung skaliert. Die Farbskala reicht von *Grün* (0) über *Blau* bis *Rot* (maximaler Wert). Diese Art der Farbkodierung hat folgende Besonderheiten zur Folge:

- Der erste Messwert einer Messung, der größer als 0 ist, wird in jedem Fall *Rot* dargestellt.
- Die Farbe *Rot* bedeutet nicht, dass ein Zustand oder Wert kritisch ist, sondern bloß, dass es der bisher größte gemessene Wert ist.
- Ein Wert, der zu Beginn einer Messung eine bestimmte Farbe hatte, kann zu einem späteren Zeitpunkt durch eine andere Farbe dargestellt werden.

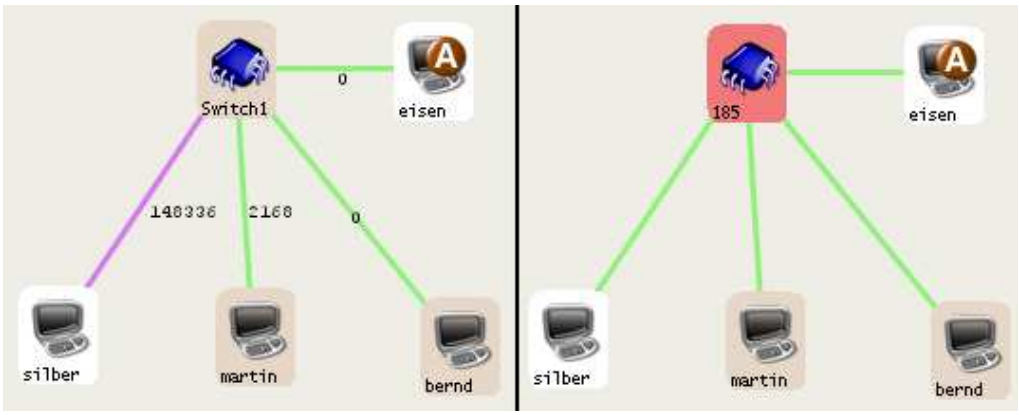



Abbildung 6.8: Echtzeitdarstellung der Messdaten

Wurde als Echtzeitdatum ein Messdatum ausgewählt, das sich auf die Leitungen des Netzes bezieht, so wird der entsprechende Messwert an die die Leitung repräsentierende Kante des Netzplans geschrieben und die Kante wie oben beschrieben eingefärbt (Abb. 6.8 links). Handelt es sich um ein Messdatum, dass sich auf einen Switch, bzw. einen anderen Knoten bezieht, so wird der Messwert anstelle des Namens des Knotens angezeigt und der Hintergrund des Knotenobjekt entsprechend gefärbt (Abb. 6.8 rechts). Nach der Messung erhält der Hintergrund wieder seine ursprüngliche Farbe und der normale Name des Gerätes wird wieder eingeblendet.

Solange eine Messung läuft, wird der aktuelle Status der Messung in einem kleinen seperaten Fenster dargestellt. In diesem Fenster werden die Start-, End- und die aktuelle Uhrzeit angezeigt, sowie welches Echtzeitdatum angezeigt wird und welches das derzeit dargestellte Messintervall ist. Zusätzlich wird am unteren Rand der Status der Messung in einem Forstschrittsbalken dargestellt. Die Messung durchläuft dabei drei Phasen:

**auf Startzeit warten:** Die Konfiguration der Messung und das Startsignal wurden an die Agenten gesendet. Die Messung ist jedoch noch nicht aktiv, da die ausgewählte Startzeit noch nicht erreicht ist. Der Fortschrittsbalken zeigt die Zeit bis zum Startzeitpunkt an.

**Messung läuft:** Die Startzeit der Messung ist vorbei und die Messung ist aktiv. Der Fortschrittsbalken zeigt die Zeit bis zum Ende der Messung. Im Netzplan wird das Messdatum angezeigt, sofern die Echtzeitansicht gewählt wurde.

**Operationen abschließen:** Die Agenten haben ihre Messung beendet, da die Endzeit der Messung erreicht wurde. Es sind jedoch noch nicht alle Messdaten an den Monitor übertragen und ausgewertet worden. Der Fortschrittsbalken blinkt bis zum endgültigen Abschluss der Messung. Die Darstellung der Messdaten im Netzplan ist ebenfalls noch nicht beendet.

Auch während einer Messung stehen die Funktionen des Netzplans, wie zum Beispiel *Zoom, Rechner suchen, -löschen, -erstellen, etc.* zur Verfügung.

### Die Replay Funktion

Nach Ablauf einer Messung können die gesammelten Daten über die *Replay-Funktion* erneut im Netzplan dargestellt werden. Die *Replay-Funktion* ist über das Hauptmenü oder die Toolbar erreichbar (Abbildung 6.1). Nach dem Aufruf dieser Funktion wird der Messkonfigurationsdialog mit eingeschränkten Auswahlmöglichkeiten angezeigt. Es kann nur das im Replay darzustellende GUI Messdatum ausgewählt werden und das Intervall, in dem die Daten wiedergegeben werden, angegeben werden. In der Regel wird man hier den minimalen Wert von 1 Sekunde wählen. Wird ein Messdatum ausgewählt, das während der wirklichen Messung nicht erhoben wurde, so wird während des gesamten Replay Vorgangs der Wert 0 an den entsprechenden Objekten angezeigt.

### Auswertung und tabellarische Ansicht der Daten


Abbildung 6.9: Auswertungs Dialog

Die tabellarische Ansicht erreicht man über das Feld „Auswertung" im Menüpunkt „Messung" (Abb. 6.1). Zunächst hat man jetzt die Wahl, welche Parameter angezeigt werden sollen (Abb. 6.9). Zur Auswahl stehen:

**Anzahl Pakete** - Unicast Pakete + Nonunicast Pakete

**Maximale Bandbreite** - maximale Bandbreite in Kb/sec.

**Anzahl Bits** - empfangene Bits + gesendete Bits

**Fehlerhafte Pakete** - Anzahl fehlerhafter Pakete

**Kollisionen** - Anzahl der Kollisionen im Switch

**Auslastung** - Maximum aus eingehender und ausgehender Auslastung

**Fehlerhäufigkeit** - $\frac{FehlerhaftePakete}{(UnicastPakete+MulticastPakete)}$

**Güte** - 1 - Fehlerhäufigkeit

Es können entweder die Daten der kompletten Messung abgefragt werden oder nur Daten aus einem bestimmten Zeitabschnitt. Um die gesamten Daten abzufragen, muss die Checkbox „komplett" selektiert werden. Um die Daten eines bestimmten Zeitabschnitts abzufragen, muss die Checkbox „von-bis" selektiert werden. Es erscheinen dann zwei Eingabefelder, in die der Anfangs- und der Endzeitpunkt der auszulesenden Daten eingegeben werden kann. Mit „OK" gelangt man dann zur eigentlichen Ansicht. Die erste Spalte enthält stets die IP-Adresse und den Hostnamen des Netzwerkgeräts, über das die Daten ausgegeben werden. Besonderheiten sind dabei, dass Geräte, die an einem Hub hängen, immer die gleichen Werte haben. Das liegt daran, dass sie in einem Segment hängen, das von allen Rechnern im gleichen Segment belastet wird. Verbindungen zwischen zwei

Switchen sind besonders gekennzeichnet. Sie haben in der Spalte der IP Adressen zwei Adressen, die durch einen Slash getrennt sind (../..) stehen. Diese zwei Adressen stehen für die Adressen der beiden Switche, zwischen denen die ausgewertete Verbindung läuft. Solche Verbindungen werden zwei mal aufgeführt. Einmal mit den Messergebnissen des einen Switches, und einmal mit den Messergebnissen des anderen Switches. Diese Ergebnisse können mitunter variieren, was auf eine Ungenauigkeit in der Hardware zurückschliessen läßt. Bestimmte Parameter können je nach Messung nicht angezeigt werden. Möchte man sich beipielsweise die Fehlerrate ansehen, hat aber in der Messung „fehlerhafte Pakete" nicht selektiert, erscheint in der Spalte die Meldung „Berechnung nicht möglich". Die Auswertung der Messergebnisse kann je nach Menge der ausgewerteten Daten und Geschwindigkeit des PCs bis zu einer Minute und mehr in Anspruch nehemen.

#### Export der Daten in das *.csv Format

Über den Menüpunkt „Daten exportieren" (Abbildung 6.1) können die gemessenen Daten in eine Datei mit dem sogenannten *CSV-Format* exportiert werden. Dabei werden die Daten in Tabellenform, getrennt durch Semikolon gespeichert. Dieses Format wird durch alle gängigen Tabellenkalkulationsprogramme unterstützt, was die Möglichkeit bietet, die Messwerte mit externen Programmen auszuwerten (z.B. MS-Excel).

Beim Exportieren werden alle möglichen Messwerte ausgegeben. Messwerte, die während der Messung nicht erhoben wurden, werden mit Nullen gefüllt. Die erste Zeile der Ausgabedatei enthält die Spaltenüberschriften. Danach folgen die Messdaten in der folgenden Sortierung:

1. Switch (zufällig)
2. Intervallnummer bzw. Intervallzeit (0 bis $(Intervallanzahl - 1)$)
3. Portnummer (aufsteigend)

**Anwendungsbeispiel**

*Da während unserer Beispielmessung ein Echtzeitdatum ausgewählt war, konnte während der Messung beobachtet werden, wie die Messdaten an die entsprechenden Objekte geschrieben wurden und diese eingefärbt wurden. Aus der Menüleiste oder der Symbolleiste kann nun die Replay-Funktion aufgerufen werden. Es öffnet sich erneut der Messungskonfigrations-Dialog, wobei jedoch nur die Echtzeiteinstellungen manipuliert werden können. Wird nun ein anderes Messdatum gewählt und das Refreshintervall auf 1 Sekunde gesetzt, so wird der Messungsverlauf für das gewählte Messdatum in 10-fachem Zeitraffer wiedergeben (die 10 Sekunden Intervalle der Messung werden im Replay im 1 Sekunden-Takt wiedergegeben).*
*Nun kann über das Menü die Auswertung der Messdaten aufgerufen werden. In dem Auswertungsdialog werden alle Daten selektiert und der Zeitaum auf „komplett" eingestellt. Anschließend werden die oben aufgezählten Daten angezeigt.*
*Über den Menüpunkt „Daten exportieren" oder das entsprechende Symbol können die Messdaten schließlich ins* CSV *Format exportiert und mit einem Tabellenkalkulationsprogramm weiterbearbeitet werden. Dazu ist lediglich der Name für die Exportdatei anzugeben.*

### 6.1.6 Bugs & Troubleshooting

**Fehlerhafte Darstellung der Log-Meldungen:** Während der Arbeit mit Speedtrap kann es vorkommen, dass die Informationen des Logfensters teilweise im Hauptfenster erscheinen. Dabei handelt es sich um einen Fehler in der Anzeige der *JVM*[1], der durch die Auswahl des

[1] Java-Virtual-Machine

Kontextmenüpunktes „Ansicht aktualisieren" im Kontextmenü des Netzplanfensters behoben werden kann.

**Keine Warnung bei nicht gespeicherten Daten:** In der aktuellen Version von *Speedtrap* fehlen Warnmeldungen, falls Änderungen noch nicht gespeichert wurden. Es wird daher empfohlen, in regelmäßigen Abständen den Netzplan zu speichern, insbesondere vor dem Beenden des Programms.

## 6.2 Administrationshandbuch

Das *Administrationshandbuch* richtet sich vor allem an die Administratoren, die für die Installation, Konfiguration und Wartung von Speedtrap zuständig sind. Nach einer kurzen Einführung wird auf die Installationshinweise von Monitor,Agenten, und NIC sowohl unter Unix als auch unter Windows eingegangen. Außerdem werden die Schnittstellen zwischen den bedien Softwarekomponenten (Monitor und Agent) ausführlich beschrieben und erklärt.

In diesem Handbuch werden Kenntnisse der folgenden Bereiche vorausgesetzt:

- Netzwerkkenntnisse
- Kenntnisse über die Installation von Programmen und ein sicherer Umgang mit dem jeweiligen Dateisystem

Speedtrap ist eine verteilte Netzwerk Anwendung, mit deren Hilfe Messungen (auch) in großen Computer Netzwerken durchgeführt werden können, um die Lastsituation in den einzelnen Bereichen des gegebenen Netzwerks abschätzen zu können. Speedtrap erleichtert dem Administrator die Aufgabe, lastbedingte Schwach- und Engstellen (sog. „Flaschenhälse") finden und beheben zu können. Speedtrap liefert die für die Netzwerk Administration wichtigen Management Daten.

### 6.2.1 Installationshinweise für den Monitor

Der Speedtrap Monitor ist die zentrale Komponente von Speedtrap. Mit dem Monitor werden die Messungen durchgeführt und ausgewertet. Der Monitor wird nur auf einem Rechner im Netzwerk installiert. Es ist unbedingt darauf zu achten, dass im Netzwerk nur ein Monitor zur gleichen Zeit aktiv ist. Um den Monitor starten zu können, muss auf dem Rechner mindestens die Version 1.4.1 der *JVM* installiert sein. Sofern diese richtig in der Pfad-Umgebungsvariablen definiert ist, ist für die Installation des Monitors nichts weiter zu tun, als die Datei `Speedtrap.jar` auf den Rechner zu kopieren. Unter Windows genügt dann ein Doppelklick auf die JAR-Datei, um das Programm zu starten. Unter Linux lautet der Programmaufruf wie folgt:

```
[Java-Pfad/]java -jar [Speedtrap-Pfad/]Speedtrap.jar
```

Gegebenenfalls muss der Aufruf noch um die spezifischen Pfadangaben ergänzt werden. Nähere Informationen zur Installation der benötigten *JVM* sind später später in diesem Kapitel zu finden.

*Anmerkung:* Nach der Installation können Sie den Monitor direkt starten und sich die günstigsten Rechner für die Installation der Agenten anzeigen lassen.

### 6.2.2 Installationshinweise für einen Agenten

Um Messungen in Computer Netzwerken durchführen zu können, müssen auf bestimmten Rechnern die Speedtrap Software Agenten installiert werden. Diese sind in der Lage, SNMP fähige Geräte (z. B. Switches) auszulesen. Die Agenten senden die so gewonnenen Managementdaten an den Speedtrap Monitor, wo diese dann ausgewertet und dargestellt werden können. Auf welchen Rechnern eine Installation von Agenten für die Messungen günstig ist, wird vom Speedtrap Monitor vorgeschlagen (siehe Benutzungshandbuch). Die Agenten können aber prinzipiell auf jedem Rechner installiert werden.

#### Installation als Dienst unter Windows NT, 2000 und XP

Bei der automatischen Installation des Agenten werden die Java-Umgebung und die ausführbare Agenten-Jar-Datei auf das Ziellaufwerk kopiert und als Dienst installiert. Dazu wird die Installations-CD benötigt. Auf dieser CD befindet sich eine Datei „install-agent.bat", die ausgeführt werden muss. Für die Ausführung werden Administrationsrechte benötigt, da sich das Programm bei Windows registriert. Während der Installation wird gefragt, ob er einen Eintrag in der Windows Registry vornehmen soll. Dieses muss bejaht werden. Standardmäßig wird der Agent ins Verzeichnis „c:\daten\agent" kopiert, wo er dann nicht mehr verschoben werden sollte, da er als Dienst in Windows eingetragen wird. Die Batch-Datei „install-agent.bat" kann den Wünschen entprechend angepasst werden, wo Ziel-Verzeichnis und benötigte Java-Umgebung geändert werden können. Dies wird aber nur für erfahrene Benutzer empfohlen.

Nun kann der Dienst gestartet werden. Unter Windows XP klickt man dazu auf Start und Run (bei der deutschen Version auf Ausführen). Es öffnet sich ein Fenster in dem „services.msc" eingeben und bestätigt wird. Danach öffnet sich folgendes Fenster:

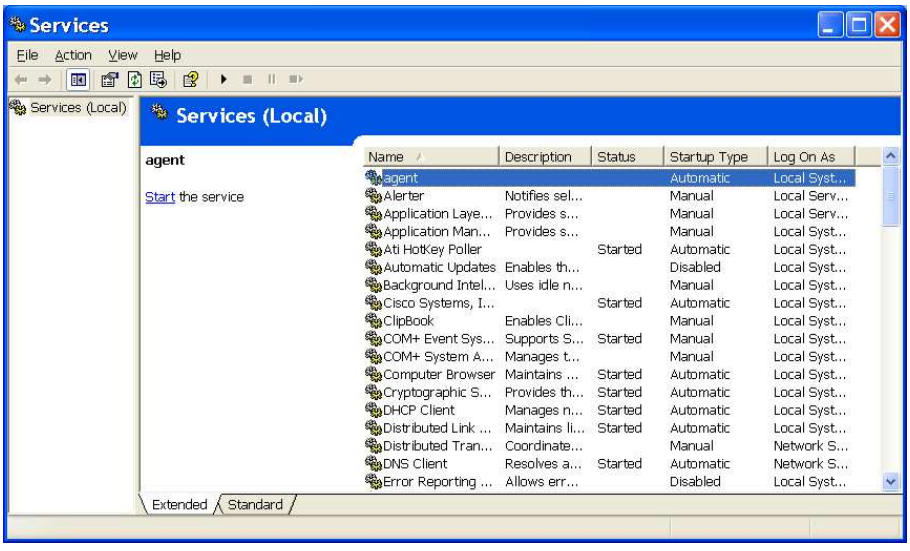



Abbildung 6.10: Liste der Dienste in Windows XP

Hier wählt man nun „agent" aus und klickt auf Start. Der Agent wird bei jedem Systemstart automatisch mitgestartet. Ist dies nicht erwünscht, kann man das auch unterbinden, indem man mit der rechten Maustaste auf „agent" klickt, Properties (Eigenschaften) auswählt und „startup type" auf manuell setzt.

#### Manuelle Installation unter Windows 9x, NT, 2000, XP

Der Agent kann auch ohne Administrationsrechte gestartet werden. Hierfür wird eine Java-Umgebung benötigt. Falls schon eine auf dem jeweiligen Rechner vorhanden ist, brauchen nur die Dateien „agent.jar" und „Run_Agent.bat" in ein beliebiges Verzeichnis kopiert. Danach kann der Agent mit "Run_Agent.bat" gestartet werden. Falls keine Java-Umgebung existiert, muss diese vorher installiert werden. Dies wird dadurch erreicht, dass die Datei „j2sdk-1_4_1_01-windows-i586.exe" im Verzeichnis „Java-Umgebung" auf der Installations-CD ausgeführt wird.

#### Manuelle Installation unter Linux und allen Unix-Derivaten

Es wird davon ausgegangen, dass eine lauffähige Java-Version vorinstalliert ist. Zur Ausführung des Agenten wird nur die Datei „agent.jar" benötigt. Diese sollte in ein beliebiges Verzeichnis kopiert werden. Danach kann der Agent mit „java -jar agent.jar" gestartet werden.

### 6.2.3 Installationshinweise für ein NIC-Backend

Falls das Netzwerk keine SNMP fähigen Geräte oder Rechner aufweist, die für die Messungen durch die Agenten in Frage kommen, ist es möglich, dass der Agent seine Messungen lokal an der Netzwerkkarte (ETH0) durchführt. Dieses Verfahren hat jedoch den Nachteil, dass einige Messdaten nicht erhoben werden können, die bei einer Messung mit SNMP zur Verfügung stehen. Wir empfehlen daher, dieses Verfahren anzuwenden, wenn die Installation eines SNMP Dienstes nicht möglich oder unerwünscht ist. Um direkt an der Netzwerkkarte messen zu können, müssen zuvor weitere Komponenten installiert werden, bevor anschliessend der Agent installiert werden muss:

#### NIC-Installation unter Windows 9x, NT, 2000, XP

Falls keine Java-Umgebung auf dem Zielrechner installiert ist, muss diese vorher installiert werden. Dies wird dadurch erreicht, dass die Datei „j2sdk-1_4_1_01-windows-i586.exe" im Verzeichnis „Java-Umgebung" auf der Installations-CD ausgeführt wird. Danach muss WinPcap installiert werden. Die zugehörige Installationsdatei „WinPcap_3_01.exe" befindet sich im Verzeichnis „Nic Installation" auf der Installations-CD. Als nächstes müssen einige Jpcap-Dateien, welche sich auch im „Nic Installation"-Verzeichnis befinden, ins Java-Verzeichnis kopiert werden. Das Java-Verzeichnis befindet sich normalerweise unter „C:\j2sdk\". „Jpcap.dll" wird ins „[Java-Verzeichnis]\jre\bin" und „jpcap.jar" nach „[Java-Verzeichnis]\jre\lib\ext" kopiert.

#### NIC-Installation unter Linux

Falls keine Java-Umgebung auf dem Zielrechner installiert ist, muss diese vorher installiert werden. Die Installationsdatei „j2sdk-1_4_1_01-linux-i586-rpm.bin" für das Linux-Betriebssystem liegt im Verzeichnis „Java Umgebung" auf der Installations-CD. Danach muss libpcap installiert werden. Die zugehörige Installationsdatei „libpcap-0.7.1.tar.gz" befindet sich im Verzeichnis „Nic Installation" auf der Installations-CD. Als nächstes müssen einige Jpcap-Dateien, welche sich auch im „Nic Installation"-Verzeichnis befinden, ins Java-Verzeichnis kopiert werden. „libjpcap.so" wird ins „[Java-Verzeichnis]/jre/lib/i386" oder [Java-Verzeichnis]/jre/lib/sparc und „jpcap.jar" nach „[Java-Verzeichnis]/jre/lib/ext" kopiert.

### 6.2.4 Schnittstellen

Die Kommunikation zwischen den beiden Softwarekomponenten (Monitor und Agent) erfolgt mittels UDP und einem besonderen, von uns selbst entworfenen Kommunikationsprotokoll. Statt UDP wird üblicherweise TCP zur Kommunikation verwendet auf das wir verzichten, um den Kommunikationsoverhead möglichst gering zu halten, um die gemessenen Werte nicht unnötig zu verfälschen. Während der Monitor die Agenten kontrolliert und die Anweisungen des Benutzers an sie weiterleitet, können die Agenten die empfangenen Kommandos ausführen und entsprechend antworten. Durch die Vielfalt der Möglichkeiten, die Speedtrap bietet, ist ein aussagekräftiges, aber dennoch „schlankes" Kommunikationsprotokoll für beide Komponenten von entscheidender Bedeutung. Ein Kommunikationsstring beginnt immer mit einem ‚*'-Symbol, um das Risiko abzufangen, dass eine Nachricht, die fälschlicherweise an den Agenten gesandt wurde (z.B. von einer anderen Anwendung) nicht irrtümlich als Kommando interpretiert wird. Dann folgt die zweistellige Sektionsnummer, die die Art der Nachricht identifiziert (siehe unten), gefolgt von einer zweistelligen Kommunikationsnummer, deren Aussagekraft von der Sektion abhängig ist. Nun folgt entweder eine ‚#', oder ein ‚$'-Zeichen. Mit der ‚#' schließt eine Nachricht immer ab, das ‚$'-Symbol zeigt an, dass noch weitere Informationen folgen, die dann (einige Zeichen später) mit dem ‚#'-Symbol abgeschlossen werden.

Dieses Kommunikationsprotokoll gliedert sich in verschiedene Teilbereiche, die Sektionen genannt werden:

#### ‚01' Sektion: Statusabfrage

Erlaubt dem Monitor, den aktuellen Status des bzw. der Agenten zu überprüfen. Eine Statusabfrage vom Monitor an den Agenten sieht folgendermaßen aus: ***0100#**

Die Antwort des so angesprochenen Agenten ist nun von seinem inneren Status abhängig:

***0100#** - System bereit, Agent wartet auf Anweisungen.

***0110#** - Messung läuft, der Agent führt gerade eine Messung durch.

***0120#** - Lastgenerierung läuft, der Agent führt gerade eine Lastgenerierung durch.

***0190#** - ERROR, Switchport nicht erreichbar, der Agent wäre zwar prinzipiell bereit, aber er kann das ihm zugewiesene Switch nicht erreichen. Das kann z.B. daran liegen, dass noch kein Switch zugewiesen wurde, oder dass der Switch nicht erreichbar, bzw. ausgefallen ist, oder das Passwort (Community String) nicht richtig angegeben wurde. Wenn dieser Agent kein Switch überwacht, sondern lokal am SNMP Server messen soll, ist diese Meldung ein Hinweis darauf, dass der lokale SNMP Dienst nicht installiert oder deaktiviert ist. In diesem Fall kann der Agent nur noch an der lokalen Netzwerkkarte messen.

#### ‚02' Sektion: Konfiguration

Erlaubt dem Monitor, die aktuelle Konfigurationssituation des angesprochenen Agenten zu überprüfen oder zu ändern. Eine Konfigurationsabfrage sieht folgendermaßen aus:
***0200#**

Der Agent antwortet dann folgendermaßen:

***0200$AgentIP$SwitchIP#** - Der Agent übermittelt also seine eigene IP-Adresse und die Adresse des ihm zugewiesenen Switches. So kann der Monitor überprüfen, ob eine zuvor geschehene Zuweisung eines Switches zu dem Agenten korrekt übernommen wurde. Diese Sektion erlaubt dem Monitor auch eine Änderung der Konfiguration.

Die Zuweisung eines neuen Switches an den Agenten erfolgt folgendermaßen:

***0201$SwitchIP$Community String#** - Der Monitor kann mit diesem Kommando dem Agenten seinen Switch und den zugehörigen community string vermitteln.

Der Agent antwortet dann:

***0201$AgentIP$SwitchIP#** - Bestätigung der Konfigurationsänderung

Eine weitere wichtige Konfigurationsänderung stellt die Zeitsynchronisation dar. Da der Agent nicht (zwangsläufig) über die gleiche Systemzeit verfügt, wie der Monitor, muss der Monitor regelmäßig die aktuelle Zeit an den Agenten übermitteln, damit die Messungen zeitlich genau durchgeführt werden. Hierfür wird als Zeitformat UNIXTIME gewählt (Anzahl der Millisekunden seit 1970 als long-Datentyp). Der Agent berechnet dann die Differenz der übertragenen Zeit und der eigenen und berücksichtigt diese dann bei allen zeitkritischen Aktionen. Der Monitor übermittelt die Zeit folgendermaßen:

***0207$UNIXTIME#** - Die aktuelle Systemzeit des Monitors wird an den Agenten übergeben. Eine Antwort des Agenten auf eine Zeitsynchronisation vom Monitor ist nicht notwendig, da die Synchronisationsintervalle genügend kurz gewählt sind.

Eine weitere wichtige Kommunikationsoption seitens des Monitors ist der sog. MAC Server. Diese zum Netzplan (des Monitors) zugehörige Komponente ist Teil jedes Agenten. Der MAC Server beantwortet Anfragen des Monitors bzgl. der Position des Agenten im Netzwerk. Eine solche Anfrage sieht folgendermaßen aus:
***0206#**

Der Agent antwortet dann:

***0206$SwitchIP#** - Der Agent gibt die IP des Switches zurück, das er kontrolliert.

#### ‚03' Sektion: Messauftrag

Ein Agent hat primär die Aufgabe, Messungen durchzuführen und die Messwerte lokal in einem Speicher zu halten, bis sie vom Monitor „abgeholt" werden. Wenn der Messauftrag seitens des Monitors ergangen ist, wird der Agent sie zum angegebenen Zeitpunkt starten und die Messung dann so lange durchführen, wie in der Messdauer angegeben ist. Die Messwerte werden während der Messung nach kurzen Zeitintervallen abgerufen, z.B. alle 10 Sekunden. Da i. d. R. ein Switch „überwacht" wird, müssen während eines Intervalls die Daten zu allen Ports (Anschlüssen) abgerufen und gespeichert werden. Zudem müssen für jeden Port verschiedene Messwert Typen „gesammelt" werden, z.B. die Auslastung oder die übertragenen Bits. Außerdem gibt es auch solche Messwert Typen, die nicht für die Ports im Einzelnen gelten, sondern für das gesamte Switch. Wenn der Monitor nun eine Messung in Auftrag geben soll, übermittelt er folgendes an den jeweiligen Agenten:

***0301$Startzeit$Dauer$Intervall#** - Alle drei Werte sind in Millisekunden angegeben und die Startzeit ist im UNIXTIME Format.

Der Agent ermittelt nun, ob er bei der gegenwärtigen Konfiguration eine Messung starten könnte. Hierzu versucht er zuerst, an dem ihm zugewiesenen Switch zu messen. Ist dies nicht möglich, versucht er eine Messung beim lokalen SNMP Dienst. Ist auch dies nicht möglich, versucht der Agent, an der Netzwerkkarte zu messen (NIC). Falls nun ach hier keine Messung möglich ist, weil die Netzwerkkarte z.B. gesperrt ist, sendet er eine Fehlermeldung zurück:

***0399#** - ERROR. Die Messung konnte nicht gestartet werden.

Im anderen Fall (also falls eine Messung möglich ist) sendet der Agent die Bestätigungsnachricht:

***034242#**

an den Monitor zurück. In diesem Fall ist die Messung angenommen und kann zum angegebenen Zeitpunkt durchgeführt werden. Der Agent startet dann die Messung wie vorgesehen. Manchmal kann es notwendig sein, eine in Auftrag gegebene Messung wieder abzubrechen. In diesem Fall sendet der Monitor folgendes Kommando:

***0300#** - Messung soll abgebrochen werden

In diesem Fall stoppt der Agent die Messung oder terminiert den Messauftrag, falls die Messung noch nicht begonnen hat. Das Stoppen einer Messung kann nicht schief gehen (außer der Agent stürzt ab), sodass immer folgende Bestätigungsnachricht gesendet wird, nachdem die Messung abgebrochen wurde:

***0300#** - Messung wurde gestoppt

### ‚04' Sektion: Daten abholen

Nachdem nun eine Messung erfolgreich durchgeführt wurde, liegen die Messdaten lokal beim Agenten vor. Da der Agent nur reagiert und nie irgendwelche Aktionen eigenmächtig startet, ist es notwendig, dass der Monitor die Messdaten anfordert. Nach einer solchen Anforderung sendet der Agent die Daten an den Monitor, der diese dann auswerten und darstellen kann. Da in der Kommunikation zwischen Monitor und Agent aus o. g. Gründen grundsätzlich UDP verwendet wird, ist die Länge der zu übermittelnden Nachrichten auf 64kByte pro Datenpaket begrenzt. Leider fallen aber bei den Messungen insgesamt relativ große Datenmengen an, die diesen Rahmen sprengen und es ist somit notwendig, mehrere Pakete zu senden. Da UDP als verbindungsloses Netzwerk Kommunikationsprotokoll nicht auf die korrekte Reihenfolge der Pakete achtet, werden die Pakete „durchnummeriert". Um das zu gewährleisten, enden die Nachrichten solcher Pakete, die noch Folgepakete nach sich ziehen mit einem ‚&Paketnummer&', und das letzte Paket zu einer Messreihe mit ‚&Paketnummer#'. Des Weiteren enthalten einzelne Pakete immer nur ganze Intervalle, d.h. kein Intervall wird aufgespalten. Die gerade beschriebene Lösung ist zwar realisiert, kommt jedoch im Betrieb von Speedtrap so gut wie nie zur Anwendung, da die Daten auch selektiv zurückgesendet werden können. Daher soll nur das Verfahren der selektiven Datenrücksendung hier genauer illustriert werden; Das o. g. Verfahren ist zu diesem bis auf die angesprochenen Modifikationen bzgl. der Unzulänglichkeiten des UDP- Protokolls analog. Der eigentliche Gewinn des Verfahrens zur selektiven Rücksendung der Messdaten liegt darin, dass nur die Daten für bestimmte (vom Monitor in der Anfrage angegebenen) Intervalle gesendet werden müssen. Dadurch verringert sich natürlich die Menge der zu übertragenden Daten beträchtlich. Dieses Verfahren hat aber auch den Vorteil, dass die Messdaten bereits während der Messung abgerufen werden können, wodurch auch während der Messung bereits einige Ergebnisse vorliegen. Dadurch ist die „live" Darstellung bei Speedtrap möglich und ein Abrufen der Daten nach der ersten Methode ist für fast alle Fälle unnötig. Ein weiterer wichtiger Punkt ist der Messdaten Typ. Beim Abholen der Messdaten muss der Typ explizit angegeben werden. Prinzipiell könnte man auch einige oder alle Messwert Typen gleichzeitig übertragen, was allerdings zu Lasten des wertvollen Platzes in den UDP beschränkten Paketen gehen würde, was mit diesem Verfahren vermieden werden sollte. Falls der Monitor nun die Messdaten abholen will, sendet er folgende Nachricht an den Agenten:

***04XY$Startintervall$Endintervall#** - Das XY steht für den Messwert Typ und wird weiter unten genauer entschlüsselt, Startintervall und Endintervall ist in Form von Zahlen angegeben, die das entsprechende Intervall kennzeichnen (so ist z.B. ‚7' als das siebte Intervall zu sehen).

Der Agent antwortet dann:

***04XY%Startintervall$Endintervall%$[Wert für Port 1]$[Wert für Port 2]$[Wert für Port 3]$ ... $%$[Wert für Port 1]$[Wert für Port 2]$ ... $%$ ... $#**

Das *04... bedeutet, dass jetzt Messdaten übertragen werden. Das XY bezeichnet die Messart (XY steht für eine Zahl, die den Messwert Typ codiert). Danach werden Start- und Endintervall noch mal zurückgegeben, damit bei einer Durchmischung der gesendeten Pakete keine Verwechselungen auftreten können. Dann folgt ein ‚%', welches anzeigt, dass jetzt die Daten für das erste Messintervall kommen. Nun folgen die Messwerte für die einzelnen Ports (ohne eckige Klammern), welche in ‚$'-Zeichen eingebettet sind. Nach dem letzten Port kommt wieder ein ‚%'-Zeichen. Dieses deutet an, dass jetzt die Messdaten für alle Ports im Startintervall übertragen worden sind. Nach dem ‚%' folgen (bis zum nächsten ‚%') die Messdaten für alle Ports aus dem folgenden Intervall. Diese schließen auch wieder mit einem ‚%' ab. Nach diesem ‚%' folgen die Werte für alle Ports aus dem 3. Intervall, u. s. w.. Wenn nun alle Intervalle (vom Start- bis zum Endintervall) behandelt sind, kommt am Ende die gewohnte ‚#'. Diese symbolisiert das Ende der übertragenen Nachricht. Pakete, die aufgrund der Unzulänglichkeiten von UDP nicht beim Monitor angekommen sind, werden einfach noch einmal angefordert.

### ‚05' Sektion: Lastgenerator

Für zahlreiche Anwendungen des Speedtrap Frameworks ist die Generierung von Lasten von zentraler Bedeutung. Der Lastgenerator bietet dem Agenten die Möglichkeit, künstliche Lastsituationen im Netzwerk zu erzeugen, die von anderen Agenten dann gemessen werden können. Hierbei werden UDP Pakete erzeugt, die für eine bestimmte Dauer in den durch Intervalle definierten Zeiträumen an einen Zielhost gesendet werden. Der Benutzer hat auch die Möglichkeit die Größe dieser Pakete anzugeben. Der Agent, der die Last erzeugen soll, füllt die Pakete dann in der angegebenen Größe mit sinnlosen Zeichen, damit die Pakete von der Netzwerkkarte des Zielhosts verworfen werden. Damit hat man eine reale Lastsituation künstlich erzeugt. Falls nun der Monitor eine Last generieren will, sendet er folgende Nachricht an den Agenten:

***0501$ZielIP$Startzeit$Dauer$Intervall$Paketgröße in Byte#** - ein Lastauftrag wird an den Agenten gesendet.

Der Agent antwortet dann:

***0500#** - Lastauftrag angenommen und bestätigt.

Die ZielIP ist im allgemein üblichen (oktalen) Format anzugeben, also etwa: 192.168.0.22 Die Startzeit, Dauer und das Intervall sind in Millisekunden anzugeben, Startzeit im UNIXTIME Format. Die Paketgröße darf nicht kleiner sein als 10 Byte, da der Header eines UDP Pakets bereits bei 8 Byte liegt, maximal darf die Größe eines Pakets bei 64kByte liegen. Der Agent wartet dann bis zur Startzeit und sendet dann in jedem Intervall ein Paket in der angegebenen Größe. Beim Lastgenerator ist es sehr wichtig, dass die Erzeugung der Lasten gestoppt werden kann, da sich ansonsten in Netzwerken Horrorszenarien ereignen können. Um eine Lastgenerierung zu stoppen, muss der Monitor folgendes Kommando senden:

***0500#** - Stoppe Lastgenerierung

Der Agent antwortet dann:

***0500#** - Lastgenerierung gestoppt.

Das Stoppen der Lastgenerierung kann nicht schief gehen (außer im Falle eines Absturzes), daher gibt es auch hier keine ERROR Meldung. )

# Kapitel 7

# Fazit der PG Mitglieder

Abschließend haben in diesem Kapitel alle Beteiligten die Möglichkeit, in einem kleinen Fazit ihre Eindrücke aus den letzten beiden Semestern zu schildern. Die Texte wurden in der letzten Woche vor dem Fachgespräch geschrieben, als bereits alle Ziele außer dem Fachgespräch und Endbericht erreicht waren.

## 7.1 Babak Davani

Diese PG hat gezeigt, dass sich reale Arbeitswelt und Universität nicht unbedingt widersprechen. Es hat die Höhen und Tiefen der Arbeitsgruppe, die es später auf der Arbeit geben wird, gut wiedergegeben. Mir hat die Aufgabe, als einer der Projektmanager zu operieren, sehr zugesagt und dabei hat es auch viel an Erfahrungsgewinn mit sich gebracht. Unter anderem konnte ich feststellen, inwiefern das angewendete Vorgehensmodell für unsere PG geeignet war, welche Auswirkungen Verzögerungen bestimmter Tasks auf das Projekt hatten und wie wichtig eine gute Aufgabenverteilung an den Teammitgliedern ist.

Dass Termine nicht eingehalten werden konnten, gehört natürlich genauso dazu, die Realität im späteren Berufsleben wiederzuspiegeln, genauso wie es Konflikte unter den Projektmitgliedern gegeben hat. Ich würde das nicht unbedingt als was Negatives auffassen, sondern als Bereicherung der ersten "Berufserfahrung" ansehen. Es hat sich dargestellt, wie Terminüberschreitungen das ganze Projekt in Verzug bringen können, wenn es gerade die Aufgaben sind, die nicht parallel abgearbeitet werden können, wie zum Beispiel zwischen der Programmierung und der Testeinsätze. Dass Fristüberschreitungen in der Arbeitswelt völlig normal sind, kann ich aufgrund meiner persönlichen Erfahrung bestätigen. Diese Probleme waren aber häufig nicht aufgrund fehlender Planung, "mangelnder Kompetenz" oder Faulheit der Mitglieder zu sehen, sondern meistens auch aufgrund nachlassender Motivation. Es ist nicht gerade schön in einer Atmosphäre zu arbeiten, wenn die Arbeiten der Mitglieder meistens als selbstverständlich hingenommen, ständig kritisiert, oder gar als absolut schlecht abgestempelt werden. Es hätte die Stimmung um einiges verbessert, wenn auch mal Lob ausgesprochen worden wäre. Meiner persönlichen Meinung nach finde ich, dass wir ein sehr gutes Produkt entwickelt haben, das auf dem Markt sicherlich konkurrenzfähig zu anderen Produkten der gleichen Kategorie der Netzwerk-Anlayse-Tools ist.

Ich denke, dass wir die Konflikte untereinander gut lösen konnten und die Projektmitglieder im Großen und Ganzen gut zusammengearbeitet haben. Besonders bei der Zusammenarbeit der einzelnen Arbeitsgruppen gab es nichts zu beanstanden, da die vorher abgemachten Schnittstellen gepasst haben. Schwierigkeiten gab es nur am Ende der Implementierungsphase, wo sich noch einige erheblich Bugs im Programm befanden, die dazu führten, dass die Termine der Testeinsätze verschoben werden mussten.

Wenn ich wieder eine PG machen müsste, würde ich wieder mit allen unseren Projektmitgliedern zusammenarbeiten wollen. Hier möchte ich Christian meinen Dank aussprechen, der besonders viel für unsere PG geleistet hat. Unter anderem hat er viel Fleißarbeit bei der Programmierung als auch bei der Erstellung des Zwischen- und Endberichts erbracht.

## 7.2 Hayat Elfassi

Mir hat die Projektgruppe viel gebracht. In diesen 2 Semestern habe ich viel Neues kennengelernt. Für mich persönlich haben nicht nur die informativen Vorträge wie z.B. über JavaSpaces, SNMP und Warteschlangentheorie etwas gebracht, sondern auch die Erfahrung Vorträge zu halten, ohne die anderen zu langweilen. Auch habe ich erfahren, wie wichtig es ist, in einem Team mit Teamgeist zu arbeiten. Ich bin sehr zufrieden an dieser PG teilgenommen zu haben, weil alle Teilnehmer interessante und nette Menschen sind.

## 7.3 Daniel Grabowski

Diese PG hat mir persönlich sehr viel gebracht. Ich habe u.a. die besonderen Vorzüge, aber auch die Probleme kennengelernt, die Gruppenarbeit mit sich bringt. Auch inhaltlich habe ich viel gelernt: Neben den Seminarthemen, die z.T. eine Vertiefung bereits vorhandenen Wissens mit sich brachten, habe ich auch viel neues gelernt. Außerdem konnte ich meine Programmierkenntnisse in Java festigen und erweitern.

## 7.4 Alexander Harsch

Die Projektgruppe ist für mich eine interessante und positive Erfahrung gewesen. Die Konstelation der Mitglieder war wohl eher ungewöhnlich für eine PG, die sich mit der Entwicklung von Software befasst, da nur die wenigsten sich intensiv mit Java oder Programmierung an sich, außerhalb des Software Praktikums, auseinander gesetzt haben. Daher kam es auch so, dass extrem viele „Überraschungen" auf uns zukamen, mit denen nicht gerechnet wurde. Vielleicht wäre der Ansatz des „Hard Codings" für uns eine besere Entscheidung gewesen. So wäre vielen schon früher klar geworden, wo Probleme auf uns zukommen. Insgesamt wurde aber doch von jedem in seinem Rahmen eine gute Arbeit abgeliefert. Schade war, das die Gruppe als Ganzes wenig zueinander gefunden hat. Die IRB ist ein weiterer Punkt der noch erwähnt werden sollte, war es doch eines unserer Ziele, den Datenfluss des von der IRB betreuten Netzwerks des LS4 zu messen. Die Verantwortlichen machten jedoch eine Zusammenarbeit völlig unmöglich, zudem legten manche Administratoren der IRB ein asoziales Verhalten an den Tag, wovon ich lediglich Gerd Sokolies ausschliessen möchte, der uns mit Rat und Tat zur Seite stand.

## 7.5 Ulrich Hilmer

Die Projektgruppe war auch für mich eine interessante Erfahrung, auch wenn es nicht meine "Erstwunsch-PG" gewesen ist. Das erste Seminar, dass wir ausserhalb der Universität durchgeführt haben, hat neben den Aufwerfen neuer fachlicher Aspekte auch den Vorteil, dass die Gruppe sich ein bischen besser kennen lernen konnte.
Im Gegensatz zu vielen anderen PGs wurde bei uns erstaunlicherweise nicht endlos über jeden noch so kleinen Aspekt rumdiskutiert sondern schnell, aber konstruktiv Lösungsansätze für die anstehenden Probleme gesucht. Andererseits war aber die Einhaltung der selbstgesteckten Termine nicht unbedingt eine Stärke unserer PG.

Dabei haben wir alle erdenklichen Tools, die die Gruppenarbeit erleichtern sollten, ausprobiert: Message Boards, IRC- Channel, Groupware und Instant Messaging. Die Relevanz solcher Mittel sollte also nicht überschätzt werden. Keines davon wurde von der Gruppe richtig angenommen und letztlich waren die regelmäßigen Sitzungen, zu denen auch immer alle anwesend waren, der zentrale Dreh- und Angelpunkt.
Die eigentliche Programmierung von Speedtrap verlief im Vergleich zu anderen Tasks ziemlich im Zeitplan. Die benötigten Arbeitsmittel waren vorhanden (schnelle PCs im PG Pool, ein SNMP-fähiger Switch, CVS, Druckquota). Speedtrap ist allerdings ein Tool, dass die Sicherheit und den Datenschutz von IT-Anlagen berührt. Und aus dieser Richtung taten sich nicht zu erahnende Schwierigkeiten auf: Die IRB, die die Hoheit über die Sun-Maschinen und die PG-Pool PCs besitzt, war nicht bereit, uns Zugriff auf die Hardware (die NIC) eines einzigen Rechners zu geben. Wir sind dabei der IRB so weit entgegengekommen, wie es uns möglich war, in der Hoffnung, dass die IRB uns wenigstens einen abfragbaren SNMP-Daemon installiert, den wir dann wiederum abfragen können. Besonders für die PG-Mitglieder, die das NIC-Backend (zu Hause) entwickelt haben, war dies eine große Enttäuschung. Ausdrücklich ausnehmen möchte ich allerdings Gerd Sokolies, der die PG nach Kräften unterstützt hat, soweit es sein Resort betroffen hat.
Im Vergleich dazu war die Zusammenarbeit mit der Kreispolizeibehörde Unna vollkommen unproblematisch. Hier gab es vollen Zugriff auf die Switche und Rechner, obwohl hier sehr viel "heissere" Daten durchs Netz gehen, als am LS4. Auch war immer ein Ansprechpartner vor Ort.

Ich glaube, mit dem erreichten Ergebnis kann die PG zufrieden sein. Speedtrap ist ein Tool, dass sich an den SNMP Standard hält, einen großen Funktionsumfang bietet und Ergebnisse ansprechend graphisch präsentiert. Nebenbei haben wir alle viel über Netzwerke, Leistungsbewertung und Messungen im LAN gelernt.

## 7.6 Chiagozie Madukife

Diese Projektgruppe ist für mich ein Beweis dafür, dass verschiedene Leute aus unterschiedlichsten Hintergründen sich zusammenfinden können und trotzdem gut zusammenarbeiten; Stichwort „Multi-Kulti". Ich habe insgesamt sechs verschiedene Länder in drei Kontinenten die in irgendeiner Art und Weise etwas zu diesem „Cocktail" beigetragen haben gezählt. Noch interessanter ist die Tatsache, dass fast keine zwei Leute unter uns sich vorher kannten.

Das Einführungsseminar hat meiner Meinung nach viel dazu beigetragen, dass wir uns schnell kennenlernten. Wie im normalen Leben haben sich manche Leute viel näher kennengelernt und die anderen weniger. Die Zusammenarbeit in kleineren Gruppen innerhalb der PG hat Wunder bewirkt. Die ersten Gruppen, die für die Praktika gebildet wurden haben sich abgesehen von geringfügigen Änderungenn bis zum Ende bewährt (Why change a winning team?). Das habe ich als sehr positiv empfunden, auch wenn man meinen könnte, dass das der Grund sei, warum die Zusammenarbeit in der ganzen Gruppe nicht optimal war. Ich bin ziemlich zufrieden mit der Zusammenarbeit! Ich werde sicherlich mit manchen PG-Mitgliedern nach der endgültigen Beendigung der PG in Kontakt bleiben.

Meine Teilnahme an dieser Projektgruppe hat meine Kenntnisse in Bereich Computernetzwerke und Simulation sehr stark bereichert. Ausserdem hat mir die PG die Gelengenheit gegeben, die „Kunst" der Programmierung endlich zu erproben und hat mir die Angst davor endgültig genommen. Das ist für mich persönlich ein großer Pluspunkt.

Wenn mir etwas in dieser PG gefehlt hat, ist das sicherlich, dass wir nie vor möglichen Gefahren früh genug gewarnt wurden. Jedoch haben wir aus Fehlern gelernt und das ist auch gut so. Die verschiedenen Probleme denen wir im Laufe des Projekts begegnet sind, sind für mich nichts anderes, als die „Simulation" des wirklichen Lebens. Wir werden sicherlich ähnlichen Problemen in unserem Berufsleben begegnen, daher finde ich die gar nicht so schlimm, schließlich sind wir jetzt gut gewappnet, um mit solchen Problemen in der Zukunft fertig zu werden.

Was ich hier insgesamt gelernt habe wird mich mein ganzes Leben begleiten, sowohl bei zwischenmenschlichen Beziehungen als auch in meinem Berufsleben. Das Projekt *Speedtrap* hat viel hinter sich und wurde am Ende pünktlich und erfolgreich abgeschlossen. Ich bin wirklich stolz ein Teil dieses Projekts zu sein. Ich kann mir es wirklich vorstellen in Zukunft *Speedtrap* weiter zu entwickeln. Diese PG ist eine von den wenigen Lehrveranstaltungen im Informatikstudium, die meine Begeisterung bis zum Ende erhalten hat. Ich möchte hier mit einem Lob an uns alle abschließen, denn *Speedtrap* ist ein dynamisches Tool geworden, das alle ihm gestellten Aufgaben erfüllt. Ich gratuliere allen Teilnehmern dieser PG und wünsche euch allen noch ein erfolgreiches Studium und ein zufriedenstellendes Berufsleben.

## 7.7 Jürgen Mäter

Die Aufgabenstellung „Entwicklung eines Java-basierten Frameworks für verteilte Netzwerkmessungen" erfordert Kenntnisse sowohl über Java und objektorientierte Entwicklungsumgebungen als auch über Leistungsbewertungstechniken, insbesondere über Messungen.

Zwei interessante Einsätze des Frameworks waren vorgesehen, nämlich Engpassanalysen im lokalen Netz des Lehrstuhls Informatik 4 bei der verteilten Berechnung von Markov-Ketten und im Rechnernetz der Kreispolizeibehörde Unna, mit der bereits die Projektgruppen 336 und 358 sehr gut zusammengearbeitet haben.

Dem überwiegenden Teil der PG-Mitglieder waren Java und Entwicklungsumgebungen zumindest aus dem Softwarepraktikum bekannt während Leistungsbewertungstechniken völliges Neuland darstellten. Aus diesem Grund stammten die Themen des ersten Blockseminars schwerpunktmäßig aus dem Bereich der Leistungsbewertung (9 Vorträge). Als Ergänzungen dienten Themen aus dem Projektmanagement (2) und Rechnernetzen (1). Nach Sichtung der ersten Literaturstellen entschlossen sich 2 Teilnehmer, die Projektgruppe zu verlassen, sodass aus dem Bereich Leistungsbewertung der Vortrag über Experiment-Design und ein Vortrag aus dem Projektmanagement entfielen.

In dem extern veranstalteten Kompaktseminar kristallisierten sich die ersten Probleme hinsichtlich Pünktlichkeit, Sozialverhalten und Zusammenarbeit heraus. Die hier extern beginnende Kennenlernphase der PG-Mitglieder und unserer Partner hat meiner Meinung nach kaum Einfluss auf die weitergehende Projektarbeit gehabt. Die Vorträge waren inhaltlich bis auf einen gut bis sehr gut.

Der von den beiden gewählten Projektmanagern ausgearbeitete Projektplan für das erste PG-Semester fand gleich meine Zustimmung. Bei der Durchführung des Planes sind jedoch erhebliche Terminüberschreitungen aufgetreten, weil es weder den Projektmanagern noch mir gelungen ist, die Gruppe derart zu motivieren, die Aufgaben pünktlich zu erledigen. Über eine zentral eingerichtete ToDo-Liste und ein Messageboard wurde zwar lange diskutiert, jedoch wurde deren Vorhandensein ignoriert. Mein Eindruck war, dass einige Mitglieder nicht den richtigen Einsatz zeigten und ausloten wollten mit welchen Minimalleistungen sie durchkommen?

Aufbauend auf dem Blockseminar, sollten in einem Meß- und Modellierungspraktikum die Grundlagen der Messung und Modellierung in der Leistungsbewertung für Rechensysteme vertieft werden. Obwohl die Modellierungsaufgaben direkt von Vorgängerprojektgruppen, bzw. wörtlich aus der Basisliteratur für die Seminarvorträge entnommen wurden, haben die PG-Mitglieder sie erst nach mehrmaligen Rücksprachen und Korrekturen lösen können. Es hat sich hier klar herausgestellt, dass die PG-Mitglieder kein Interesse gezeigt haben, sich tiefer in das Gebiet einzuarbeiten. Das Meßpraktikum, dass aufgrund nicht erhaltener Zugriffsrechte sehr einfach ausfiel, wurde dagegen zur Zufriedenheit absolviert.

In den sich anschließenden Phasen Anforderungsdefinition, Analyse und Entwurf zeigte sich deutlich, dass die Teilnehmer mehr einzelkämpferisch als in Untergruppen zu arbeiten gewohnt waren. Insbesondere zeigte sich das darin, dass sich nicht auf eine einheitliche Software-Entwicklungs-Umgebung geeinigt werden konnte. Jeder arbeitete mit seiner Lieblingsumgebung! Das Ergebnis

war entsprechend und erforderte erhebliche Nacharbeiten. Da die verwendeten Tools zum größten Teil reine Implementierungs-Umgebungen boten, fällt demzufolge die Gesamtdokumentation sehr dürftig aus, z.B. unterstützen sie kein Round-Trip-Engineering, so dass Strukturmodelle des Entwurfs nicht automatisch nachgepflegt wurden. Auf mein Drängen hin, akzeptierte die Gruppe zwar ein Vorgehensmodell, das durch projektspezifische Anpassungen aus dem SoPra-Vorgehensmodell konstruiert wurde, richtete sich aber nicht streng danach.

Der Zwischenbericht, in dem die Ergebnisse des ersten Semesters bis hin zum Entwurf verzeichnet sind, benötigte auch zu viele Iterationen bis er letztendlich verspätet akzeptiert werden konnte.

Das zweite Semester begann mit einem Spezialisierungsseminar. Auch hier wurde in einem Fall wieder versucht, sich die Arbeit so einfach wie möglich zu machen, indem der Inhalt des Vortrages innerhalb des vorzutragenden Gebietes so verschoben wurde, dass er sich auf Trivialitäten beschränkte und von dem eigentlichen Problem nichts beinhaltete.

Für den Rest des Semesters wurde Besserung gelobt, was sich aber bereits nach zwei Wochen wieder legte. Die Implementierung erfolgte modulweise in Untergruppen. Da die Kommunikation zwischen den Untergruppen schlecht funktionierte, wurde relativ spät erkannt, dass feste Termine, an denen möglichst alle Mitglieder im Pool zusammen arbeiten, vielleicht eine Abhilfe bringen könnten.

Der von der Projektmanagement-Gruppe vorgeschlagene Zeitplan zur Planung und Durchführung des Testeinsatzes des entwickelten Frameworks fand zwar Zustimmung, überforderte aber offensichtlich die PG-Mitglieder. Es wurde ziemlich blauäugig darauf los programmiert, ohne an etwa auftretende Probleme zu denken, bzw. parallel zur Implementierung bereits den Testeinsatz zu planen und abzustimmen. Insbesondere wenn externe Einheiten, wie die Kreispolizeibehörde, die Informatik-Rechner-Betriebsgruppe oder der Lehrstuhl Informatik IV mit beteiligt sind, wäre eine vorausschauende frühzeitige Abstimmung sicher erfolgversprechender gewesen und einiges an Ärger und Frust hätte vermieden werden können.

Das Produkt Speedtrap entspricht leider nicht meinen Erwartungen. Von einer Mannschaft von 10 Personen hatte ich mehr erwartet. Es ist leider nur ein minimaler Prototyp entstanden. In Speedtrap wurde proprietäre Fremdsoftware integriert, so dass eine selbständige Nutzung von Speedtrap für unseren Projektpartner, der Kreispolizeibehörde Unna, fraglich ist. Auch nach mehrmaliger Korrektur des Administrationshandbuches ist mir nicht gelungen, das Programm ohne Hilfestellungen zu installieren. Die Ausgaben der Messergebnisse sind nur für Insider weiterverarbeitbar, was einer leichten Interpretation der Messergebnisse entgegensteht. Gerade in diesem Bereich wäre mit minimalem Aufwand eine graphische Ausgabe der Messergebnisse implementierbar gewesen. Schade.

Die PG hat die minimale Funktionsweise an 2 Messeinsätzen demonstriert und somit auch das Minimalziel der PG erreicht.

Zum guten Schluss, möchte ich allen mit auf den Weg geben, dass sie noch viel dazulernen müssen. Gerade wenn sie in der Berufspraxis in größere Projekte eingebunden werden oder vorher noch, wenn Sie ihre Diplomarbeit anfertigen. Hierzu wünsche ich allen viel Erfolg.

## 7.8 Firas Raed

Die Erfahrung, die ich während der PG gemacht habe, war ganz interessant. Diese Erfahrung wird mir mit Sicherheit in meinem beruflichen Leben viel helfen. Vor allem war interessant zu lernen, wie man im Team arbeitet. Ich kann ehrlich sagen, die meisten Probleme, die wir während der PG gehabt haben, waren mehr menschliche als technische Probleme. Mir ist ganz klar geworden, die Fähigkeit im Team zu arbeiten, ist genauso wichtig wie die wissenschaftlichen Kenntnisse. Zusammenfassend kann ich sagen, dass ich mit der PG sehr zufrieden war. Ich konnte meine Kenntnisse sowohl auf technische als auch auf menschliche Ebene verbessern und vertiefen.

## 7.9 Martin Schindler

Mein Fazit für die Projektgruppe: Schön dass es fast vorbei ist.

Zwar empfand ich die PG trotzdem als die interessanteste Lehrveranstaltung des ganzen Studiums, aber die Probleme während des zweiten Semesters bei (oder eigentlich nach) der Implementierung ließen mich öfters nicht mehr daran glauben, dass wir nicht mindestens drei Semester brauchen würden.

Der Arbeitsaufwand für die PG war im ersten Semester auch sehr viel geringer als ich erwartet hatte. Im zweiten Semester war es dann während der Implementierung richtig viel Aufwand. Glücklicherweise machte das aber auch Spaß. Java war mir neu, gefällt mir aber inzwischen richtig gut. Nach der Implementierung wurde es dann auch wieder ruhiger aber die Probleme aufgrund von unzähligen Bugs wurden dafür mehr. Und das hat dann gar keinen Spaß gemacht.

Am meisten hat mich gestört, dass die Möglichkeiten bei der Entwicklung so stark beschränkt waren. Wir hatten zwar einen Switch zum Testen bekommen, aber mit nur einem Switch lässt sich nunmal überhaupt nicht testen, ob unser Tool die Verbindungen zwischen zwei Switchen richtig erkennt. Und das Problem, dass verschiedene Switche sich leider völlig unterschiedlich verhalten bei gleichen SNMP-Anfragen ist dadurch erst im Einsatz in Unna aufgetreten.

Über die IRB kann ich mich nur wundern. Die Tatsache, dass wir an der KPBU machen konnten, was wir wollten, obwohl dort die Rechner und Daten um ein vielfaches sensibler und wichtiger sind, passt nicht dazu, dass wir an der Uni, trotz extra unterschriebener Nutzungsvereinbarung, nicht einmal lesenden Zugriff auf nur eine Netzwerkkarte kriegten und dadurch das geplante Backend für Segmente nicht entwickelt werden konnte. Dabei sollte man meinen, auch die IRB hat als Teil der Uni den Auftrag, Forschung und Lehre zu unterstützen.

Über Mitglieder und Beteiligte der Projektgruppe hab ich jetzt, wo wir fast fertig sind, nicht mehr viel zu meckern. Das tu ich deshalb auch nicht. Zeitweise war der Ärger sehr groß und es war tatsächlich so, dass wir mit weniger Leuten in dieser Zeit mehr erreicht hätten, aber geschafft haben wir es ja nun trotzdem. Insgesamt finde ich auch, dass wir schon Glück gehabt haben mit den Mitgliedern unserer Gruppe. Ich habe wirklich gute Leute kennengelernt und auch einiges Neues gelernt.

Jetzt folgt also nur noch die endlose Endbericht-Korrektur. Nach den Erfahrungen mit dem Zwischenbericht wird das lange dauern, aber irgendwann ist auch das vorbei. Und ich bin froh, dass das nächste Semester wieder schön locker wird. Mit wenig Stress, ausreichend Zeit zum Arbeiten und auch etwas mehr Freizeit.

## 7.10 Björn Schmidt

Am Ende steht ein Fazit, das Ende der PG 422 ist absehbar, jedoch ist die PG nocht nicht gänzlich abgeschlossen. Was die bisherige PG Arbeit angeht, so bin ich mit dem Erreichten zufrieden, auch wenn es unter anderen Umständen hätte deutlich mehr sein können.

Wir haben ein funktionsfähiges und bedienfreundliches Softwareprodukt erstellt, dass den Anforderungen laut PG Antrag genügt. Auch die weiteren für die PG vorgesehenen Aufgaben, wie die Praktika und Analysen wurden erfüllt.

Ich habe meine Programmierkentnisse vertiefen können und einige – wenn auch wenige – neue Techniken im Bereich des Projektmanagements gelernt. Zudem bin ich nun, was LaTeX angeht bestens für meine Diplomarbeit gerüstet.

Was die Projektarbeit angeht, so denke ich, ist es allen PG Mitgliedern mehr oder weniger klar, dass wir als Team nicht besonders gut funktioniert haben, wobei zu dem Team nicht zuletzt auch ein Teamleiter gehört.

Das liegt auf der einen Seite daran, dass es uns nicht gelungen ist, wirklich „ein" Team zu bilden, woran, so denke ich, wirklich alle PG Mitglieder Schuld sind. Daß die PG Mitglieder einfach sehr unterschiedliche Charaktere sind, ist dabei sicherlich auch keine Entschuldigung.

Auf der anderen Seite, ist es natürlich auch schwer, ein Produkt zu erstellen, wenn Projektmitglieder permanent den Projektfortschritt durch schlichtes Nichteinhalten von Terminen blockieren und selbst gesteckte Ziele nicht erreichen.

Darüber hinaus ist, meiner Meinung nach, allerdings auch unser Projektleiter mit Schuld daran, daß unsere PG nur knapp oberhalb der Minimalziele schliessen kann, hier wäre – z.B. bei wiederholten Terminüberschreitungen – ein ernergischeres Einschreiten sicherlich angebracht gewesen, da dem Projektmanagement als Teil der Gruppe natürlich Grenzen gesetzt sind.

Als vorlaufiges Fazit bleibt, dass sicherlich alle an dieser PG Beteiligten noch etwas im Bezug auf Teamarbeit dazulernen müssen, was aber im Rückblick auf diese PG sicher auch passieren wird. Die PG als Lehrveranstaltung existiert schliesslich nicht zuletzt auch deswegen, um eine Möglichkeit zu haben, relativ schmerzlos aus Fehlern zu lernen.

## 7.11 Christian Winkler

Im großen und ganzen kann ich sagen, dass sich meine Erwartungen an die PG erfüllt haben. Besonders durch die Seminare konnte ich meine Kenntnisse in Bezug auf IP-Netze und deren Leistungsbewertung vertiefen und habe in diesem Bereich ebenfalls einiges neues gelernt, wie beispielsweise Details zu SNMP. Von den Erfahrungen, die ich mit *Java* und *Latex* sammeln konnte, –was beides Neuland für mich war–, werde ich sicherlich noch weiter in Studium und Beruf profitieren können. Mit dem Ergebnis der PG, dem Tool *Speedtrap*, bin ich recht zufrieden, da es im Grunde alle Anforderungen der PG erfüllt und vor allem funktionstüchtig ist. Außerdem ist es recht benutzerfreundlich, so dass es auch für den Einsatz bei unseren "Kunden", der Polizei in Unna, geeignet ist.

Zu diesen positiven Erfahrungen, kommen aber leider auch einige negative. So musste ich feststellen, dass die Arbeit in größeren Gruppen nicht immer reibungslos verläuft, insbesondere was das Einhalten von Fristen angeht. Es ist schon traurig, dass es der Gruppe nicht möglich war, –trotz eingeführter Strafen–, auch nur einen einzigen Termin vollständig einzuhalten. Zudem lies die Arbeitsaufteilung bzw. das Engagement einzelner PG-Mitglieder oftmals zu wünschen übrig.

Schließlich noch ein Wort zur IRB. Es ist schade, dass von Seiten der IRB, ihr "guter Ruf" mal wieder bestätigt wurde, indem sie uns so manchen Stein in den Weg gelegt hat. Es ist schon seltsam, dass es nicht möglich ist, so einfache und unproblematische Dinge wie das Installieren von SNMP-Agenten auf einzelnen Rechnern, oder Zugriff auf die Rechnerhardware von der IRB zu bekommen, während dies bei der Kreispolizeibehörde kein Problem darstellt. Dabei stellt sich mir die Frage, wer wohl die kritischeren Netze bzw. Daten hat, oder ob es hier einfach um Aufwandsvermeidung ging, was wohl kaum der Sinn der IRB sein dürfte.

# Literaturverzeichnis


[Abts00] Abts, Dietmar: Grundkurs Java. 2. Aufl. Braunschweig: Vieweg Verlag, 2000.

[Alve97] Alvestrand, Harald: Object Identifiers Registry. http://www.alvestrand.no/-objectid/. - Aktualisierungsdatum: 10.02.1997

[Cisc03] Cisco Systems, Inc. : Cisco Catalyst 5000 Series Switches Catalyst 5500 Switching System, http://www.cisco.com/en/US/products/hw/switches/ps679/-products_data_sheet09186a008009260b.html, Aktualisierungsdatum: 2003.

[Doug01] Douglas R. Mauro & Kevin J. Schmidt: Essential SNMP, Help for System and Network Administrators, Edition 1, Köln: O'Reilly, July 2001

[Flan98] Flanagan, David: Java in a nutshell. 2. Aufl. Köln: O'Reilly, 1998.

[HPOp97] Hewlett-Packard, HP Open View: A Guide To Scalability and Distribution for Network Node Manager (NNM), Edition 1, Fort Collins, U.S.A., April 1997

[JSpa01] Halter, Steven L.: JavaSpaces[tm] Example by Example First edition ISBN 0-13-061916-7, Prentice Hall Upper Saddle River NJ USA, 2001

[Kobo00] Rose-Gerd Koboltsching, Explorative Datenanalyse, Skriptum zur Vorlesung, Universität Klagenfurt, WS 2000/2001

[Kmie01] Kmietsch, Jochen "Der Admin und seine Turnschuhe", Vortrag vor der LUG Braunschweig, 2001

[Kred02] Kredel, Heinz; Yoshida, Akitoshi: Thread- und Netzwerk- Programmierung mit Java. Heidelberg: dpunkt.verlag, 2002.

[Nets03] NET-SNMP Project: "The NET-SNMP Project Home Page" http://net-snmp.-sourceforge.net, Aktulisierungsdatum: 12.03.2003

[Perk97] David Perkins, Evan McGinnis: SNMP MIBs, Edition 1, Upper Saddle River, NJ: Prentice Hall, Inc., 1997

[P358EB] Projektgruppe 358: Endbericht Projektgruppe 358, Universität Dortmund, Lehrstuhl Informatik 4, Arbeitspapier 8/2/2001, 2001.

[P422ZB] Projektgruppe 422: Zwischenbericht Projektgruppe 422, Universität Dortmund, Lehrstuhl Informatik 4, Arbeitspapier 25/04, 2002.

[Schm01] Prof. Dr. Volker Schmidt, Stochastik für Informatiker, Physiker, Chemiker und Wirtschaftswissenschaftler, Vorlesungsskript, Universität Ulm, Abteilung Stochastik, Juli 2001

[Schr02] Schreiber, Hendrik: Performant Java programmieren. München: Addison-Wesley Verlag, 2002.

[Schw02] Schwägli, Tobias: Jini vs. Web Services Ein Leistungsvergleich, Fachgruppe Verteilte Systeme, FB Informatik, TH-Zürich, Dissertation, 2002.

[Sour02] Sourceforge Project: Network Packet Capture Facility for Java, http://sourceforge.net/projects/jpcap, Aktualisierungsdatum: 06.11.2002.

[SunM99] Sun Microsystems: Java Object Serialization Specification, http://java.sun.com/-j2se/1.3/docs/guide/serialization/spec/serialTOC.doc.html, Aktualisierungsdatum: 10.12.1999.

# Tabellenverzeichnis

# Abbildungsverzeichnis