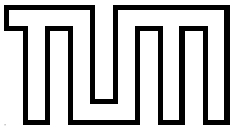


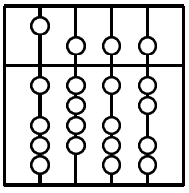
INSTITUT FÜR INFORMATIK

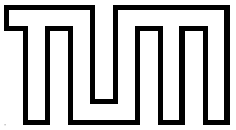
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

SEP

Entwicklung eines Systems zum automatisierten Energie-Management eines Testlabors

Bearbeiter:	Markus Gillmeister, Martin Metzker
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dipl.-Inform. Ralf König Dipl.-Inform. Vitalian Danciu Dipl.-Inform. Leonhard Bär (amasol AG)





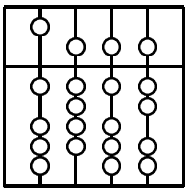
INSTITUT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

SEP

Entwicklung eines Systems zum automatisierten Energie-Management eines Testlabors

Bearbeiter:	Markus Gillmeister, Martin Metzker
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dipl.-Inform. Ralf König Dipl.-Inform. Vitalian Danciu Dipl.-Inform. Leonhard Bär (amasol AG)
Abgabetermin:	01. November 2007



Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 01. November 2007

. . . . . .

(Unterschrift des Kandidaten)

IT-Infrastrukturen werden immer leistungsfähiger und verbrauchen im Gegenzug immer mehr Energie. Die gestiegenen Energiekosten und die CO_2-Debatte führen momentan bei vielen Firmen zu einem Umdenken. Der Begriff der GreenIT, also der umweltschonenden und energiesparenden Hardware, rückt immer mehr in den Vordergrund. Trotzdem gibt es bisher nur wenige brauchbare Ansätze und keine Gesamtlösung. Ziel dieser Arbeit ist die Entwicklung eines Systems, um nicht genutzte Rechner (Clients + Server) policy- und heuristikbasiert ein- und auszuschalten. Das sogenannte Powerman-System wurde dabei für die Münchner IT-Firma amasol AG und ihren konkreten Anwendungsfall entwickelt. Es wurde ein modulares und skalierbares System entwickelt, dass für zukünftige Anwendungsfälle in anderen mittelständisches Unternehmen gut geeignet ist. Mit Hilfe des Systems war es amasol möglich ihre Energiekosten um 45% zu senken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Das Powerman-Projekt - so wurde dieses SEP intern bezeichnet. Was ursprünglich als kleines Webmanagement-Projekt begann, hat sich im Laufe von zahlreichen Meetings und Brainstormings zu einem großen umfassenden Management- und Monitoring-Tool mit vielen Funktionen (u.a. Client-Monitoring, Heuristiküberwachung, Kennlinienauswertung, Command-Line Interface) entwickelt. Dabei wurde stets darauf geachtet, das Powerman-System möglichst dynamisch und flexibel zu programmieren, um auch Möglichkeiten für zukünftige Modifikationen zu schaffen.

Zusätzlich zu dem reinen Software-Teil musste aber auch auf diverse Hardwarekomponenten (Strommessgeräte, Temperatursensor(en), schaltbare Steckdosenleiste) zurückgegriffen werden. Hierzu mussten Teile davon selbst entwickelt und gebaut werden.

Hierbei möchten wir uns bei Herrn Christof Häfele vom Leibniz Rechenzentrum München bedanken, der uns freundlicherweise ein qualitativ hochwertiges Strommessgerät für unsere Kennlinienerfassung zur Verfügung gestellt hat. Durch dieses Messgerät war es uns möglich präzise Messdaten der Clients zu erhalten.
Außerdem möchten wir uns bei Herrn Maurer und Herrn Preuße von der amasol AG bedanken, die uns neben unserem Betreuer Herr Bär mit viel Input und Feedback bei der Entwicklung unterstützt haben und beim Deployment kompetent zur Seite standen.

1 Motivation & Auftraggeber

IT-Infrastrukturen werden immer leistungsfähiger und verbrauchen immer mehr Energie. IBM schätzt, dass sich der Stromverbrauch von Datenzentren weltweit aktuell auf 100 Milliarden Kilowattstunden pro Jahr beläuft - mit stark ansteigender Tendenz. [WEB 1] Laut einer Studie von Jonathan Koomey, Professor an der Stanford-Universität, hat sich der gesamte Stromverbrauch für Server zwischen 2000 und 2005 verdoppelt.

Firmen stehen hier vor einer schwierigen Situation. Auf der einen Seite müssen sie mit gestiegenem Datenaufkommen zurechtkommen, was zwangsläufig in der Aufstockung Ihrer Hardware resultiert. Das bedeutet aber gegebenenfalls zusätzliche Investitionen in die Kühlung der EDV-Systeme, denn ein Großteil der Energie, die Rechner, Speichersysteme und Peripherie verbrauchen, wird in Wärme umgesetzt. Diese Wärme muss wiederum von leistungsfähigen Klimaanlagen abgeführt werden. Die Klimaanlagen verursachen sowohl Anschaffungs- als auch beträchtliche Betriebskosten. Auf der anderen Seite müssen Unternehmen versuchen die Betriebsmittelkosten möglichst gering zu halten.

Im Serverbereich und bei Rechenzentren ist das Einsparungspotential gering, da die Server dort in der Regel 24 Stunden am Tag permanent laufen müssen. Hier kann man als Unternehmen versuchen stromsparendere Hardware (z.B. neue Prozessoren) einzusetzen und kleinere Server in Virtual Machines zu konvertieren und auf einen größeren Server zu migrieren.

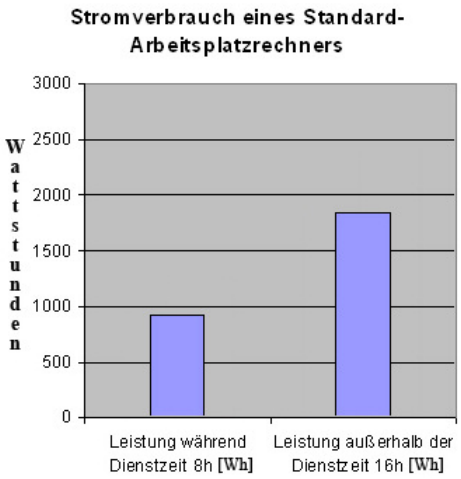


Abbildung 1.1: Stromverbrauch eines Arbeitsplatzes

Im Clientbereich hingegen gibt es ein deutlich größeres Einsparpotenzial. Hier handelt es sich meistens um fest installierte Workstations, die von Mitarbeitern während den normalen Geschäfts- und Bürozeiten genutzt werden. Gerade in größeren Firmen kann man häufig beobachten, dass Mitarbeiter die Rechner zum Feierabend nicht ausschalten, sondern lieber gleich eingeloggt bleiben (mit gesperrtem Bildschirm) um am nächsten Tag nicht das „störende“ Booten abwarten zu müssen. Oder aber die Administratoren haben alle Mitarbeiter angewiesen die Rechner nicht auszuschalten, damit nachts Updates und Wartungsarbeiten remote durchgeführt werden können. Nicht benöttigte Clients nachts in Betrieb zu halten bedeutet aber nicht zu unterschätzende Kosten für Unternehmen. Das Zentrum für Informations- und Mediendienste (ZIM) der Universität Duisburg/Essen hat diesbezüglich Untersuchungen angestellt und den Verbrauch von Arbeitsplatz-PCs, Monitoren und Peripherie während Arbeits- und Nachtzeiten vermessen. [WEB 2] Nebenstehende Grafik zeigt deutlich den hohen Verbrauch während der Leerlaufzeit. Neben den hohen Stromkosten hat die Leerlaufzeit negative Auswirkungen auf die Umwelt. Laut ZIM sorgt nebenstehender Beispielrechner für 12kg CO_2 Ausstoß, bei nächtlicher Abschaltung würde sich die CO_2-Belastung auf 4kg reduzieren. Hierbei hat man sich auf die Primär-Energieerzeugung mittels Steinkohle bezogen.

Natürlich könnte ein Unternehmen seine Mitarbeiter anweisen, die Rechner abends auszuschalten. Aber mit zunehmender Unternehmensgröße dürfte diese Bitte weniger fruchten, zumal hier oftmals die Administratoren gegen eine Abschaltung sind. Aus diesem Grund ist es nötig ein System zu entwickeln, um Rechner nach diversen Kriterien automatisch und manuell aus der Entfernung zu starten und herunterzufahren.

Das Ziel dieser Arbeit ist die Entwicklung eines solchen Powermanagement-Systems, um kleinen Unternehmen zu helfen, nicht-genutzte Ressourcen abzuschalten, langfristig Geld zu sparen und gleichzeitig Administratoren bei der Verwaltung eines kleinen Netzwerks zu unterstützen.

Unser System wird dabei in Zusammenarbeit mit der Münchner Firma amasol AG entwickelt. Diese möchte am Ende der Arbeit durch den Einsatz des Systems ihre hohen Energiekosten reduzieren und ihre Testserver steuern.

1.1 Kurzportfolio über die amasol AG

Die amasol Aktiengesellschaft für Informations- und Kommunikationstechnologie ist ein IT-Consulting-Unternehmen mit Sitz in München. Der Fokus der Firma liegt auf System-Management im Bereich Netzwerk, System, Applikationen und Service. Bei ihren Kunden handelt es sich ausschließlich um Unternehmen mit größeren bzw. komplexeren IT-Umgebungen, wie z.B. Audi AG, Bayer Business Services, DEVK Versicherungen, IBM Global Services und MAN IT Services.

Die amasol AG setzt dabei hauptsächlich auf die offene, standard-basierte Softwarelösung „eHealth Suite“ von CA. Mit Hilfe der Software ist es möglich vollautomatisch Trend-, Performance- und Kapazitäts-Management Berichte zu erstellen und durch kontinuierliche Analysen Aussagen über den Zustand des IT-Systems zu geben. Ihre Aufgabe ist neben der Planung, Konfiguration und Schulung (im Haus oder direkt beim Kunden) auch der ständige Support der IT-Abteilung des Unternehmens.

1.2 Szenario vor Ort

Zum Prozessablauf von Supportleistung gehört auch diverse Szenarien und Topologien von Unternehmen zu Testzwecken lokal nachzustellen und Situationen zu rekonstruieren. Hierfür stehen im Serverraum zahlreiche Testserver bereit, die aber auch gelegentlich für In-House-Schulungen verwendet werden.

Die Testserver sind dabei aber nicht vergleichbar mit den im normalen Sprachgebrauch gebrauchten Begriff „Server“, denn sie werden in der Regel nicht 24 Stunden pro Tag betrieben (mit der Ausnahme von Dauertests). Bisher wurde es so gehandhabt, dass Mitarbeiter, die einen Testrechner benötigen, in den Serverraum gehen und dort den Rechner einschalten und dann ggf. als Startmeldung eintragen, dass Sie derzeit den Rechner reserviert haben. Die Sun-Rechner verfügen über ein „Lights Out Management“, d.h. sie können auch remote gestartet und ausgeschalten werden.

Neben den Testservern gibt es auch einige Produktivserver für Web-, Mail- und Fileserver-Dienste. Sie finden in unserer weiteren Betrachtung keine Beachtung, da zum Zeit des Projekts nicht die Absicht besteht deren Betriebszeiten zu automatisieren.

Der grundsätzliche Aufbau des Serverraums kann man Abbildung 1.2 entnehmen. Der Serverraum besteht aus zwei Schränken mit jeweils 42 Höheneinheiten. In den Schränken befinden sich neben den Produktivservern der Firma (Storage, Mail, WWW etc.) 15 echte physikalische Testserver. Auf diesen Testservern laufen die Betriebssysteme Sun Solaris oder Microsoft Windows. Von allen Testservern werden wir am Ende dieses Projekts 12 Server (alle Testserver im rechten Schrank) über unser System steuern. Es handelt sich dabei um 5 Sun Netra X1, 3 Sun Sunfire V100 sowie 4 x86-kompatiblen PCs mit unterschiedlichen Versionen von Windows.

Problematisch für amasol ist, dass die Mitarbeiter häufig abends die Testrechner nicht mehr abschalten, diese dann über Nacht laufen und damit unnötig Kosten erzeugen. Zudem ist der Serverraum kein Serverraum im klassischen Sinne. Er befindet sich im Firmengebäude im vierten Stock, direkt unter dem Dach mit einem Fenster nach Osten. Das bedeutet, dass am Vormittag die Sonne direkt auf die beiden Racks scheint und so für zusätzliche Hitze sorgt. Im Laufe eines sonnigen Tages staut sich die Hitze und der Serverraum erreicht Temperaturen $\geq$30°Celsius. Der Raum verfügt über eine kleine Klimaanlage, die von den Mitarbeitern bei Bedarf eingeschaltet wird. Wegen dem hohen Stromverbrauch wird es aber eher vermieden, die Klimaanlage ständig betriebsbereit zu lassen. Es sind genau diese Rechner, die das meiste Einsparpotential bieten, weil sie wohl sehr oft eingeschaltet sind, jedoch selten auch über den ganzen Zeitraum hinweg benutzt werden.

Da unser System einen zentralen Server benötigen wird, um alle Rechner zu steuern, haben wir einen kurzen Blick auf relevante Teile der Netzstruktur von amasol geworfen. Das Netzwerk teilt sich in zwei große Teil-

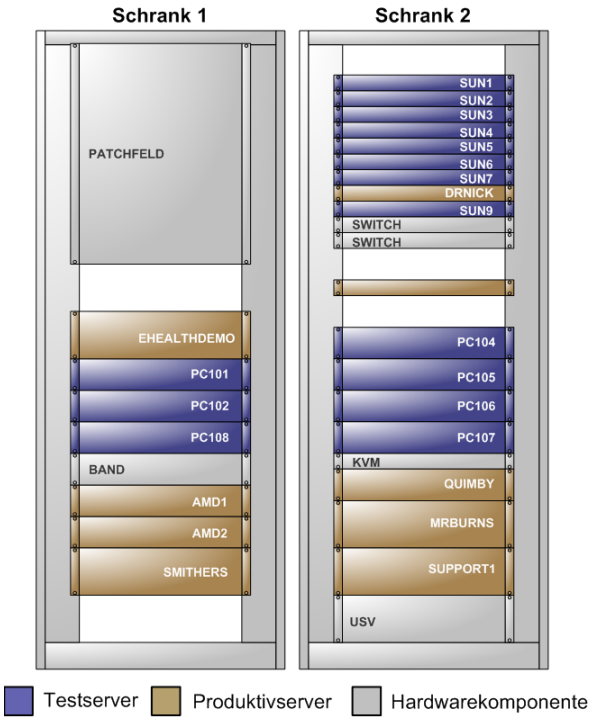


Abbildung 1.2: Skizze des Serverschranks (Frontansicht)

netze auf. Im Testing-Netz (192.168.43/24) befinden sich alle Testserver, im Inside-Netz (192.168.42/24) alle Produktivserver, Peripherie und Arbeitsplätze. Beide Netze sind über einen zentralen Router (Bart) miteinander verbunden. Der Router ist zugleich ein Konsolenserver, an dem alle LOMs der Sun Server angeschlossen sind. Somit können Mitarbeiter via `telnet bart <port>` auf die jeweiligen SUNs zugreifen und diese u.a. starten.

Abbildung 1.3: Übersicht über das Netzwerk (gekürzte Fassung)

1.3 Erfassen der Vorher-Situation

Um später herauszufinden, ob und wie stark sich das Projekt gelohnt hat, ist es nötig sich von der momentanen Situation ein Bild zu machen. Hierzu musste der Stromverbrauch über mehrere Wochen hinweg gemessen und protokolliert werden.

1.3.1 Messgerät

Zur Aufzeichnung des Stromverbrauchs der Serverschränke wurde zunächst eine digitale Aufzeichnungsvariante favorisiert. Jedoch konnte diese Idee aus Kostengründen nicht realisiert werden. Aus diesem Grund haben wir uns für ein relativ einfaches aber hinreichend genaues Messgerät entschieden, das bereits im privaten Repertoire vorhanden war: Das Stromverbrauchsmessgerät EM-600 von Elektronikdistributor ELV. Das Gerät beherrscht 5 Anzeige-Modi: Energiekosten in €, Energie in kWh, aktuelle Wirkleistung in Watt, momentane Netzspannung in Volt, Verbraucher-Gesamtzeit/ Einschaltzeit in Stunden. Die Genauigkeit dieses Messgerätes ist hinreichend für unsere Ansprüche. Bei dem Energieverbrauch liegt sie beispielsweise bei 1% (± 3 digit) laut Herstellerangaben. Für unsere Messungen benötigen wir lediglich den Energieverbrauch sowie die Gesamtzeit.

Abbildung 1.4: Messgerät ELV EM-600

1.3.2 Messung des Stromverbrauchs

Die Erstellung der Vorher/Nachher-Messreihe wurde mit Hilfe von Mitarbeitern bei amasol durchgeführt. Zunächst haben wir die Stromverkabelung im Serverraum verändert, so dass das Messgerät den Verbrauch der relevanten Server erfassen kann. Im Rahmen des regelmäßigen wöchentlichen Backupband-Wechsels durch einen Mitarbeiter wurden die Daten unseres Messgeräts in einer Tabelle erfasst und eine neue Messung gestartet. Notiert wurden Datum des Ablesezeitpunkts, der Energieverbrauch und die gemessene Zeit. Auf diese Weise erhielten wir nahezu gleichgroße Messintervalle, die wir zur ersten Prognose bzw. Auswertung heranziehen konnten.

1.3.3 Ergebnisse der Vorher-Messreihe

Messtag	gemessene Zeitspanne	Stromverbrauch
06.12.2006	113 h	22,69 kWh
13.12.2006	168 h	46,42 kWh
20.12.2006	167 h	49,17 kWh
27.12.2006	168 h	43,18 kWh
03.01.2007	167 h	49,80 kWh
10.01.2007	168 h	75,91 kWh
17.01.2007	168 h	58,20 kWh
24.01.2007	167 h	55,39 kWh
31.01.2007	168 h	62,17 kWh
07.02.2007	173 h	57,32 kWh
14.02.2007	164 h	59,77 kWh
21.02.2007	166 h	52,14 kWh

Tabelle 1.1: Messergebnisse der Vorher-Situation

Wie die Messreihe gezeigt hat, liegt der mittlere Stromverbrauch der Testserver bei 53,71 kWh pro Woche. Das entspricht bei einem Strompreis von 16ct/kWh (Angabe von amasol) etwa €412 pro Jahr. Da die Kosten bereits relativ niedrig sind, kann natürlich auch das Einsparpotenzial nicht sehr groß sein. Ziel der Arbeit soll es aber sein, zu zeigen, dass etwas eingespart werden kann.

2 Anforderungsspezifikation

Wie es für jedes Software-Projekt üblich ist, wird zu Beginn gemeinsam mit dem Kunden ein Lastenheft (Anforderungsspezifikation) angelegt. In dieser Spezifikation stehen alle Anforderungen und Wünsche an dieses Projekt, die später umgesetzt werden müssen. Uns wurde empfohlen, hierzu das Volere Requirements Specification Template als Grundlage zu verwenden. Das Template dient als umfassende Vorlage für die Aufnahme aller Arten von Anforderungen. Die Anforderungen lassen sich dabei wie folgt gliedern:

- Systemrandbedingungen
  Hier werden allgemeine Fakten, wie z.B. den Zweck des Projektes, die Kunden sowie Randbedingungen festgehalten. Diese Punkte werden durch die ersten Kapitel dieses Dokumentes beschrieben.
- Funktionale Anforderungen
  Unter diesem Punkt versteht man die tatsächlichen Funktionen und Daten, die das System am Ende haben soll („harte“ Fakten).
- Nicht-funktionale Anforderungen
  Im Gegensatz zu den funktionalen Anforderungen stehen in den nicht-funktionalen Anforderungen alle nicht „greifbaren“ Features des Systems. So kann es z.B. vom Auftraggeber bestimmte Anforderungen an die Oberfläche, die Benutzbarkeit, die Performance oder die Sicherheit geben. Aber auch politische oder gesetzliche Anforderungen müssten bei größeren Projekten hier auftauchen.
- Projektrandbedingungen
  Hier tauchen alle Punkte auf, die nach der Entwicklung auftauchen. Dazu zählen ggf. offene Punkte (Kapitel 6.2), neue Probleme, die Inbetriebnahme (Anhang B), Risiken oder die Benutzerdokumentation (Anhang A). Wir widmen diesen Punkten eigene Kapitel in diesem Dokument.

Alle Anforderungen werden mittels Volere Atomic Requirement Template (oder auch Snowcard genannt) festgehalten und sind nach Anerkennung von allen Beteiligten verbindlich. Eine typische Snowcard enthält folgende Punkte:

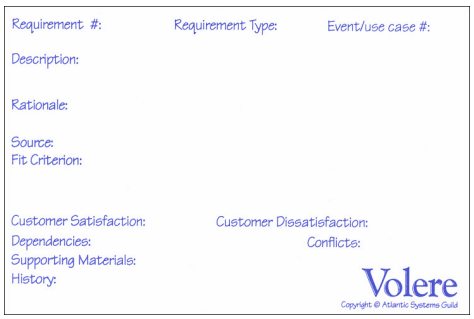


Abbildung 2.1: Volere Snowcard

- eindeutige Anforderungsnummer
- Beschreibung der Anforderung
- optional: Begründung
- Prüfkriterien für die Erfüllung der Anforderung
- Quelle (Person, der die Anforderung gestellt hat)
- Un- und Zufriedenheit (Priorisierung der Anforderung)
- Abhängigkeiten (Verweise auf andere Anforderungen)
- Zusätzliche Materialien (Verweise auf Definitionen, Modelle und Dokumente, die die Anforderung verdeutlichen)

In unserem vorliegenden Fall handelt es sich um ein sehr kompaktes Entwicklungsprojekt. Es gibt keine mehrstufigen Abhängigkeiten (z.B. Team A entwickelt Komponente X, die Team B für die Entwicklung von Komponente Y benötigt) und das Projektteam besteht aus nur 6 Mitgliedern (davon 2 Entwickler). Aus diesem Grund haben wir uns entschlossen für dieses Projekt kompaktere Snowcards einzusetzen, die unseren Anforderungen immer noch voll genügen.

2.1 Anforderungen der amasol AG

Anforderungs-ID	01	Anforderungstyp	Funktional
Anforderung	**Command Line Interface (CLI)**		
Priorität	wichtig		
Beschreibung	Ein CLI um die wichtigsten Funktionen des Powerman Servers zu steuern. Es soll die Möglichkeit geben, Rechner ein- bzw. auszuschalten oder neuzustarten. Außendienstmitarbeiter benötigen hin und wieder Testrechner aus dem Serverraum, die sie via SSH-Verbindung einschalten sollen. Ein Tunnel zum Webfrontend ist nicht praxistauglich.		
Prüfkriterium	CLI bietet genannte Möglichkeiten im Testbetrieb.		

Anforderungs-ID	02	Anforderungstyp	Nicht funktional
Anforderung	**Simples Benutzermanagement**		
Priorität	nötig für das Projekt		
Beschreibung	Da alle Mitarbeiter in der Firma gleichberechtigt beim Zugriff auf die Testrechner sind, ist kein kompliziertes Benutzermanagement mit Rechten erforderlich. Um dennoch Schutz vor unbefugten Zugriff zu gewährleisten, genügt eine einfach HTACCESS-Authentifizierung.		
Prüfkriterium	Web-Frontend kann nur nach Passwortauthentifizierung aufgerufen werden.		

Anforderungs-ID	03	Anforderungstyp	Funktional
Anforderung	**Webinterface**		
Priorität	nötig für das Projekt		
Beschreibung	Eine zentrale Steuerung aller implementierten Funktionen über ein Webfrontend. Grafische Aufbereitung und Auswertung aller gesammelten Informationen.		
Prüfkriterium	Geforderte Funktionalität vorhanden.		

Anforderungs-ID	04	Anforderungstyp	Funktional
Anforderung	**Zeitbasiertes Ein/Ausschalten von Clients**		
Priorität	wichtig		
Beschreibung	Clients sollen durch Regeln geschalten werden. Diese Regeln sollen jederzeit überschreibbar sein.		
Prüfkriterium	Erfolgreicher Abschluss des Dauertests.		

Anforderungs-ID	05	Anforderungstyp	Nicht Funktional
Anforderung	**Temperaturüberwachung des Raums**		
Priorität	optional		
Beschreibung	Die Temperatur des Serverraums soll überwacht werden können. Hierzu sollte nach einer günstigen Hardwarelösung gesucht werden und in das Webfrontend integriert werden.		
Prüfkriterium	Temperaturverlauf des Sensors im Frontend abrufbar.		

Anforderungs-ID	16	Anforderungstyp	Projektrandbed.
Anforderung	**Separate Dokumentation für Benutzer**		
Priorität	nötig		
Beschreibung	Eine kurze, übersichtliche Dokumentation für Benutzer des Systems.		
Prüfkriterium	Benutzung des Systems ohne Rückfragen zur Bedienung.		

2.2 Anforderungen des Lehrstuhls

Anforderungs-ID	06	Anforderungstyp	Nicht funktional
Anforderung	**Kennlinien zur Abschätzung des Stromverbrauchs**		
Priorität	sinnvoll		
Beschreibung	Von den verschiedenen Rechnerklassen (Sun, PC) bei amasol soll ein „Fingerabdruck“ für den Stromverbrauch eines Rechners unter verschiedenen Lastsituationen erzeugt werden. Die gewonnenen Kennlinien sollen später dazu dienen, um den Stromverbrauch des Serverschranks abzuschätzen.		
Prüfkriterium	Stromverbrauch durch Lasterzeugung mit dem Testprogramm ist vorhersagbar.		

Anforderungs-ID	07	Anforderungstyp	Funktional
Anforderung	**Schaltbare Steckdosenleiste für Clients ohne LOM**		
Priorität	wichtig		
Beschreibung	Clients ohne LOM (Lights Out Management) und Watchdog-Card sollen ebenfalls zurückgesetzt werden können (powercycle). Um das zu realisieren müssen diese Clients an einer schaltbaren Steckdosenleiste hängen. Hierzu muss eine günstige Hardwarelösung gefunden werden und in die Steuerung bzw. das Frontend eingebunden werden.		
Prüfkriterium	Clients können mittels Web-Frontend vom Stromnetz getrennt werden.		

Anforderungs-ID	08	Anforderungstyp	Funktional
Anforderung	**Heuristiken**		
Priorität	wichtig		
Beschreibung	Eine reine policybasierte Steuerung der Clients genügt nicht, da sie zu unflexibel und starr ist. Es ist sinnvoll, dass der Steuermechanismus auf Basis von statistischen Daten Entscheidungen trifft um Clients abzuschalten (bei langer Inaktivität) und ggf. ein Herunterfahren zu unterbinden (wenn noch Benutzer eingeloggt sind).		
Prüfkriterium	Rechner fahren sich bei Inaktivität selbstständig herunter.		

Anforderungs-ID	09	Anforderungstyp	Nicht funktional
Anforderung	**Kalenderdarstellung im Frontend**		
Priorität	empfehlenswert		
Beschreibung	Im Frontend soll es für jeden Client eine Möglichkeit geben, grafisch visualisiert einen Überblick über die eingetragenen Start/Stopp-Zeiten eines Monats zu bekommen.		
Prüfkriterium	Grafik im Web-Frontend visualisiert Betriebszeiten.		

Anforderungs-ID	10	Anforderungstyp	Funktional
Anforderung	**Quickbooking**		
Priorität	wichtig		
Beschreibung	Im Web-Frontend soll es die Möglichkeit geben mit nur wenigen Handgriffen einen Rechner für die nächsten X Stunden zu aktivieren ohne Berücksichtigung sonstiger Einstellungen.		
Prüfkriterium	Rechner wird sofort automatisch gestartet und nach Ablauf der Zeit heruntergefahren.		

Anforderungs-ID	14	Anforderungstyp	Funktional
Anforderung	**Kompaktes Benutzermanagement**		
Priorität	notwendig		
Beschreibung	Gewisse Funktionen des Systems sollten Administratoren vorbehalten bleiben. Außerdem sollte transparent zurückverfolgt werden können welcher Benutzer welchen Aktion ausgelöst hat. Aus diesem Grund ist ein kleines Benutzermanagement erforderlich, das zwei Gruppen (Administratoren und Benutzer) kennt.		
Prüfkriterium			

Nachdem nun die Anforderungen an das Projekt definiert wurden, ist es nötig, sich einen Überblick über bestehende Marktlösungen zu verschaffen. So kann herausgefunden werden, ob es bereits Hersteller gibt, die auf das Powermanagement zugeschnittene Lösungen verkaufen und wie wichtig Eigenentwicklungen durch uns sind.

3 Marktüberblick bereits existierender (Teil-)Lösungen

Bevor an die Realisierung des Powerman gedacht werden kann, soll zunächst im Rahmen dieser Arbeit untersucht werden, welche Lösungen bereits auf dem Markt existieren. Dabei gehen wir sowhl auf Hardware- als auch auf Software-Lösungen ein. Aus Platzgründen untersuchen wir nur eine Auswahl von Lösungen.

3.1 Hardware

Bei Hardware-Lösungen im Bereich Powermanagement geht es hauptsächlich darum, Verbraucher vom Stromnetz zu trennen und bei Bedarf per Fernsteuerung wieder anzuhängen. Wir analysieren im Folgenden eine Auswahl der Produkte der größeren Hersteller aber auch zwei unkonventionelle bzw. billigere Lösungen. Außerdem werden wir uns in diesem Kapitel Temperatursensoren zuwenden.

3.1.1 Schaltbare Netzschalter

Mit schaltbaren Netzschaltern ist es möglich, die Endgeräte vollständig vom Netz zu trennen. Computer, die nach der ATX-Spezifikation gebaut sind, werden selten komplett abgeschaltet. Das Mainboard wird in den Soft-Off-Zustand versetzt, aus dem es auf verschiedene Arten geholt (aufgeweckt) werden kann. Zu den Möglichkeiten des Aufweckens gehören meist ein Knopf an einer für den Benutzer gut erreichbaren Stelle am Gehäuse und das für unsere Entwicklung wichtige Wake On Lan, bei dem ein Computer über das Netzwerk aufgeweckt werden kann. Echt abschalten lassen sich diese Computer meist nur über einen Schalter direkt am Netzteil, während einige sogar gar keinen Schalter dafür besitzen. Im Standby-Zustand verbrauchen PCs, nach eigener Messung, bis zu 5 Watt. Um einen maximalen Einspareffekt zu erzielen, ist es wichtig den Rechner vollständig abzuschalten. Dies wird durch (fern-)steuerbare Netzschalter erreicht. Zudem ermöglichen diese Netzschalter bei Rechnern, die nicht mehr auf Benutzereingaben reagieren, einen Reset aus der Ferne. Für die Steuerung der Netzschalter gibt es keinen Standard, jeder Hersteller setzt in der Regel auf seine eigenen Lösungen. Je nach Größe des Systems erfolgt die Steuerung über ein Web-Frontend (bei IP-basierten Lösungen) oder über spezielle Software (bei direkt an einen PC angeschlossene Lösungen).

Leunig GmbH

Die Firma Leunig GmbH hat zahlreiche Powermanagement-Komponenten in Ihrem Portfolio. Die Geräte sind in 19” Bauweise und können somit in ein Rack gebaut werden. Alle Netzschalter sind IP-fähig und können via Web-Frontend gesteuert werden. Nur die teuerste Serie des Herstellers (X-Serie) ist SSL-fähig. Bei allen anderen muss man - je nach Netzinfrastruktur - selbst für Sicherheit sorgen, beispielsweise durch die Verwendung von VLANs. Die Geräte der M- und X-Serie sind kaskadierbar. Dadurch können je nach Serie bis zu 156 Steckdosen mit einem Interface gesteuert werden. Um die Netzschalter in Serie zu schalten benötigt man ein Master-Gerät der jeweiligen Serie, welches neben 8 Steckdosen das Management-Interface enthält.

Abbildung 3.1: ePower X-Serie, Quelle:Hersteller

Infratec AG

Die Geräte der Firma Infratec sind vom Leistungsumfang nahezu identisch zu denen der Firma Leunig GmbH. Exemplarisch haben wir ein Modell aus der Produktpalette für die Betrachtung ausgewählt. Alle Geräte können über ein Web-Frontend gesteuert werden. Eine SNMP-Funktionalität kann bei allen Modellen (sofern nicht schon vorhanden) nachgerüstet werden.

Abbildung 3.2: Infratec PM-821, Quelle:Hersteller

Aten

Der Hersteller Aten hat sich, ebenso wie die beiden oben genannten Firmen, auf Netzwerkperipherie spezialisiert. Aten stellt eine Steckdosenleiste her. Das Besondere an diesem Produkt ist, dass mit dem Gerät die kumulative Last aller angeschlossenen Verbraucher gemessen werden kann. Die Geräte können via Web-Frontend oder Telnet gesteuert werden.

CT-Netzschalter

Hierbei handelt es sich um ein Selbstbauprojekt der Computerzeitschrift c't. [WEB 3] Vollständig aufgebaut kann die Schaltung bis zu sechs 230-Volt-Geräte ein- und ausschalten. Bei zwei der Geräte kann zusätzlich Strom und Spannung überwacht werden. Die Schaltung besitzt eine Netzwerkschnittstelle und kann komfortabel per Web-Frontend gesteuert werden. Kernstück der Schaltung ist ein Atmel Mikrocontroller, auf dem ein Mini-Webserver und die Steuerung der Steckdosen implementiert sind. Die Netzwerkschnittstelle wurde mit einem XPort-Baustein realisiert. Der Atmel Chip kommuniziert dabei über die integrierte serielle Schnittstelle mit dem XPort-Chip, der aus den Daten dann ein Netzwerkpaket erstellt bzw. ankommende Daten für den Atmel aufbereitet. Diese Lösung erfordert neben guten Lötkenntnissen einige Vorkenntnisse über die Programmierung von Atmel-Controllern und kann ohne zusätzliche Bauteile (Atmel Programmierboard) nicht durchgeführt werden. Die Teilelisten und der Sourcecode kann von der Projektwebseite heruntergeladen werden. Komplette Teilesätze werden von SEGOR-electronics[1] vertrieben, sind aber preislich nicht rentabel.

Abbildung 3.3: CT-Netzschalter, Quelle:Heise CT

Gembird

Der Gembird-Netzschalter ist die einzige hier vorgestellte Steckdosenleiste, die nicht via IP sondern nur über USB gesteuert werden kann. Sie wird in Deutschland u.a. unter dem Namen Revolt Intelli-Plug von der Firma Pearl vertrieben. Von der Verarbeitung und der Bedienung ist sie eher für den Consumer-Markt gedacht.

[1] http://www.segor.de/L1Bausaetze/NetzSchalter.shtml

Gesamtübersicht der technischen Daten

Hersteller	Modell	Steckdosen	Schnittstelle	Kaskadierbar	Marktpreis Master	Slave
Leunig GmbH	ePower M-Serie	8	Ethernet *	ja (5 Geräte)	€ 683,-	€ 544,-
Leunig GmbH	ePower X-Serie	8	Ethernet *	ja (17 Geräte)	€ 1142,-	€ 574,-
Infratec AG	PM821	8	Ethernet	nein	€ 469,-	
Aten	PN9108	8	Ethernet	ja (16 Geräte)	€ 596,-	
CT	Netzschalter	6	Ethernet	nein	€ 205,-	
Gembird	Netzschalter	4	USB	nein	€ 30,-	

* nur Master-Gerät

3.1.2 Temperatursensoren

Bisher besitzt der Serverraum von amasol keine automatisierte Temperaturüberwachung. Bei Bedarf wird die Klimaanlage manuell bedient. Mit Hilfe von Temperatursensoren lassen sich präzisere Aussagen treffen und sie können gleichzeitig als Überwachung (Überhitzung) eingesetzt werden. Der Markt für Temperatursensoren, die über PC ausgelesen werden können, ist sehr klein, wie unsere Recherchen gezeigt haben. Meist sind die Kauflösungen nur als komplette Wetterstation (mit Messungen für Luftdruck, Niederschlag usw.) zu haben, was den Preis in diesem Fall unnötig in die Höhe treibt und damit für das Projekt nicht geeignet sind.

W&T Web-Thermograph pt100

Die Firma Wiesemann & Theis GmbH hat sich auf Schnittstellen zwischen Daten- und Messsensoren auf der einen und Anwendungen (z.B. SNMP) auf der anderen Seite spezialisiert. In Ihrem Sortiment findet sich ein sogenannter Web-Thermograph mit dem die Temperatur eines Sensors via SNMP, Webbrowser, E-Mail oder mittels kostenlos bereitgestellter Software ausgelesen werden kann. Leider hat diese professionelle Handarbeit auch Ihren Preis. Mit €298 ist der Temperatursensor im Rahmen dieses Projekts unerschwinglich.

ELV Funk-Datenlogger

Der Funk-Datenlogger von dem Elektronikdistributor ELV ist eigentlich eher für den Consumer Bereich gedacht. Er kann Daten verschiedener Klimasensoren erfassen, aufzeichnen und zur Auswertung auf den PC übertragen. Neben der USB-PC-Schnittstelle besitzt er ein LCD-Display, in dem die Daten der Sensoren direkt angezeigt werden können. Für €99,95 bekommt man den Datenlogger im Set mit einem Temperatursensor. Um „nur“ die Temperatur im Serverraum zu Erfassen ist das Gerät damit überdimensioniert und schlichtweg zu teuer.

1-Wire Temperatursensor im Eigenbau

Wie auch schon beim schaltbaren Netzschalter gibt es auch eine - in diesem Fall - sehr günstige Selbstbauanleitung. Richard Lippmann [WEB 4] beschreibt auf seiner Webseite den Aufbau der Schaltung. Im Wesentlichen besteht die Schaltung aus einem kleinen 1-Wire Temperatursensor von Dallas Semiconductor (Modell DS1820). Dieser IC enthält bereits fast die nötige Schaltungselektronik um mit dem PC via serieller Schnittstelle zu kommunizieren. Mit lediglich sechs anderen Bauteilen, etwas Lötzinn und Kabel läßt sich die Schaltung komplettieren. Preislich liegt der Sensor bei unter €10 (ohne Gehäuse und ggf. notwendige USB-auf-Seriell-Converter).

3.2 Software/Projekte

Eine Powermanagement-Komponente in einer großen Management-Applikation ließ sich während der Vorrecherchen bei keinem Produkt finden. Zur näheren Betrachtung standen HP Openview, Nagios und CheopsNG im Raum.

HP Openview scheint bereits einen Großteil der von uns geforderten Fähigkeiten zu besitzen. Da dieses Produkt jedoch ausschließlich komerziell vertrieben wird, haben wir es auch aus finanziellen Gründen nicht untersuchen können. Mit Nagios ist es möglich die komplette Infrastruktur einer Firma abzubilden. Das Tool dient primär der Überwachung und nicht der Steuerung. Zwar wäre es möglich Nagios zu erweitern, so dass weitere Funktionen wie „Starten/Stoppen“ eines Rechners implementiert werden, aber die komplexen Anforderungen an den Powerman erfordern zuviele Modifikationen an Nagios. Somit wird ein eigener Serverdienst erforderlich (siehe nächstes Kapitel). CheopsNG ist ähnlich wie Nagios ein Tool um Netzwerke abzubilden. Dabei können zwar bereits benutzerdefinierte Aktionen für einzelne Hosts einfach implementiert werden. Allerdings wurde CheopsNG zuletzt 2003 weiterentwickelt und erwies sich in unserem Test als äußerst instabil.

4 Die eigene Lösung Powerman

Unsere Anforderungen an die Gesamtlösung lassen sich in drei Teilaspekte aufteilen. Da sich unser Projekt mit Stromverbrauch beschäftigt, ist der offensichtlichste Teil das **Ein- bzw. Ausschalten von Computern**. Dieser Bereich soll sich mit den technischen Aspekten wie Hardware und Übertragunstechnik beschäftigen. Ein anderer Aspekt ist die **Verwaltung**. Der Benutzer soll durchaus Einfluss auf das Verhalten des Systems nehmen können, zum Beispiel durch das Anlegen von Grundbetriebszeiten (Kapitel 4.1). Der ganze Aspekt Verwaltung beschäftigt sich mit abstrakten Daten und Strukturen, die losgelöst von der Hardware existieren. Der dritte Teil ist die **Steuerlogik**. Sie ist die Schnittstelle zwischen Verwaltung und Hardware. Hier werden die Verwaltungseinstellungen auf die Hardware abgebildet, z.B. veranlasst das System einen Computer dazu herunter zu fahren, wenn eine Grundbetriebszeit dies vorsieht.

Alle vorgestellten Lösungen haben die Eigenschaft, dass sie sehr speziell sind. Bisher verfügbare Software beschränkt sich weitestgehend auf das Überwachen von Netzwerkkomponenten, jedoch bleiben alle Aspekte der Verwaltung und der daraus resultierenden Automatisierung unerfüllt. Bei Hardware ist das Problem ähnlich. Außer der grundlegenden Funktion des Schaltens von Steckdosen sind die Funktionen sehr beschränkt. Bei einigen Produkten, wie z.B. der Gembird Steckdosenleiste, ist es möglich einen Zeitplan einzurichten, anhand dessen die Steckdose selbstständig schaltet. Dieser Funktionsumfang ist uns jedoch zu gering. Zusätzlich fehlt jede Art der Abstraktion, wodurch wiederum Aspekte der Verwaltung auf der Strecke bleiben.

Aus diesen Gründen entwickeln wir unsere eigene Lösung, welche die drei oben genannten Teilaspekte vereint.

4.1 Begriffe und Definitionen

Im weiteren Verlauf der Arbeit beziehen wir uns oft auf am Powerman-System beteiligte Computer und gewisse Vorschriften dafür. Um Umstände und Beziehungen nicht wiederholt beschreiben zu müssen führen wir hier einige wenige Begriffe ein.

Server Der Server oder Powerman-Server ist derjenige Computer, auf dem die Software der Steuerlogik und der Verwaltung installiert sind.

Client Als Clients werden allgemein alle Rechner bezeichnet, die potentiell vom Server aus gesteuert werden können.

Betriebszeit Eine Betriebszeit beschreibt eine Zeitspanne, zu der ein Computer an sein soll. Eine Betriebszeit besteht aus einer Start- und einer Stoppzeit zwischen 00:00 Uhr und 23:55 Uhr. Da diese Zeiten eine Uhrzeit sind, sprechen wir auch von Start- und Stoppzeitpunkt.

Grundbetriebszeit Grundbetriebszeiten sind ein Wochenplan, bestehend aus einer Menge von Betriebszeiten, die je einem Wochentag zugeordnet sind. Da es sich um einen Wochenplan handelt, enthalten Grundbetriebszeiten keinerlei Datumsinformationen. Grundbetriebszeiten werden getrennt von Clients verwaltet, d.h. eine Grundbetriebszeit kann beliebig vielen Clients zugewiesen werden. Damit ein Client einer Grundbetriebszeit ensprechend gesteuert wird, muss dem Client diese Grundbetriebszeit explizit in einer Regel zugewiesen werden.

Regel Die Zuweisung von einer Grundbetriebszeit zu einem Client heißt Regel. Bei dem Erstellen einer Regel müssen ein Start- und ein Enddatum angegeben werden. Diese Daten definieren den Gültigkeitsbereich der Grundbetriebszeit für diesen einen Computer, also den Zeitraum, innerhalb dessen die Grundbetriebszeit bei dem Client Anwendung findet. So lange die Regel gültig ist, wiederholt sich der Wochenplan immer wieder. Start- und Enddatum müssen daher nicht auf Montag bzw. Sonntag fallen und können beliebig weit auseinander liegen. Eine Regel kann für nur einen Tag gelten, sie kann sich jedoch

auch über mehrere Jahre erstrecken. Eine Regel ist lediglich eine Zuweisung, keine Kopie der Grundbetriebszeit. Eine Änderung der Grundbetriebszeit betrifft alle Rechner, denen diese Grundbetriebszeit zugewiesen ist.

Eine Grundbetriebszeit g heißt aktiv an einem Zeitpunkt t für einen bestimmten Client c, wenn t zwischen dem Start- und Enddatum der Regel die g zu c zuweist liegt und t zwischen Start- und Stoppzeit einer Betriebszeit aus g für den gegenwärtigen Tag liegt.

Heuristik Der Powerman-Server holt in regelmäßigen Abständen Informationen über die Clients ein, welche als Entscheidugnsgrundlage zur Steuerung der Clients benutzt werden können. Diese Entscheidungen werden immer nach dem selben Schema getroffen. Der Server prüft ein Prädikat (eine Aussage, die zu jedem Zeitpunkt entweder wahr oder falsch ist) auf den erhaltenen Informationen. Ist dieses Prädikat für einen bestimmten Zeitraum wahr, so wird die mit diesem Prädikat verknüpfte Aktion ausgelöst. Eine schematische Darstellung von Heuristiken findet sich in Tabelle 4.1.

Ein Prädikat zusammen mit der Angabe eines Zeitraums und einer Aktion heißt Heuristik.

Eine Heuristik h heißt aktiv an einem Zeitpunkt t, wenn zum Zeitpunkt t das Prädikat von h mindestens für den Zeitraum von h ununterbrochen wahr ist.

Eine Heuristik wird immer einem Client direkt zugewiesen. Die Aktion ist immer entweder „herunterfahren“, „neustarten“ oder „block“ (vgl. Tabelle 4.1).

Aktion	**Bedeutung**
herunterfahren	fährt den betroffenen Client herunter
neustarten	startet den betroffenen Client neu
block	blockiert andere Entscheidungen der Steuerlogik für den betroffenen Client

Tabelle 4.1: Aktionsmöglichkeiten von Heuristiken

<Heuristik> :=	<Prädikat> <Zeitraum> <Aktion>
<Zeitraum> :=	**1,2,...,n Minuten**
<Aktion> :=	**neustarten** | **herunterfahren** | **block**
<Prädikat> :=	<Merkmal> <**größer als** | **kleiner als** > <Anzahl> | <Prozess>
<Merkmal> :=	**Anzahl angemeldeter Benutzer** | **CPU-Last in %** | **Netz-Last in %** | **Temperatur in °C** | **Anzahl Zombieprozesse**>
<Anzahl> :=	**1,2,...,n**
<Prozess> :=	<Name: > **läuft nicht**
<Name: > :=	**Prozessname**

Tabelle 4.2: Heuristiken in BNF

Ausnahme Ausnahmen sind Spezialfälle von Grundbetriebszeiten. Ausnahmen sind ein einziger, sich nicht wiederholender, beliebig langer Zeitraum, in dem ein Client durchgehend entweder an oder aus sein soll.

Eine Ausnahme heißt aktiv an einem Zeitpunkt t, wenn t zwischen Start- und Endzeitpunkt der Ausnahme liegt.

4.2 Grundlegender Aufbau

Das Powerman-System setzt sich aus mehreren Komponenten zusammen. Wir benötigen auf jedem Client spezielle Software (Powerman-Clientsoftware, Kapitel 4.6) und auf dem Server muss die Powerman-Serversoftware (Kapitel 4.7) installiert sein. Die Powerman-Serversoftware besteht aus der Datenbank (Kapitel 4.7.1), dem

Command Line Interface (Kapitel 4.7.2) der Client Control Unit (kurz CCU, Kapitel 4.7.3) und dem Powerman-Frontend (Kapitel 4.7.4), welches der sichtbarste und aufwändigste Bestandteil ist. Dazu kommen noch Komponenten, die direkt zum Ein- und Ausschalten von Clients benutzt werden (Managementkomponenten) z.B. Gembird Steckerleisten oder das Sun Lights Out Management (LOM). Abbildung 4.1 zeigt grob den Aufbau unseres Systems. Die Clientsoftware nimmt Befehle entgegen und stellt Informationen zur Verfügung, während auf dem Server drei Softwarekomponenten interagieren. Managementkomponenten sind für die Benutzung des Powerman prinzipiell nicht notwendig, erlauben jedoch eine präzisere Steuerung.

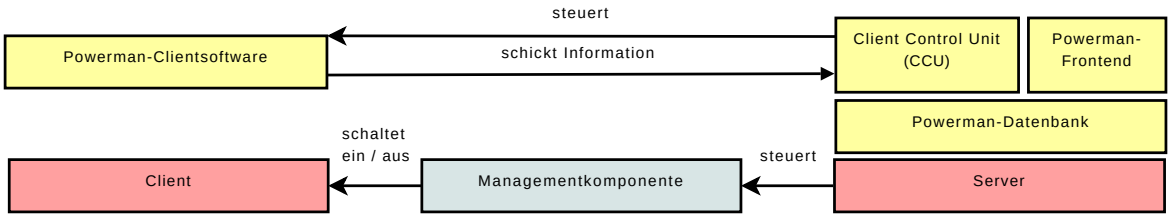


Abbildung 4.1: Grundlegender Aufbau des Powerman-Systems

4.3 Kontrollschichten und Struktur

Unser System soll die Möglichkeit bieten das Energiemanagement zu automatisieren. Dazu muss es mit Daten versorgt werden, aufgrund welcher die Steuerlogik Entscheidungen treffen kann. Wir haben drei Kontrollschichten für die Automatisierung vorgesehen:

- Grundbetriebszeiten
- Heuristiken
- Ausnahmen

Die Kontrollschichten unterscheiden sich nicht nur in ihrem Verhalten, sondern auch in ihrer Priorität. Grundbetriebszeiten können durch Heuristiken ausgehebelt werden und diese wiederum sind den Ausnahmen unterlegen. Grundbetriebszeiten sind ein Wochenplan. Sie veranlassen die Steuerlogik einen Client zu Beginn einer Betriebszeit zu starten und zum Ende wieder herunter zu fahren. Heuristiken haben nur Auswirkungen auf laufende Rechner. Die Aktion einer Heuristik wird in Abhängigkeit des Zustandes des Clients ausgelöst. So können Heuristiken z.B. einen Client der für einen bestimmten Zeitraum ununterbrochen unausgelastet ist (niedrige CPU-Last) herunterfahren, obwohl der Client inmitten einer Betriebszeit ist. Heuristiken können Aktionen blocken, so kann z.B. eine Heuristik das Herunterfahren eines Clients verhindern, wenn noch Benutzer auf dem Client arbeiten. Dies gilt jedoch nicht bei Ausnahmen. Ausnahmen hebeln sowohl Grundbetriebszeiten als auch Heuristiken aus. Das bedeutet, dass während die Ausnahme aktiv ist weder Grundbetriebszeiten noch Heuristiken den Zustand eines Clients ändern können.

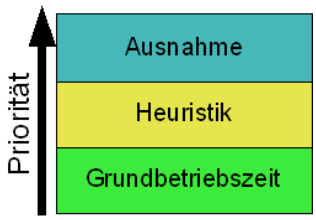


Abbildung 4.2: Priorität der einzelnen Kontrollschichten

4.4 Kommunikation

Die zu verwaltenden Computer müssen auf der einen Seite vom Powerman-Server gesteuert werden. Auf der anderen Seite müssen sie regelmäßig Status-Informationen (z.B. CPU-Auslastung) bereitstellen, die der Server als Entscheidungsgrundlage für weitere Aktionen benutzt. Die Schnittstelle auf den Clients, die die geforderten Informationen und Funktionen bereitstellt, soll auf allen Client-Betriebssystemenen identisch und erweiterbar sein.

Hier gab es für uns zwei Möglichkeiten: Die Verwendung einer generischen Schnittstelle (SNMP) oder die Programmierung eines eigenen Daemons. Wir haben uns für SNMP entschieden, da ein eigener Daemon zuviele Nachteile bezüglich Wartbarkeit, Aufwand, Erweiterbarkeit, Portabilität und Sicherheit gebracht hätte. Das Open-Source-Projekt „Net-SNMP“ hat sich dabei als ideale Kommunikationsschnittstelle erwiesen: Das Projekt wird von einer großen Community gepflegt und aktualisiert, was den Einsatz selbst auf unternehmenskritischen Server rechtfertigt. Zudem wurde es für zahlreiche verschiedene Plattformen entwickelt.

Die Kommunikation zwischen SNMP-Client und Powerman-Server geschieht dabei nicht über die Standard-UDP-Ports 161 und 162. Wir möchten eventuell vorhandene SNMP-Agents nicht stören und verwenden daher die unprivilegierten Ports 4123 (SNMP Abfragen) und 4124 (SNMP Traps). Die Kommunikation geschieht anstelle dem sonst üblichen UDP über TCP. Das hat den Vorteil, dass selbst in größeren Netzwerken mit viel Traffic sichergestellt ist, dass die Kommunikation zuverläsig bleibt und Befehle ausgeführt werden. Jedoch ergibt sich auch der Nachteil, dass der Server nicht einfach Anweisungen verschicken und dann weiter arbeiten kann. Bei TCP muss der Server warten, bis das Paket entweder zugestellt wurde oder eine bestimmte Zeitspanne (Timeout) abgelaufen ist.

Da Powerman-Clients spezielle Funktionen und Informationen bereitstellen müssen, haben wir hierzu der MIB-Baum erweitert und in einen separaten Baum eingegliedert. Bis zu diesem Punkt haben wir eine generische Kommunikation, d.h. sie funktioniert für alle Clients gleich. Die untere Schicht, das tatsächliche Auslesen von Systemstatusinformationen sowie die Ausführung von Befehlen, ist clientabhängig und muss einmal für jedes Betriebssystem entwickelt werden.

Zuletzt muss noch das An-/Abmelden eines Clients betrachtet werden. Jeder Client muss sich beim Hochfahren am Powerman-Server anmelden, um seinen Online-Status zu signalisieren, denn sonst hat der Server keine Möglichkeiten herauszufinden, ob und welche Clients aktiv sind. Das geschieht durch einen SNMP-Trap, den der Client an den Server sendet. Daraufhin registriert der Server den Client als „an“ und fragt in regelmäßigen Abständen (5 Minuten-Takt) Systeminformationen von dem Client ab. Diese Informationen können dann als Entscheidungsgrundlage für Heuristiken verwendet werden. Beim Herunterfahren sendet der Client ebenfalls einen Trap an den Server. Der Server nimmt den Client aus der Liste der aktiven Clients und stoppt seine regelmäßigen Abfragen.

Schickt ein Client keinen Trap an den Server, so liegt wahrscheinlich ein Problem vor, welches Benutzerinteraktion am Client erfordert.

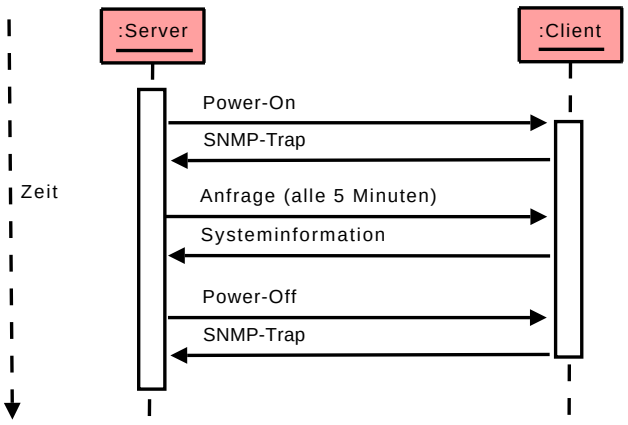


Abbildung 4.3: Skizze der Kommunikation zwischen Client und Server

4.5 Powerman-MIB

Für eigene SNMP Erweiterungen steht ein eigener Teilbaum (iso.org.dod.internet.private.enterprise = 1.3.6.1.4.1) in der Internet-MIB zur Verfügung. Man kann jederzeit einen eigenen Teilbaum für sich reservieren und eintragen lassen. Dies geschieht auf der Webseite der Internet Assigned Numbers Authority (IANA) [1]. Ebenso kann man dort eine vollständige Liste mit reservierten Teilbäumen abrufen[2]. Für unsere Zwecke wollten wir keine eigene Enterprise-OID beantragen und haben uns entschlossen einen Unterteilbaum einer OID, die für die Technische Universität München registriert ist, zu nutzen (enterprise OID 19518). Unter dieser OID bewegen wir uns im Knoten 4123 (entspricht unserem genutzen TCP-Port). Somit lautet der vollständiger Pfad zu unserem Wurzelknoten
1.3.6.1.4.1.19518.4123.

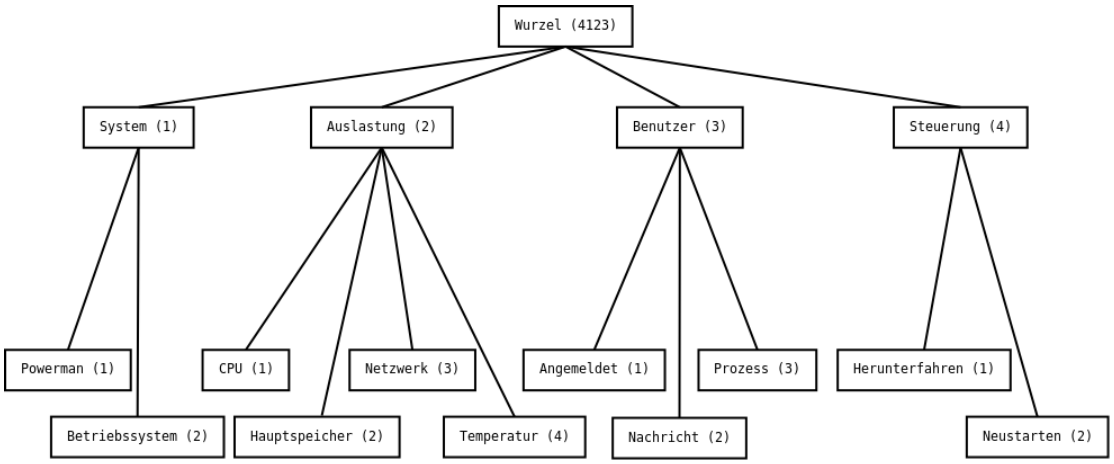


Abbildung 4.4: Der Powerman oid-Baum, die oid des jeweiligen Knoten steht in Klammern

Der Teilbaum unseres Net-SNMP Moduls enthält ein Blatt für jede von uns geforderte Information und Funktion. Die Forderungen sind kurz in Tabelle 4.3 zusammengefasst und werden in Kapitel 4.6 behandelt. Ein zusätzliches Blatt enthält den Namen des Moduls. Wir unterteilen die Blätter in vier Kategorien.

1. Die erste Kategorie enthält lediglich Information über den Agent, nämlich den Namen des Powerman Moduls und das Betriebssystem auf dem er läuft. Die Information über das Betriebssystem existiert auch ohne unser Modul im Net-SNMP, allerdings viel umfangreicher, als wir es für unsere Zwecke benötigen, deswegen implementieren wir eigene Funktionen dafür.

2. Die zweite Kategorie von Informationen gibt Auskunft über die Auslastung des Systems. Hier finden sich CPU-, Hauptspeicher- und Netzwerkauslastung, zusammen mit der vom Computer gemessenen Temperatur.

3. Kategorie drei sind Informationen und Funktionen, die bereits direkt den Benutzer treffen. Hier findet man die Anzahl der im Moment eingeloggten Benutzer. Diese Funktion zählt nicht Benutzernamen, sondern alle offenen Sitzungen. Dazu zählen Konsolen, VNC-Verbindungen, Windows- und X-Sitzungen. Zur dritten Kategorie zählt außerdem das Auslesen und setzen der Systemnachricht, welche dem Benutzer bei jedem Anmelden an einer Konsole, oder unter Windows als Hintergrundbild angezeigt wird. Das letzte Blatt der dritten Kategorie ermöglicht das Abfragen ob ein bestimmter Prozess läuft.

4. Die vierte Kategorie enthält die Steuerfunktionen zum Herunterfahren und Neustarten des Computers, keine Informationen. Auf diese Kategorie kann deshalb nur schreibend zugegriffen werden.

[1] http://www.iana.org/
[2] http://www.iana.org/assignments/enterprise-numbers

4.6 Powerman-Clientsoftware

Die Powerman-Clientsoftware muss folgende Informationen und Funktionen zur Verfügung stellen:

Information	**Beschreibung**
Betriebsystem	Das auf dem Client installierte Betriebssystem mit Versionsangabe
CPU-Auslastung	Die durchschnittliche Auslastung der CPU seit der letzten Abfrage bzw. seit dem Start
Hauptspeicherauslastung	Die Auslastung des Hauptspeichers zum Zeitpunkt der Abfrage
Netzwerkauslastung	Die durchschnittliche Auslastung der Netzwerkschnittstellen seit der letzten Abfrage bzw. seit dem Start
Temperatur	Die momentan vom Client gemessene Temperatur
Eingeloggte Benutzer	Die Anzahl der momentan am Client angemeldeten Benutzer (lokal und remote)

Funktion	**Beschreibung**
Systemnachricht setzen	Eine für alle Benutzer des Clients sichtbare Nachricht
Prozess überprüfen	Prüft, ob momentan ein Prozess mit dem angegeben Namen läuft
Neustarten	Startet den Computer neu
Herunterfahren	Fährt den Computer herunter

Tabelle 4.3: Anforderungen an die Powerman Clientsoftware

4.7 Powerman-Serversoftware

Die Clients sind nach der Anmeldung am Server zu 100% passiv, das heißt, dass der Server die Informationen von den Clients einholen und verarbeiten muss. Aufgrund der gesammelten Daten trifft der Server Entscheidungen und schickt gegebenenfalls Anweisungen an die Clients. Wie zuvor in Kapitel 4.2 aufgeführt, ist die Powerman-Serversoftware nur ein abstrakter Name für ein ganzes Paket an Software.

4.7.1 Datenbank

Der Powerman soll auf einem gewöhnlichen x86-basiertem System installiert werden. Dies hat den Vorteil, dass wir relativ losgelöst von Performance-Angelegenheiten agieren können, da heute x86-kompatible Rechner ausreichend schnell sind. Unser Zielbetriebssystem ist Linux, wodurch wir aus einem großen Angebot von Datenbanksystemen wählen können. Auf Wunsch des Auftraggebers hin benutzen wir im Folgenden SuSe Linux. Da wir keine Auflagen seitens der Planung haben, beschränken wir unsere Auswahl auf MySQL und PostgreSQL. Seit Version 5.1 erfüllt MySQL sehr viele Voraussetzungen, die man von einem Datenbanksystem erwartet, wie zum Beispiel Foreign Keys. Unser Ziel ist die Datenbank so aufzusetzen, dass sie sich selbstständig Konsistent hält. Dafür benötigen wir vor allem Foreign Keys, Triggers und stored Procedures. Da MySQL diese Funktionen inzwischen auch bietet sind wir weitestgehend indifferent gegenüber den beiden Alternativen. Aufgrund von persönlichen Erfahrungen mit den verschiedenen GUIs für diese Systeme und der möglichkeit der prozedualen Programmierung bevorzugen wir PostgreSQL.

Entity	Inhalt	Besonderheit bei Implementierung
Shutdown	Zeitpunkt an dem die mit einem Rechner assoziierte Managementkomponente den Rechner abschaltet	-
Characteristic	Stromverbrauchscharakteristika	-
Heuristik	Heuristiken	Die Menge der zuweisbaren Heuristiken ist im Code vorgegeben. Gespeichert werden nur Zuweisungen.
Socketoutlet	Managementkomponenten	-
Monitoring	Für Heuristiken relevante von Clients bezogenen Daten	-
User	Benutzer für Zugang zum Frontend	-
Log	Aktivitäten der Benutzer im Frontend	-
Policy	Namen von Grundbetriebszeiten und Ausnahmen	-
Time	Betriebszeiten mit Wochentag	-

Tabelle 4.4: Entities der Powerman-Datenbank

Ein ER-Diagram wäre aufgrund der vielen Attribute relativ unübersichtlich. Deshalb sind in Abbildung 4.5 Attribute und Entities zu Listen zusammengefasst. Außerdem sind die Attribute der Vollständigkeit halber schon wie in den resultierenden Tabellen typisiert. Die Tabelle Systemconf ist wie man der Abbildung entnehmen kann vollkommen unabhängig. Das Verhalten des Powerman-Systems kann über Systemvariablen beeinflusst werden (siehe B.5). Diese Systemvariablen werden in der Tabelle Systemconf gespeichert. Das Powermansystem kann auch Daten von Sensoren (z.B. Temperatursensoren) speichern und aufbereiten. Das Speichern übernehmen die beiden, ebenfalls vom eigentlichen System unabhängigen, Tabellen Sensor und Sensordata. Die Datenbank spiegelt den von uns geforderten modularen Aufbau wieder. An der Wurzel steht eine Menge von Rechnern in der Tabelle Machine. Diese Tabelle hält nur Informationen über Rechner und bietet damit die Grundlage für alles weitere. Diese Tabelle speichert die für die Entwicklung am wichtigste Eigenschaft, die mid. Die mid ist eine intern einem Rechner zugewiesene eindeutige Nummer. So kann von jeder Position innerhalb des Systems ein Rechner eindeutig identifiziert und angesprochen werden. Die restlichen Entities und ihre Bedeutung sind in Tabelle 4.4 aufgelistet. Die Tabelle Monitoring speichert die per SNMP bezogenen Daten (vgl. Teilbäume 2 und 3 in Powerman-MIB, Abbildung 4.4). Relationen, die ein Element einer Entity A (z.B. Time) mit genau einem Element einer anderen Entity B (z.B. Policy) assoziieren, ist die Relation in den resultieren Tabellen als Foreign Key Feld in A implementiert. Kann jedoch ein Element einer Entity (z.B. Policy) mehreren anderen Elementen einer anderen Entity (z.B. Machine) zugewiesen werden, so ist die Relation als eigene Tabelle mit Foreign Keys implementiert.

4.7.2 Command Line Interface

Das Command Line Interface (CLI) stellt eine Schnittstelle auf Konsolenbasis zur Verfügung. Mit ihr ist es möglich, dass z.B. Remotebenutzer, die nur Konsolenzugriff besitzen, trotzdem auf die wichtigsten Funktionen des Powerman zugreifen können. Zu den wichtigsten Funktionen zählt das Ein- und Ausschalten sowie das Powercyclen von Rechnern.

4.7.3 CCU

Im Wesentlichen ist die CCU die Schnittstelle zwischen unserem abstrakten Konzept und der Hardware. Hier ist auch der Automat implementiert, der die Grundbetriebszeiten, Ausnahmen und Heuristiken aus der Daten-

bank auf Aktionen abbildet. Diese Aktionen, die auch manuell vom Benutzer durch das Frontend ausgelöst werden können, werden dann mit konkreten Steuersignalen (z.B. SNMP-Anfrage) auf den Clients ausgeführt. Umgekehrt bildet die CCU ebenfalls den tatsächlichen Zustand in die Datenbank ab. Dazu werden in regelmäßigen Abständen Informationen von den Clients eingeholt und direkt in der Datenbank gespeichert.

4.7.4 Frontend

Vom Frontend aus können Clients direkt gesteuert werden und die Kontrollschichten konfiguriert werden. Darüber hinaus kann hier einem Client eine Managementkomponente zugewiesen werden, um die Kontrolle des Powerman-Servers über den Client zuverlässiger zu machen. Das Frontend besitzt eine eigene Benutzerverwaltung, welche das Anlegen von Benutzern mit differenzierten Rechten ermöglicht. Außerdem können damit Vorgänge den einzelnen Benutzern zugeordnet werden, so dass Arbeitsschritte besser nachvollzogen werden können.

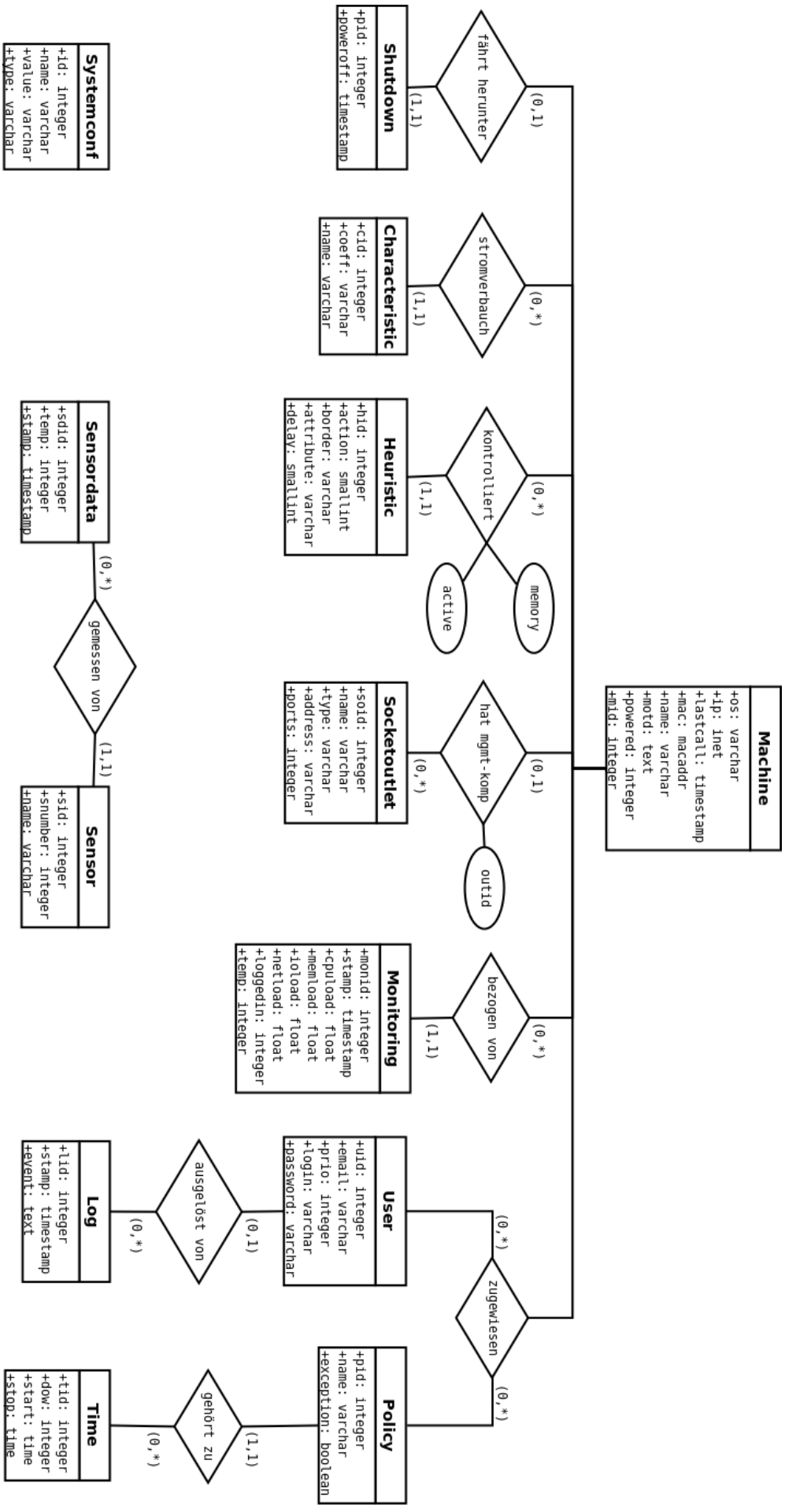


Abbildung 4.5: schematische Darstellung der Powerman-Datenbank

5 Realisierung

Mit den Konzepten aus dem vorherigen Kapitel können wir nun die Software entwickeln. Um nicht zu viel als gegeben betrachten zu müssen, entwickeln wir zunächst die Clientsoftware. Haben wir das geschafft, so ist die Grundlage für die Serversoftware gelegt, denn wir wissen schon genau welche Informationen und Funktionen wie vorliegen. Die nächste Station ist die CCU, denn die ist dafür zuständig die Steuerfunktionen der Clients aufzurufen und Informationen von den Clients zu beschaffen, welche sie in der Datenbank ablegt. Als letzte Komponente bleibt das Frontend, welches nur sinnvoll ist, wenn auch eine entsprechende Struktur vorhanden ist, die über ein Frontend Benutzern zugänglich gemacht werden soll.

5.1 Powerman Clientsoftware

Die Powerman Clientsoftware stellt neben Funktionen zum Steuern auch Informationen über den aktuellen Zustand des Client zur Verfügung (vgl. Kapitel 4.6). Wie in Kapitel 4.4 angedeutet kommt hierzu ein für diese Zwecke erweiterter SNMP-Agent zum Einsatz. Die Arbeit zur Realisierung der Clientsoftware besteht lediglich aus dem Erweitern des Net-SNMP.

Zunächst analysieren wir im folgenden Kapitel 5.1.1 die Kriterien, die für eine Erweiterung des MIB-Baums erfüllt werden müssen. Anschließend entwickeln wir im Kapitel 5.1.2 die Umsetzung unseres Teilbaums mit Hilfe der Net-SNMP API, bevor wir uns ab Kapitel 5.1.3 mit den plattformspezifischen Funktionen beschäftigen.

5.1.1 Einführung in Net-SNMP

Net-SNMP ist eine Open Source Implementierung des SNMP für Windows, Linux und andere Unix-Derivate, geschrieben in C. Der auf der Clientseite eingesetze SNMP-Agent besteht aus Modulen, welche Informationen zur Verfügung stellen und Funktionen implementieren. Der Agent organisiert diese Daten in einer Baumstruktur und macht diese über das Netzwek zugänglich. Da der Aufbau der Module einheitlich ist und zusätzliche Module die Baumstruktur beliebig erweitern können, ist Net-SNMP unsere Wahl für die Grundlage der Clientsoftware.

Vom Prinzip her ist SNMP nur darauf ausgelegt mit Daten zu arbeiten. Über das Protokoll können Daten ausgelesen, oder aber geschrieben werden. Funktionenaufrufe mit einem Konzept für Remote-Procedure-Call (RPC) sind in SNMP nicht vorgesehen. Wir werden daher unsere Funktionen hinter Daten „verstecken“, indem wir erlauben, dass bestimmte Daten in unserer Erweiterung per SNMP von einem anderen Computer aus geschrieben werden können. Für solche Fälle erlaubt es der Net-SNMP, für jedes Datenblatt eine eigene write-Funktion zur Verfügung zu stellen. Ursprünglich sind die Funktionen dazu gedacht die Datenänderung in andere Komponenten des Systems zurück zu schreiben. Wir werden sie dafür benutzen bestimmte Systemfunktionen wie. z.B. neustarten auszuführen, wenn auf das entsprechende Datenblatt lesend zugegriffen wird.

Aufbau von Net-SNMP Modulen

Ein Modul des Agents besteht im Prinzip aus der Definition eines neuen Teilbaums, dem Pfad zu dem Knoten, an dem der Teilbaum angehängt werden soll, einer Initialfunktion, die automatisch beim Laden des Moduls ausgeführt wird und mindestens einer Eintrittfunktion, die zugriffe auf Blätter des Teilbaums annimmt.

Definition des Teilbaums

Die Definition des Teilbaums erfolgt über seine Blätter. Ein Blatt besteht immer aus einem Bezeichner, aus dem Datentypen der Information, die an diesem Blatt abrufbar ist, aus den Zugriffsmöglichkeiten auf dieses Blatt, aus einem Zeiger auf eine Eintrittsfunktion und aus dem Pfad von der Wurzel des Teilbaums zum Blatt.

Eindeutiger Bezeichner

Der Bezeichner eines Blattes ist eine positive Ganzzahl und muss innerhalb des Moduls eindeutig sein. Ansonsten ist man Frei in der Wahl des Bezeichners. Er wird von der Eintrittsfunktion verwendet um Blattspezifischen Code aufzurufen.

Datentyp der Information

Ein Blatt im Teilbaum eines Moduls ist nicht unbedingt ein Blatt im Informationsbaum, der später zur Laufzeit abrufbar ist. Es gibt Datentypen, die ein Blatt zu einem inneren Knoten machen. Der Unterbaum wird dann direkt durch die zu repräsentierende Information vorgeschrieben. Der arp-Cache zum Beispiel, ist ein Blatt in einer Moduldefinition, jedoch bei nicht leerem ARP-Cache kein Blatt des Informationsbaums. Für unsere Zwecke benötigen wir nur primitive Datentypen, genauer benötigen wir nur Integer und String (Array vom Typ char), und werden deshalb nicht weiter auf die Möglichkeiten des Net-SNMP an dieser Stelle eingehen. Die von uns verwendeten Datentypen erweitern den Baum um genau einen Knoten. Aus einem im Modul definierten Blatt mit Informationstyp Integer oder String wird im Informationsbaum immer ein innerer Knoten mit genau einem Kind, welches ein Blatt im Informationsbaum ist.

Zugriffsmöglichkeiten auf Information

Es gibt genau zwei Zugriffsmodi für Information im Net-SNMP Agent. Man kann Information als read-only oder als read-write deklarieren. Versucht man schreibend auf read-only Information zuzugreifen, so lehnt der Agent die Anfrage explizit ab. Information, die auch schreibenden Zugriff zulässt, wird häufig benutzt, um Systemfunktionen zu starten. Das ist auch die Methode, die wir benutzen werden, um Funktionen auf den Clients zu starten, deren Aufgabe nicht das Sammeln von Informationen ist.

Eintrittsfunktionen

Eine Anfrage an einen SNMP-Agent kann beliebig viele Informationen abfragen. Empfängt der SNMP-Agent eine Anfrage, so ruft er sequenziell für jedes abgefragte Blatt die zugehörige Eintrittsfunktion (get-Handler) aus. Dies ist Möglich, da die Signatur von Eintrittsfunktionen im Net-SNMP vorgegeben ist. Die Eintrittsfunktion ist der Punkt an dem der Agent den Kontrollfluss an das Modul übergibt. Man kann für alle Blätter eines Moduls eine eigene Eintrittsfunktion definieren, dies ist jedoch nicht notwendig. Der Eintrittsfunktion wird beim Aufruf der eindeutige Bezeichner des abgefragen Blattes übergeben. Mit dessen Hilfe kann dann eine Fallunterscheidung durchgeführt werden kann, was zu strukturierterm und überischtlicherem Code führt, als eine eigene Eintrittsfunktion für jedes Blatt. Für Blätter, auf die auch schreibend zugegriffen werden kann, muss die Eintrittsfunktion bei jedem Aufruf den Funktionszeiger writemethod setzen, welcher auf die Funktion zeigt, die für den Schreibzugriff benutzt werden soll. Die Eintrittsfunktion muss einen Zeiger vom Typ u-char oder NULL zurück geben. Dieser Zeiger zeigt auf die Daten, die der SNMP-Agent als Anfrageergebnis zurückgibt.

Schreibfunktionen

Für den Schreibvorgang sieht der Net-SNMP mehrere Schritte vor. Theoretisch sind folgende Schritte zu implementieren.

1. Überprüfen der Daten, die geschrieben werden soll

2. Bereitstellen der Ressourcen, die für die Durchführung des Schreibvorgangs benötigt werden
3. Übernehmen der Daten in den Agent
4. Festsetzen der Änderungen
5. Rückgängig machen der Änderungen (nur wenn Übernehmen oder Festsetzen fehlschlägt)
6. Bereitgestellte Ressourcen freigeben

Das bedeutet für uns, dass eine Schreibfunktion im Normalfall fünf mal ausgeführt wird und wir das berücksichtigen müssen. Da unser Modul an dieser Stelle keine komplexen Arbeiten durchführen wird, werden wir nie alle Schritte implementieren.

Außerdem sind Schreibfunktionen nicht so einheitlich implementierbar wie Eintrittsfunktionen, weshalb man für jedes beschreibbare Blatt eine eigene Schreibfunktion definieren muss.

Pfad zur Teilbaumwurzel

Um einen Knoten im Informationsbaum eindeutig identifizieren zu können, benutzt SNMP ein Object-Identifier-Konzept (oid-Konzept) in dem jeder Knoten mit einer Ganzzahl benannt wird. Die einzige Anforderung ist, dass die Namen von Knoten, die den selben Elternknoten haben paarweise verschieden sein müssen. Die Namen von Knoten im Informationsbaum haben nichts mit den eindeutigen Bezeichnern von Blättern zu tun. Da der Name eines Knotens diesen von seinem Elternknoten aus eindeutig identifiziert, beschreibt eine Liste von Knotennamen einen Pfad im Informationsbaum. Ein so beschriebener Pfad von der Wurzel aus heißt Object-ID (oid) und identifiziert einen Knoten im Informationsbaum eindeutig. Bei SNMP werden die Knotennamen durch einen $\gg$. $\ll$ getrennt und die Wurzel wird weggelassen, da sie Teil einer jeden oid ist. Die zweite Komponente eines funktionsfähigen Moduls im Net-SNMP Agent ist also die oid der Wurzel des Moduls. Eine zusätzliche Benennung von Knoten mit Strings als Namen macht es leichter ihre Bedeutung zu verstehen, wird jedoch nur außerhalb des Agents in den sogenannten MIBs unterstützt. Für die Funktionalität des Agents ist diese Benennung jedoch unerheblich.

Initialfunktion des Moduls

Außer der Signatur ist auch der Name der Initialfunktion fest vorgegeben. Der Name muss init-$< modulname >$ sein, die Funktion selber erwartet keine Argumente und liefert auch keinen Rückgabewert. Ihre einzige Aufgabe ist das Einhängen des Moduls in den Informationsbaum. Deswegen muss sie vorhanden sein, damit ein Modul im Agent funktionieren kann.

5.1.2 Das Powerman Net-SNMP Modul

Die eben aufgezählten Eigenschaften sind die Grundbausteine eines Net-SNMP Moduls und der Quellcode dafür enthält noch keinen plattformspezifischen Code.

Den plattformspezifischen Code möchten wir in unserem Ansatz in Funktionen ausgliedern, die auf allen Plattformen die gleiche Signatur haben. Das erlaubt uns die Implementierung in zwei Teile auf zu teilen. Der erste Teil ist die Implementierung der oben aufgeführten Grundbausteine. Da wir auf allen Zielsystemen Net-SNMP einsetzen ist dieser Code portabel. Der zweite Teil ist die Implementierung der plattformspezifischen Funktionen. Da die Signaturen der Funktionen des zweiten Teils einheitlich definiert sind, muss zur Übersetzungszeit nur bekannt sein welche Implementierung des zweiten Teils verwendet werden soll.

Dieser Ansatz folgt dem Prinzip der Vererbung aus der objektorientierten Programmierung. In Objekten ausgedrückt ist das Net-SNMP Modul ein Objekt der Klasse CTum, welches eine Referenz auf Objekt der Klasse CPowermanProto enthält und mit dessen Methoden plattformspezifische Routinen ausführt. Also muss man zum Portieren des Powerman Moduls nur eine neu Klasse von CPowermanProto ableiten, die die plattformspezifischen Methoden implementieren, der Rest des Moduls bleibt unberührt.

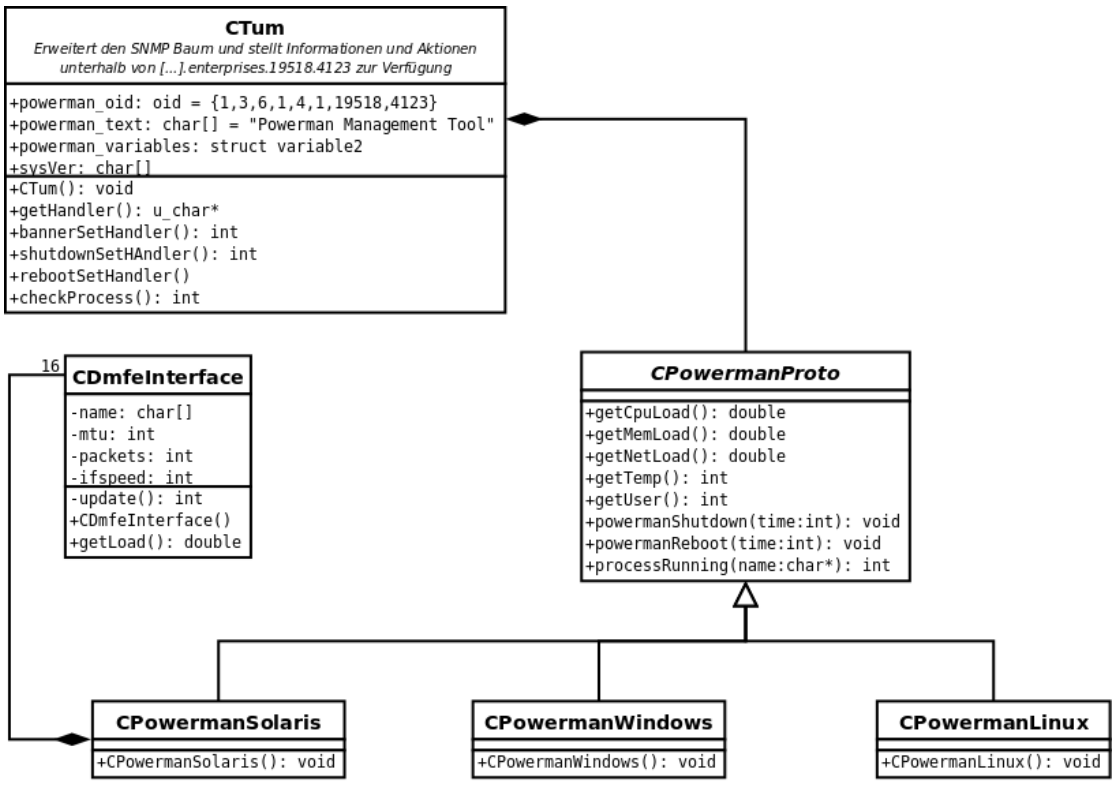


Abbildung 5.1: Das Powerman Net-SNMP Modul zu Objekten abstrahiert

Da Net-SNMP in C geschrieben ist, wurde auch unsere Erweiterung in C geschrieben. Dadurch lässt sich das Konzept mit Objekten nicht exakt umsetzen. Wie aus dem UML-Diagramm in Abbildung 5.1 ersichtlich, enthält weder die Klasse CPowermanProto, noch eine der daraus abgeleiteten Klassen Daten. Der einzige Zugriff auf diese Objekte ist also das Ausführen ihrer Methoden. Anstatt diese Funktionen in Objekten zu sammeln haben wir in C nur eine Menge an Funktionen. Diese Funktionen werden meistens von der Eintrittsfunktion aufgerufen. Definiert werden sie in plattformspezifischen Dateien, welche je nach Bedarf in der Hauptdatei eingebunden werden.

Eindeutiger Bezeichner

Da der eindeutige Bezeichner eines Blattes lediglich eine eindeutige Ganzzahl sein muss, definieren wir Konstanten, um uns den Umgang mit den Bezeichnern zu erleichtern. Als Zahlen wählen wir die Nummerierungen der letzten beiden Knoten der oid.

Listing 5.1: Eindeutige Bezeichner aus powerman.h

```
/* Kategorie 1 */
#define POWERMAN        11      /* Powerman Information */
#define VERSION         12      /* Betriebssystem */

/* Kategorie 2 */
#define CPU_LOAD        21      /* CPU-Last */
#define MEM_LOAD        22      /* Hauptspeicher-Last */
#define NET_LOAD        24      /* Netzwerk-Last */
#define TEMP            25      /* Temperatur */

/* Kategorie 3 */
#define LOG_USERS       31      /* Angemeldete Benutzer */
#define BANNER          32      /* Systemnachricht */
```

```
#define PROCESS          33      /* Prozessstatus */

/* Kategorie 4 */
#define SHUTDOWN         41      /* Herunterfahren */
#define REBOOT           42      /* Neustarten */
```

Datentypen der Informationen

SNMP benutzt die Abstract Syntax Notation 1 (ASN.1) zur Beschreibung der Datentypen, deren Handhabung und wie die Daten als Knoten in den Baum eingefügt werden. Während die Daten in der Implementiertung sehr wohl die bekannten Datentypen sind, muss den Daten im Informationsbaum ein abstrakter Datentyp zugewiesen werden. Da wir in unserem Fall nur String und Integer benutzen, sind die für uns relevanten Datentpyen ASN_OCTET_STR und ASN_INTEGER.

Zugriffsbeschränkungen

Wie in Kapitel 5.1.1 angesprochen benutzen wir Schreibzugriffe um Funktionen auszulösen. Auf die Blätter, die den Funktionen aus Tabelle 4.3 entsprechen darf lesend und schreibend (RWRITE) zugegriffen werden, auf alle anderen Werte lediglich lesend (RONLY).

Eintrittsfunktionen

Die Funktionsweise von get-Handlern wird in Kapitel 5.1.1 erläutert. Da wir nun bereits die eindeutigen Bezeichner definiert haben, können wir nun eine get-Handler Funktion schreiben. Die Signatur ist von der Net-SNMP API vorgegeben. Der Funktionsrumpf besteht im Prinzip aus nur einem switch-Statement.

Listing 5.2: getHandler() gekürzt

```
u_char*
getHandler(struct variable* vp, oid* name, size_t* length, int exact,
                                size_t* var_len, WriteMethod** write_method)
{
        switch(vp->magic){
                case VERSION:
                        *var_len = strlen(sysVer);
                        return (u_char*) sysVer;
                break;

                case TEMP:
                        logged = getTemp();
                        *var_len = sizeof(int);
                        return (u_char*) &logged;
                break;

                case REBOOT:
                        *write_method = rebootSetHandler;
                break;

        }
return NULL;
}
```

Jeder Zugriff, ob lesend, oder schreibend, löst auf jeden Fall zuerst seine Eintrittsfunktion aus. Die Variable `vp→magic` enthält den Bezeichner des Blattes, welches abgefragt werden soll. In Verbindung mit einem switch-Statement lässt sich innerhalb der Funktion blattspezifischer Code auszuführen, wodurch wir nur eine einzige Eintrittsfunktion benötigen. Die Funktion getHandler() gibt immer einen Zeiger vom Typ u_char

zurück. Wie die Daten zu interpretieren sind, weiß der Agent aus der Zuweisung des ASN.1 Datentyps (siehe Kapitel 5.1.2). Damit der Agent weiß, wieviel Information zurückgegeben wurde, müssen wir, bevor die Funktion terminiert, noch den Inhalt von var_len richtig setzen. Der erste Fall (case VERSION) zeigt, wie wir einen im Agent hinterlegten String zurückgeben, während der zweite Fall (case TEMP) einen int-Wert zurückgibt, der von einer plattformspezifischen Funktion (siehe Kapitel 5.1.3) berechnet wird.

Schreibfunktionen

Der dritte Fall aus Listing 5.2 (case REBOOT) ist komplett verschieden zu den beiden vorherigen Fällen. Dieses Beispiel soll illustrieren, wie Schreibzugriffe zustande kommen. Damit der Schreibzugriff vollzogen werden kann, muss der Funktionszeiger write_method sinnvoll mit einer Schreibfunktion belegt werden. In diesem Beispiel zeigt write_method auf die Funktion rebootSetHandler. Nachdem die Eintrittsfunktion terminiert hat, führt der SNMP-Agent die write_method-Funktion wie in Kapitel 5.1.1 beschrieben aus.

Listing 5.3: rebootSetHandler()

```
int
rebootSetHandler(int action, u_char * var_val, u_char var_val_type,
                 size_t var_val_len, u_char * statP, oid * name, size_t name_len)
{
    switch (action) {
    case RESERVE1:
        if (var_val_type != ASN_OCTET_STR) {
            DEBUGMSGTL(("powerman", "%x not octet_str type", var_val_type));
            return SNMP_ERR_WRONGTYPE;
        }
        if (var_val_len > SHUTDOWN_MAXSIZE) {
            DEBUGMSGTL(("powerman", "wrong length %x", var_val_len));
            return SNMP_ERR_WRONGLENGTH;
        }

        break;

    case ACTION:
                                shutdown_int = atoi(var_val);
        break;


    case COMMIT:
                if (shutdown_int==-1) {
                        system(CMDABOR);
                } else {
                powermanReboot(shutdown_int);
                }
        break;

    default:
        break;

    }
    return SNMP_ERR_NOERROR;
}
```

Wie bereits am Anfang des Kapitels erwähnt, löst jeder Zugriff, egal welcher Art, getHandler() aus. Ein Lesezugriff auf REBOOT wird nie ein Ergebnis liefern. Das liegt daran, dass bei einem Zugriff auf REBOOT die Rückgabe von getHander() NULL ist.

Pfad von der Teilbaumwurzel zum Blatt

Wie in Abbildung 5.1 angedeutet, ist die oid des Teilbaums eine selbstständige Variable.

Listing 5.4: Powerman oid

```
oid powerman_oid[] = {1,3,6,1,4,1,19518,4123};
```

Somit muss für die einzelnen Blätter nur noch das Suffix der oid, ensprechend Abbildung 4.4 gespeichert werden. Zusammen mit den vorherigen Informationen ergibt sich die Datenstruktur „powerman_variables“, welche den gesamten Teilbaum beschriebt.

Listing 5.5: Eindeutige Bezeichner aus powerman.h

```
struct variable2 powerman_variables[]={
        {POWERMAN,  ASN_OCTET_STR, RONLY,  getHandler,2,{1,1}},
        {VERSION,   ASN_OCTET_STR, RONLY,  getHandler,2,{1,2}},
        {CPU_LOAD,  ASN_OCTET_STR, RONLY,  getHandler,2,{2,1}},
        {MEM_LOAD,  ASN_OCTET_STR, RONLY,  getHandler,2,{2,2}},
        {NET_LOAD,  ASN_OCTET_STR, RONLY,  getHandler,2,{2,4}},
        {TEMP,      ASN_INTEGER,        RONLY,  getHandler,2,{2,5}},
        {LOG_USERS, ASN_INTEGER,   RONLY,  getHandler,2,{3,1}},
        {BANNER,    ASN_OCTET_STR, RWRITE, getHandler,2,{3,2}},
        {PROCESS,       ASN_OCTET_STR,  RWRITE, getHandler,2,{3,3}},
        {SHUTDOWN,  ASN_OCTET_STR, RWRITE, getHandler,2,{4,1}},
        {REBOOT,                ASN_OCTET_STR, RWRITE, getHandler,2,{4,2}}
};
```

Das Struct beschreibt zeilenweise ein Blatt des Teilbaums. Von links nach rechts gelesen stehen in einer Zeile die vordefinierten Konstanten als eindeutige Bezeichner, der Datentyp, die Zugriffsmöglichkeiten, die Eintrittsfunktion, die Tiefe des Blattes im Teilbaum und die das Suffix der oid als Array von Ganzzahlen. Auf die Tiefe des Blattes sind wir vorher nicht eingegangen, da dieser Wert ein Detail der Net-SNMP API ist und für das Konzept keine Bedeutung hat.

Prinzipiell ist diese Datenstruktur vom Typ „struct variableN“, wobei N die betragsmäßig größte Tiefe im Teilbaum ist; in unserem Fall ist sie vom Typ „struct variable2“.

5.1.3 Plattformspezifische Funktionen

Die plattformspezifischen Funktionen sind der in Kapitel 5.1.2 beschriebene zweite Teil der Implementierung. Eine Übersicht über die zu implementierenden Funktionen gibt nachfolgende Tabelle. Damit wir uns nicht in die einzelnen APIs der Betriebssysteme einarbeiten müssen, haben wir uns dazu entschieden, die Informationen über bereits vorhandene Programme zu beziehen. Alle Funktionen setzen einen Shellaufruf ab und geben ggf. dessen Ausgabe zurück.

5.1.4 Implementierung unter Solaris

Listing 5.6 zeigt alle Shellaufrufe, die für Solaris hinterlegt sind. Nicht in allen Fällen reicht ein Befehl um die gewünschte Information zu erhalten.

Listing 5.6: getTemp()

```
#define VERSION_CMD "/usr/bin/uname -a | awk '{print $1 \" \" $3 \" \" $4}'"
#define CPULOAD_CMD "/usr/bin/mpstat 1 2 | grep -v CPU | awk 'BEGIN {cpus=0;summe
    =0;}{arr[$1]=$16}END{for (x in arr)summe=summe=arr[x];for (x in arr)cpus=cpus
    +1;print 100-(summe/cpus);}'"

#define MEM_CMD_FREE "/usr/bin/vmstat 1 2 | tail -1 | awk '{print $5;}'"
```

Funktion	Beschreibung
void init_powerman_specific()	Plattformspezifische Initialisierungsfunktion
double getCpuLoad()	Durchschnittliche Auslastung der CPUs in Prozent
double getMemLoad()	Hauptspeicherauslastung laut Kernel in Prozent
double getNetLoad()	Durchschnittliche Auslastung der Netzwerkschnittstellen in Prozent
int getTemp()	Temperatur des Systems in Grad Celsius
int getUser()	Anzahl der am System angemeldeten Benutzer
void powermanShutdown(int time)	Veranlasst das Betriebssystem in time Minuten herunter zu fahren
void powermanReboot(int time)	Veranlasst das Betriebssystem in time Minuten neu zu starten
int processRunning(char* name)	Läuft Prozess name? 0 für nein, 1 für ja

Tabelle 5.1: Deklararion der plattformspezifischen Funktionen

```
#define MEM_CMD_TOTAL "/usr/sbin/prtconf | grep Memory | awk '{print $3;}'"

#define TEMP_CMD "/usr/sbin/lom -t 2>/devnull | /usr/bin/grep Enclosure | /usr/
    bin/awk '{print $3;}'"

#define SHUTDOWN_CMD "/usr/sbin/shutdown -y -g %d -i %i %s &"

#define CMDABOR "kill -9 `ps -ef | grep /usr/sbin/shutdown | awk '{print $2;}'`;
    rm /etc/nologin"

#define WHO_CMD "who | awk 'BEGIN {summe=0;} {summe = summe + 1;} END {print
    summe;}'"
#define NET_CMD "netstat -in | egrep -v '(lo|Name)'"
#define NET_SINGLE_CMD "netstat -in -I %s | egrep -v '(lo|Name)'"
#define NET_SPEED_CMD "kstat -p |grep %s | grep ifspeed | awk '{print $2;}'"

#define PROC_CMD "ps -ef | grep %s | grep -v grep"
```

Für die Hauptspeicher- und Netzwerkauslastung weden mehrere Befehle benutzt. Das kommt daher, dass wir zur Bestimmung der Auslastung des Hauptspeichers zwei verschiedene Programme benutzen. Für Netzwerkschnittstellen gibt es mehrere Befehle, da sich die Auslastung über gesendete und empfangene Bytes (NET_SINGLE_CMD) und die Schnittstellengeschwindigkeit (NET_SPEED_CMD) berechnet. Für die Betriebssystem-Version, die CPU-Auslastung, die Temperatur sind je nur ein Befehl notwendig und in ihrer Implementierung bis auf Namensgebungen identisch, deswegen werden wir repräsentativ nur getTemp() in Listing 5.8 aufnehmen.

Listing 5.7: Restliche Definitionen für die solarisspezifische Implementierung

```
#define INIT_HALT 5
#define INIT_REBOOT 6

#define DMFE_MAX 16
#define DMFE_NAME 8

typedef struct dmfe_interface {
        char name[DMFE_NAME];
        unsigned int    mtu;
        unsigned int    packets;
        unsigned int    ifspeed;
} dmfe_interface;
```

Listing 5.8: Powerman Solarisimplementierung

```
dmfe_interface* interfaces[DMFE_MAX];
double netLoad[DMFE_MAX];

void init_powerman_specific(){

        FILE* version, *some_fp;
        char powerman_tmp[BANNER_MAXSIZE];
        int bytesRead;
        unsigned int c;

        dmfe_interface* curr_iface;

        /*
         * Betriebssystem Information einlesen
         */
        version = popen(VERSION_CMD,"r");
        bytesRead = fread(powerman_tmp,1,BANNER_MAXSIZE,version);
        fclose(version);

        strncpy(sysVer,powerman_tmp,strlen(powerman_tmp));
        sysVer[bytesRead-1] = '\0';
        printf("Version: %s\n",sysVer);

        /*
         * Schnittstellen suchen und Counter initialisieren
         */
        for (c = 0; c < DMFE_MAX; interfaces[c++] = NULL){}
        c = 0;

        version = popen(NET_CMD,"r");
        while (!feof(version) && c < DMFE_MAX){
                // Datenstruktur anlegen
                curr_iface = (dmfe_interface*) malloc(sizeof(dmfe_interface));
                bytesRead = fread(powerman_tmp,1,BANNER_MAXSIZE,version);
                powerman_tmp[bytesRead] = '\0';
                char *tmp;
                tmp = strtok(powerman_tmp," ");
                printf("tmp: '%s' %d\n",tmp,strlen(tmp));
                strncpy(curr_iface->name,tmp,strlen(tmp));
                // mtu = Anzahl Bytes pro Paket
                curr_iface->mtu = atoi(strtok(NULL," "));
                strtok(NULL," ");strtok(NULL," ");
                // packets = Anzahl gesendete und empfangene Pakete
                curr_iface->packets = atoi(strtok(NULL," "));
                strtok(NULL," ");
                curr_iface->packets += atoi(strtok(NULL," "));

                curr_iface->name[strlen(tmp)] = '\0';
                printf("tmp: '%s' %d\ncurr_iface->name: '%s'\n",tmp,strlen(tmp),
                    curr_iface->name);
                sprintf(powerman_tmp,NET_SPEED_CMD,curr_iface->name);

                // Schnittstellengeschwindigkeit herausfinden
                some_fp = popen(powerman_tmp,"r");
                bytesRead = fread(powerman_tmp,1,BANNER_MAXSIZE,some_fp);
                powerman_tmp[bytesRead] = '\0';
                pclose(some_fp);

                // ifspeed = Schnittstellengeschwindigkeit
```

```
                curr_iface->ifspeed = atoi(powerman_tmp);

                netLoad[c] = 0.0;
                interfaces[c++] = curr_iface;
                printf("interface %d: %s\nmtu %d\npackets: %d\n",c,curr_iface->
                    name,curr_iface->mtu,curr_iface->packets);
        }

        // restliche Felder von interfaces[] mit NULL füllen
        while (c < DMFE_MAX){
                interfaces[c++] = NULL;
                netLoad[c] = 0.0;
        }
        pclose(version);
        printf("init_powerman_specific() done\n\n");
}

double getMemLoad(){
        double memload = 0;
        int bytesRead;

        FILE *stat;
        unsigned int free,total;

        char buffer[BANNER_MAXSIZE];

        stat = popen(MEM_CMD_FREE,"r");
        bytesRead = fread(buffer,1,BANNER_MAXSIZE,stat);
        buffer[bytesRead] = '\0';
        pclose(stat);
        free = atoi(buffer);
        stat = popen(MEM_CMD_TOTAL,"r");
        bytesRead = fread(buffer,1,BANNER_MAXSIZE,stat);
        buffer[bytesRead] = '\0';
        total = atoi(buffer);
        total = total * 1024;

        memload = (double)(1.0 - ((double)free/(double)total));

        pclose(stat);
        return memload * 100;
}

double getNetLoad(){
        double netload = 0.0;
        FILE *stat;
        unsigned int in,out;
        int bytesRead;
        char buffer[BANNER_MAXSIZE];
        int c,d;

        dmfe_interface* curr_iface;

        char cmd[BANNER_MAXSIZE];
        d = 0;
        /*
         * Einmal über alle Interfaces iterieren
         */
        for (c = 0; c < DMFE_MAX; c++){
                curr_iface = interfaces[c];
                if (curr_iface == NULL){
```

```
                        continue;
                }

                d++;

                sprintf(cmd,NET_SINGLE_CMD,curr_iface->name);

                stat = popen(cmd,"r");
                bytesRead = fread(buffer,1,BANNER_MAXSIZE,stat);
                buffer[bytesRead] = '\0';
                if (strlen(buffer) > strlen(curr_iface->name)){
                                        strtok(buffer," ");
                                        strtok(NULL," ");
                                        strtok(NULL," ");strtok(NULL," ");

                        in      =       atoi(strtok(NULL," "));
                                        strtok(NULL," ");
                        out=    atoi(strtok(NULL," "));

                        netLoad[d-1] =
                                (double)((in + out - curr_iface->packets) *
                                    curr_iface->mtu)
                                /
                                (double)curr_iface->ifspeed;
                        curr_iface->packets = in + out;
                }
                else netLoad[d-1] = 0.0;
                pclose(stat);
        }
        for (c = 0; c < d; c++){
                netload += netLoad[c];
        }
        return (netload / d) * 100.0;
}

int getTemp(){
        int temperature = 0;
        char* strLoad;
        FILE *stat;
        int bytesRead;

        char buffer[BANNER_MAXSIZE];

        stat = popen(TEMP_CMD,"r");
        bytesRead = fread(buffer,1,BANNER_MAXSIZE,stat);
        buffer[bytesRead] = '\0';
        strLoad = strtok(buffer,",");

        temperature = atoi(strLoad);
        pclose(stat);
        return temperature;
}


void powermanShutdown(int time){
        char cmd[BANNER_MAXSIZE];
        int grace_time = time * 60;
        sprintf(cmd,SHUTDOWN_CMD,grace_time,INIT_HALT,SHUTMSG);
        system(cmd);
}
```

```
void powermanReboot(int time){
        char cmd[BANNER_MAXSIZE];
        int grace_time = time * 60;
        sprintf(cmd,SHUTDOWN_CMD,grace_time,INIT_REBOOT,REBOMSG);
        system(cmd);
}

int processRunning(char* name){
        char cmd[BANNER_MAXSIZE];
        FILE *fp;
        int size;
        sprintf(cmd,PROC_CMD,name);
        fp = popen(cmd,"r");
        size = fread(cmd,1,5,fp);
        pclose(fp);
        if (size) return 1;
        return 0;
}
```

5.1.5 Implementierung unter Windows

Erweiterung des SNMP-Agenten

Eine ähnlich komfortable Erweiterung wie unter Solaris ist leider nicht möglich. Unter Windows ist es nicht möglich, dass der net-snmp Agent zur Laufzeit ein selbsterstelltes Modul nachlädt. Deswegen muss das Modul fest in den Quellcode von Net-SNMP integriert werden. net-snmp lässt sich unter Windows mittels verschiedener Tools und Umgebungen (Visual Studio .NET, Visual C++, MinGW, Cygwin) kompilieren. Die bevorzugte Variante ist Visual Studio .NET. Für die Kompilierung unter .NET stehen zwei Projektdateien zur Verfügung, der normale win32-agent und der win32sdk-agent. Der SDK-Agent ist dabei die vollständige Variante inklusive IP/TCP/UDP MIBGroups und wird von uns im weiteren Projektverlauf verwendet. Hierzu ist aber zunächst die Installation des Microsoft Platform SDK notwendig, welches von der Homepage bezogen werden kann. Bevor das Projekt aber unter .NET kompiliert, müssen die Schritte aus der mitgelieferten ReadME-Datei befolgt werden. Dazu zählt unter Anderem die Aktivierung des HAVE_WIN32_PLATFORM_SDK 1 - Flags in der Datei net-snmp-config.h. Zudem müssen die einzelnen Teilprojekte (einzelne Module, Bibliotheken und Programme des net-snmp) in genau definierten Reihenfolge gebaut werden, da alle Teilprojekte unterschiedlich voneinander abhängen.

Wie im nächsten Abschnitt erläutert wird, war es unter C nicht möglich an gewisse Systeminformationen zu kommen. Aus diesem Grund ruft der Windows Net-SNMP für die einzelnen Teile Hilfsskripte auf, die die gewünschten Ergebnisse zurückliefern. Eine Besonderheit in der Funktion init_powerman_specific ist der Zugriff auf die Windows-Registry, um an die Verzeichnisse der Powerman-Installation zu kommen. Stellvertretend für alle Wertermittlungs-Funktionen wird im Folgenden der Code für die Ermittlung der CPU-Auslastung mit angegeben.

Listing 5.9: Powerman Windowsimplementierung (auszugsweise)

```
void init_powerman_specific(){
        char powerman_tmp[BANNER_MAXSIZE];
        FILE* data;
        int len;
        char tmp[40];
        long lSize;
        size_t result;

        unsigned long lReg=255;
        struct HKEY__ *hkey;
        long lINback;
```

```
        for(len = 0; len < BANNER_MAXSIZE;len++){
                powerman_tmp[len] = '\0';
        }

        lINback = RegOpenKeyEx(HKEY_LOCAL_MACHINE,"SOFTWARE\\Net-SNMP",0,KEY_READ
            ,&hkey);
        RegQueryValueEx(hkey, "PowermanDir", NULL, NULL, powerman_tmp, &lReg);
        RegCloseKey(hkey);

        ...
}

double getCpuLoad(){
        int cpuload;
        FILE* data;
        data = _popen(WMICPU,"rb");
        fscanf_s(data,"%ld",&cpuload);
        _pclose(data);
        return (double) cpuload;
}
```

Leistungsdaten über WMI

WMI steht für Windows Management Instrumentations und ist eine generische Schnittstelle um Informationen über den Rechner, das Betriebssystem sowie die Hardware zu bekommen. WMI wird seit Windows 2000 fest in Windows integriert, für Windows 98 und NT gibt es Addon-Pakete. Somit ist WMI optimal geeignet um die Information über CPU/NET-Load, Temperatur etc. zu bekommen. Zudem gibt es für zahlreiche Programmiersprachen eine Schnittstelle zu WMI.

Leider gibt es aber keine Schnittstelle für reinen C-Code, wie es bei dem net-snmp Projekt der Fall ist. Für den Zugriff auf WMI benötigt man C++ Libaries. Dadurch war es nicht möglich, das Net-SNMP Projekt mit eingebautem WMI-Support zu kompilieren, da es nicht möglich war ein reines C-Projekt mit C++ Abhängigkeiten zu implementieren.

Aus diesem Grund mussten wir die WMI-Abfragen aus dem SNMP-Agent auslagern und als VB-Skripte programmieren. Diese VB-Skripte werden nun von dem Agent aufgerufen und bekommen den entsprechenden Wert zurückgeliefert. Eine WMI-Abfrage unter VB gestaltet sich immer sehr ähnlich. Zunächst wird eine Verbindung zu dem WMI-Service des betreffenden Rechners aufgebaut (theoretisch könnte man die Daten auch remote abfragen). Anschließend schickt man einen Query an den Service, der von der Syntax stark an SQL erinnert. Das Ergebnis bekommt man in einem Objekt-Array zurückgeliefert.

Listing 5.10: WMI Skript: CPU-Load

```
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set colItems = objWMIService.ExecQuery("SELECT LoadPercentage FROM
    Win32_processor",,48)
i=0:cpu=0
For Each objItem in colItems
  cpu = cpu + objItem.LoadPercentage
  i=i+1
Next
wscript.echo round(cpu/i,2)
```

Listing 5.11: WMI Skript: NET-Load

```
on error resume next
i=0:max=0:sum=0
dim bytes1()
dim bw1()
```

```
dim bytes2()
dim bw2()
Redim bytes1(0)
Redim bw1(0)
Redim bytes2(0)
Redim bw2(0)

Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")

Set colItems = objWMIService.ExecQuery("SELECT * FROM
    Win32_PerfRawData_Tcpip_NetworkInterface",,32)
For Each objItem in colItems
    if InStr(objItem.Name,"Loopback")<>0 then
      if Ubound(bytes1) < i then redim preserve bytes1(i)
      if Ubound(bw1) < i then redim preserve bw1(i)
      timestamp1 = objItem.TimeStamp_PerfTime
      bytes1(i) = objItem.BytesTotalPersec
      bw1(i) = objItem.CurrentBandwidth
      i=i+1
      timebase = objItem.Frequency_PerfTime
    end if
Next

WScript.Sleep CInt(1000)

i=0
Set colItems = objWMIService.ExecQuery("SELECT * FROM
    Win32_PerfRawData_Tcpip_NetworkInterface",,32)
For Each objItem in colItems
    if InStr(objItem.Name,"Loopback")<>0 then
      if Ubound(bytes2) < i then redim preserve bytes2(i)
      if Ubound(bw2) < i then redim preserve bw2(i)
      timestamp2 = objItem.TimeStamp_PerfTime
      bytes2(i) = objItem.BytesTotalPersec
      bw2(i) = objItem.CurrentBandwidth
      i=i+1
    end if
Next


for j=0 to i-1
    proz = ((bytes2(j)-bytes1(j))/ ((timestamp2-timestamp1) / timebase)*8192)/bw2
        (j)
    sum = sum + proz
    if max<proz then max=proz
next

wscript.echo round(sum/i,2)
```

Listing 5.12: WMI Skript: MEM-Load

```
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_ComputerSystem",,48)
For Each objItem in colItems
    totalM = objItem.TotalPhysicalMemory/1024
Next
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem",,48)
For Each objItem in colItems
    freeM = objItem.FreePhysicalMemory
Next
```

```
wscript.echo round(100-(freeM/totalM*100),2)*100
```

Listing 5.13: WMI Skript: Temperatur

```
On Error Resume Next

Set objWMIService = GetObject("winmgmts:\\.\root\WMI")
Set colItems = objWMIService.ExecQuery("SELECT * FROM
    MSAcpi_ThermalZoneTemperature",,48)
i=0
max=0
For Each objItem in colItems
    i=i+1
    sum = (objItem.CurrentTemperature - 2732) / 10
    if max<sum then max=sum
Next

average = sum/i

wscript.echo max
```

Listing 5.14: WMI Skript: Prozess-Suche

```
ret=0
Set objArgs = WScript.Arguments
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Process WHERE Caption
    ='" & objArgs(0) & "'",,48)
For Each objItem in colItems
    ret=1
Next
wscript.echo ret
```

Listing 5.15: WMI Skript: Benutzer

```
on error resume next
'Pfad finden
scriptfile = WScript.ScriptFullName
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFile(scriptfile)
scriptpath = f.ParentFolder
Set f = Nothing
x = mid(scriptpath,1,instr(scriptpath,"helper")-1)


rdp=0:vnc=0

' RDP Verbindungen zählen
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set colItems = objWMIService.ExecQuery( "SELECT * FROM
    Win32_PerfRawData_TermService_TerminalServices",,48)
For Each objItem in colItems
    rdp = rdp + objItem.ActiveSessions
Next

' VNC Verbindungen zählen
set wshshell=createobject("wscript.shell")
wshshell.run "cmd /c " & x & "tcpvcon -a -n winvnc.exe > " & x & "data\tmp_vnc",0
Set objFileSystem = CreateObject("Scripting.fileSystemObject")
Set objInputFile = objFileSystem.OpenTextFile(x & "data\tmp_vnc", 1)
```

```
Do Until objInputFile.AtEndOfStream

inputData = Split(objInputFile.ReadAll, vbNewline)
For each strData In inputData
    if InStr(strData,"ESTABLISHED") then vnc=vnc+1
Next

Loop

'Ergebnis raus
wscript.echo rdp+vnc
```

Bei der Abfrage des Temperatursensors kommt es auf die entsprechenden Mainboard-Treiber an. Fehlt ein entsprechender Treiber, so gibt dieses Skript wegen fehlender Daten „0“ zurück. Eine Besonderheit ist das Benutzer-Skript. Das Skript zählt alle via RDP bzw. lokal angemeldeten Benutzer und zusätzlich alle per VNC zugeschalteten Benutzer. Hierzu wird ein externes Tool benötigt, dass im folgenden Abschnitt behandelt wird.

Zusatztools

BGInfo Windows-Benutzer sollen über den Desktop über die Aktivitäten des Powermans informiert werden. Dazu zählt, dass auf dem Desktop die aktuelle Planung sowie die gesetzte Systemnachricht für den Client angezeigt werden soll. Ein ehemaliges Sysinternals-Tool (jetzt Microsoft-Tool) übernimmt diese Aufgabe. Das Programm BGInfo kann auf Befehl den Hintergrundbildschirm des Desktops nach Vorgaben austauschen lassen. In unserem Fall stammen die Informationen aus einer externen Datei, die der SNMP-Agent schreibt. Somit erhält der Benutzer alle Informationen auf dem Desktop.

TCPVCON Bei dem Tool „TCPView“ handelt es sich ebenfalls um ein ehemaliges Sysinternals-Tool, das mittlerweile von Microsoft übernommen wurde. Es handelt sich im Prinzip um ein stark verbessertes „netstat“. Das Tool zeigt alle offenen Verbindungen in Zusammenhang mit den entsprechenden Programmen an. Somit können wir mittels VB-Skript gezielt nach einem VNC-Prozess suchen um herauszufinden ob eine Verbindung mit einem Client besteht.

SRVany + eigener Dienst Der Powerman-Client wird zwar als Windows-Dienst eingetragen und bei jedem Systemstart gestartet, jedoch kann er nicht selbst den Anmelde-Trap senden. Eine diesbezügliche Modifikation des Sourcecodes war nicht möglich. Da das Anmelden beim Hochfahren passieren soll und nicht erst wenn sich ein Benutzer einloggt, kann man nicht auf die Windows Bordmittel zurückgreifen, sondern braucht die Hilfe von srvany (Microsoft Tool). Mit SRVany ist es möglich, eine normale ausführbare Datei als Windows Service starten zu lassen. Mit SRVany erstellen wir einen zweiten „Dienst“, der den Trap losschickt, sobald der SNMP-Agent gestartet ist.

Listing 5.16: WMI Skript: Anmelde-Trap

```
'Pfad finden
scriptfile = WScript.ScriptFullName
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFile(scriptfile)
scriptpath = f.ParentFolder
Set f = Nothing
z = mid(scriptpath,1,instr(scriptpath,"helper")-1)

x=0
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Do
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Service WHERE Caption
    = 'Powerman SNMP Agent'",,48)
For Each objItem in colItems
```

```
    If objItem.Started=false Then
        WScript.Sleep CInt(1000)
    Else
        x=1
    End If
Next
Loop Until x=1
Set wshshell=createobject("wscript.shell")
wshshell.run z & "\snmptrap tcp:192.168.43.106:162 ''
    1.3.6.1.4.1.19518.4123.1.1.0",o
```

Erstellung eines Installationsskriptes

Die Installation auf den Clients muss auf jeden Fall automatisiert möglich sein. Der erste Ansatz war die Erstellung eines MSI-Paketes. Leider müssen für den Powerman Client zur Installationszeit einige Aktionen durchführt werden bzw. Parameter im System eingetragen werden. Das hätte von uns eine sehr tiefe Einarbeitung in die MSI-Technik abverlangt. Deswegen haben wir uns schließlich dagegen entschieden und einen Installer mit VBS geschrieben. Das Skript verlangt vom Benutzer die Zieleingabe, wohin die Clientsoftware installiert werden soll und kümmert sich dann um die Installation.

Listing 5.17: VBS Skript: Installation des Powerman

```
'Init
Set fso = CreateObject("Scripting.FileSystemObject")
Set objWMIService = GetObject("winmgmts:\\.\root\CIMV2")
Set WshShell=CreateObject("WScript.Shell")

'Variablen
pfad = "c:\powerman"
scriptpath =""
scriptfile =""

'Herkunft
scriptfile = WScript.ScriptFullName
Set f = fso.GetFile(scriptfile)
scriptpath = f.ParentFolder
Set f = Nothing

'Meldungen
M01 = "Setup abgebrochen"
M02 = "Bitte Installationspfad eingeben (ohne Backslash am Ende)"
M03 = "Powerman Setup"
M04 = "Installation abgeschlossen"

' Installverzeichnis herausfinden
Set objArgs = WScript.Arguments
If WScript.Arguments.Count>0 Then
        pfad = objArgs(0)
Else
        result = InputBox(M02,M03,pfad, 100, 100)
  If result="" Then
                WScript.Echo M01
        WScript.Quit()
  End If
  pfad = result
End If

' Ordner und Dateien kopieren kopieren
fso.CopyFolder "programm", pfad
```

```
fso.CopyFile  scriptpath & "\install\shutdown.exe",fso.getspecialfolder(1) & "\
    shutdown.exe", true

' Registry-Einträge hinzufügen
WshShell.RegWrite "HKEY_USERS\.DEFAULT\Software\Sysinternals\BGInfo\EulaAccepted"
    , "00000001", "REG_DWORD"
WshShell.RegWrite "HKLM\Software\Net-SNMP\SNMPCONFPATH",replace(pfad,"\","/"),"
    REG_SZ"
WshShell.RegWrite "HKLM\Software\Net-SNMP\SNMPSHAREPATH",replace(pfad,"\","/") &
    "/share/snmp","REG_SZ"
WshShell.RegWrite "HKLM\Software\Net-SNMP\InstallDir",replace(pfad,"\","\\"),"
    REG_SZ"
WshShell.RegWrite "HKLM\Software\Net-SNMP\Version","5.4.0mm","REG_SZ"
WshShell.RegWrite "HKCU\Software\ORL\WinVNC3\RemoveWallpaper","00000000","
    REG_DWORD"
WshShell.RegWrite "HKLM\Software\ORL\WinVNC3\RemoveWallpaper","00000000","
    REG_DWORD"
WshShell.RegWrite "HKLM\Software\Microsoft\Windows\CurrentVersion\Run\PowermanUSM
    ","cscript //B " & replace(pfad,"\","\\") & "\\helper\\update_motd.vbs"
WshShell.RegWrite "HKLM\Software\Net-SNMP\PowermanDir",replace(pfad,"\","\\"),"
    REG_SZ"

' PowermanAgent als Dienst anmelden
WshShell.Run pfad & "\snmpd.exe -register -quiet  -Lf " & pfad & "\snmpd.log"

' PowermanHelper installieren
WshShell.Run pfad & "\helper\instsrv.exe PowermanHelper " & pfad & "\srvany.exe",
    0
WshShell.RegWrite "HKLM\SYSTEM\CurrentControlSet\Services\PowermanHelper\
    Parameters\Application", "cscript", "REG_SZ"
WshShell.RegWrite "HKLM\SYSTEM\CurrentControlSet\Services\PowermanHelper\
    Parameters\AppDirectory", ".", "REG_SZ"
WshShell.RegWrite "HKLM\SYSTEM\CurrentControlSet\Services\PowermanHelper\
    Parameters\AppParameters", "//B " & replace(pfad,"\","\\") & "\\helper\\logon
    -to-powerman.vbs", "REG_SZ"

'Kurze Pause
WScript.Sleep(4000)

' Alles starten
Set objOutParams = objWMIService.ExecMethod("Win32_Service.Name='Powerman SNMP
    Agent'", "StartService")
Set objOutParams = objWMIService.ExecMethod("Win32_Service.Name='PowermanHelper'"
    , "StartService")

WScript.echo M04
```

5.2 Powerman CCU

Die CCU ist u.a. die Schnittstelle zwischen unserem System und der zu Grunde liegenden Soft- und Hardware. Eine Anweisung des Powerman-Systems an Managementkomponenten oder Clients wird von der CCU in echte Steuersignale übertragen (vgl. Kapitel 4.7.3). Die weiteren Funktion der CCU sind das automatische Auswerten und Ausführen der Regeln und Heuristiken das Sammeln von Informationen und das Registrieren von Statusänderungen.

5.2.1 Steuerbibliothek

Nach aller Logik und Abstraktion sind die vier Grundfunktionen, die der Powerman bietet das An- und Abschalten, Neustarten und das Powercyclen (Rechner kurzzeitig vom Stromnetz nehmen). Je nach Betriebssystem und Architektur müssen diese Funktionen unterschiedlich implementiert werden, bzw. können die Funktionen überhaupt implementiert werden. Die Aufgabe der Schnittstelle ist es diese Funktionen zur Verfügung zu stellen, unabhängig von Architektur, Betriebssystem oder Managementkomponente.

Unser erster Ansatz ist das Benutzen von Wake On Lan (WOL) zum Anschalten der Computer. Dabei handelt es sich nicht um ein echtes Anschalten des Rechners, sondern lediglich um das „Aufwecken“ aus dem Soft-Off Zustand (vgl. Kapitel 3.1.1). Die Erfahrung zeigt jedoch, dass die WOL-Implementierungen der Hardwarehersteller nicht immer funktionieren. In Sun-Servern scheint es gar nicht implementiert zu sein. Aus diesem Umstand folgt direkt die Forderung nach Managementkomponenten, die das Anschalten für uns übernehmen.

Schließt man einen PC an das Stromnetz an, so wechselt dieser im Allgemeinen sofort in den Soft-Off-Zustand. Die meisten PCs bieten jedoch die Möglichkeit das Verhalten beim Anschließen an das Stromnetz im BIOS einzustellen. Konfiguriert man die PCs so, dass sie anstatt in den Soft-Off-Zustand zu wechseln direkt starten, so ist man nicht mehr auf Techniken wie WOL angewiesen, allerdings entsteht ein neues Problem, denn man muss nun die Computer vom Stromnetz nehmen und zum Einschalten wieder an das Stromnetz anschließen. Dafür verwenden wir in unserem Projekt die per USB steuerbare Gembird Steckerleiste. Nun können wir PCs einschalten, indem wir die Steckerleiste dazu veranlassen die entsprechende Steckdose mit Strom zu versorgen. Sun-Server sind mit einem ähnlichen Feature ausgestattet, der Sun-ALOM-Einheit (kurz: LOM). Die LOM funktioniert unabhängig vom Server und sie kann diesen ein- und ausschalten. Sun-ALOM steht für Sun Advanced Lights Out Management. In unserem Projekt sind die seriellen Anschlüsse der LOMs alle auf unterschiedlichen TCP-Ports der IP-Adresse zu erreichen. Während also die LOMs per TCP/IP gesteuert werden, wird die Gembird Steckerleiste per USB gesteuert.

Einen Computer ausschalten, bzw. in den Soft-Off-Zustand zu versetzen macht im Allgemeinen das Betriebssystem. Unser Server muss also lediglich die Powerman-Clientsoftware (siehe Kapitel 5.1) dazu veranlassen das Betriebssystem herunterzufahren. Dabei kann es ebenfalls zu Problemen kommen. Bei ungünstiger Hardward- und/oder Softwarekonfiguration ist es dem System unmöglich den Computer in den Soft-Off-Zustand zu versetzen. Ebenso kann es passieren, dass ein beschädigtes Betriebssystem nicht vollständig herunterfahren kann. Auch diese Probleme können mit der Hilfe von Managementkomponenten gelöst werden. In unserem Fall kann die Steckerleiste PCs einfach vom Stromnetz nehmen und die LOMs stellen eine Funktion „poweroff“ zur Verfügung, welche dem Server ebenfalls die Stromzufuhr kappt.

Beim Powercyclen wird ein Rechner kurzzeitig vom Stromnetz genommen. Mit diesem Werkzeug kann man eingefrorene (Betriebs-)Systeme unabhängig von ihrem Zustand zu einem Neustart bewegen. Diese Möglichkeit steht bei jedem Sun-Server zur Verfügung, da die LOM immer die Möglichkeit des Aus- und wieder Einschaltens hat. PCs bieten jedoch keine analoge Funktionalität und so steht die Möglichkeit des Powercyclens nur für diejenigen PCs zur Verfügung, die an eine Managementkomponente (konkret: Gembird Steckerleiste) angeschlossen sind.

Das Neustarten ist die einzige Funktion, die zuverlässig bei jedem Rechner als Betriebssystemfunktion, welche von der Clientsoftware ausgelöst wird, zur Verfügung steht. Im Verlauf des Projektes hat sich jedoch herausgestellt, dass wir diese Funktion praktisch nur an einer Stelle (Prozessüberwachung) benötigen.

Ziel ist es, dass wir die Clients nur durch unsere Schnittstelle steuern. Es gibt noch zwei weitere Funktionen, die nicht nur in der Clientsoftware implementiert werden. Das sind die Funktionen zum Setzen der Systemnachricht und zum Abbrechen eines Herunterfahrvorgangs. Beide Funktionen benötigen Zugriff auf die Datenbank und müssen daher Aktionen auf dem Server und dem Client durchführen. Da die Clients passiv sind, müssen diese Funktionen vom Server ausgelöst

Das Ergebnis ist die im UML-Diagramm in Abbildung 5.2 modelierte Bibliothek. Die öffentlichen Methoden der Klasse CInterface übernehmen als Argument die eindeutige mid des betroffenen Rechners. Individuelle Besonderheiten wie Managementkomponente und IP werden innerhalb der Methode ermittelt und die entsprechenden Maßnahmen durchgeführt. Mit dieser Maskierung erfolgt die Benutzung einer Managementkomponente implizit, gesteuert durch die gespeicherten Einstellungen in der Datenbank. Dies ist bereits ein erster Schritt in Richtung Automatisierung, denn unser Automat muss später nur noch entscheiden ob und welche

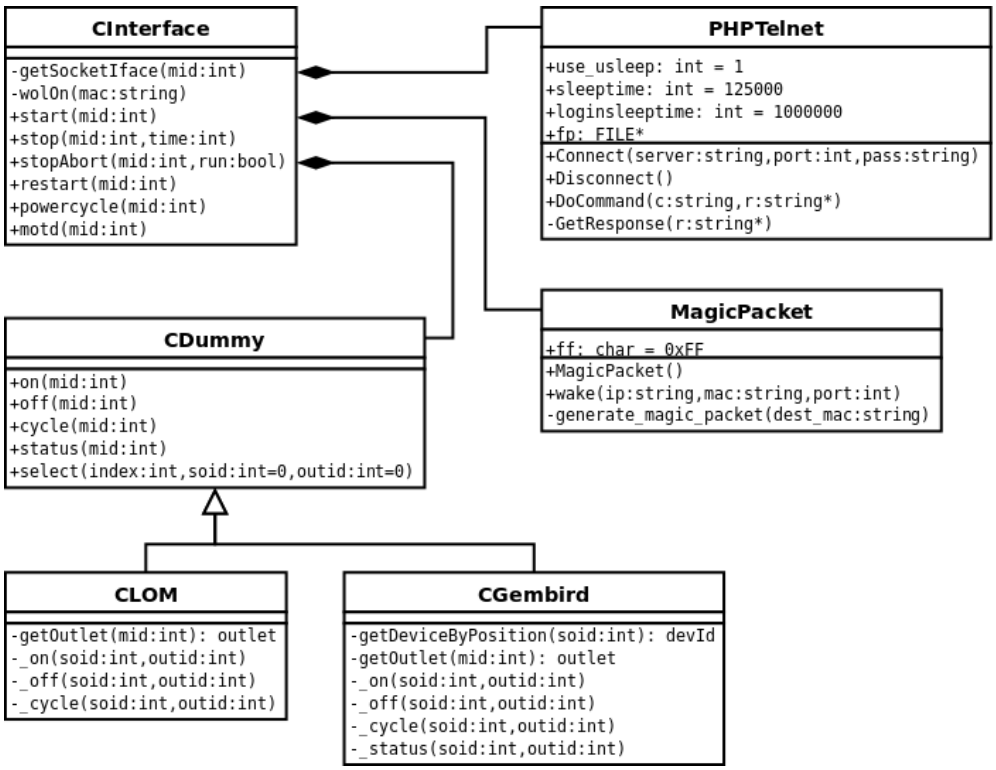


Abbildung 5.2: Bibliothek der CCU für Steuerfunktionen

Aktion (Einschalten, Ausschalten, Neustarten; siehe Grundlegender Aufbau, Kapitel 4.2) durchgeführt werden soll. Die Art und Weise ist transparent durch unsere Bibliothek.

5.2.2 Powerman-Logik

Die Powerman-Logik ist der Automat zur Auswertung und Durchsetzung von Grundbetriebszeiten, Heuristiken und Ausnahmen. Alle Entscheidungen über das Ein- und Ausschalten der Clients werden hier getroffen. Dazu müssen Ausnahmen, Heuristiken zusammen mit gesammelten Informationen und Grundbetriebszeiten ausgewertet werden. Außerdem muss die Hierarchie zwischen diesen drei Kontrollschichten eingehalten werden.

Listing 5.18: Dieser SQL-Query entscheidet über alle automatischen Zustandsänderungen der Clients

```
SELECT * FROM (
        SELECT
                mid,'0' as originator,CASE
                        WHEN r.begin::time=%2::time AND tid IS NULL then 1
                        WHEN r.begin::time=%2::time OR start=%2::time then 3
                        WHEN r.end::time=%2::time then 5
                        ELSE 0
                END as action, polid as attribute
        FROM
                (
                        rule as rj
                        LEFT JOIN policy AS pj(polid,name,exception) ON rj.pid=pj
                            .polid
                ) AS r
                LEFT JOIN time as t ON t.pid=r.pid
        WHERE
```

```
        1=%30 AND r.exception=true AND r.mid=%1 AND (%2 BETWEEN r.begin
            AND r.end) AND (t.dow=extract(dow FROM TIMESTAMP %2)
        OR t.dow IS NULL
) UNION (
SELECT
        mid,'1' as originator, CASE
                WHEN action=0 THEN 1
                WHEN action=1 THEN 2
                WHEN action=2 THEN 4
        END as action,hid as attribute
FROM
        heuristic
WHERE
        1=%31 AND mid=%1 AND memory>=delay
) UNION (
SELECT
        mid,'2' as originator,CASE
                WHEN stop::time=%2::time then 5
                ELSE 3
        END as action,polid as attribute
FROM
        (
                rule as rj
                LEFT JOIN policy AS pj(polid,name,exception) ON rj.pid=pj
                    .polid
        ) AS r
        LEFT JOIN time as t ON t.pid=r.pid
WHERE
        1=%32 AND r.exception=false AND r.mid=%1 AND (%2 BETWEEN r.begin
            AND r.end) AND (t.start=%2::time OR t.stop=%2::time) AND t.
            dow=extract(dow FROM TIMESTAMP %2)
) UNION (
SELECT
        mid,'1.5' as originator, CASE
                WHEN SUM(detector)=0 THEN 3
                ELSE 5
        END as action, 0 as attribute
FROM
        (
                SELECT
                        *, CASE
                                WHEN r.end=%2 OR stop=%2::time THEN -1
                                WHEN r.begin=%2 OR start=%2::time OR %2::
                                    time BETWEEN start AND stop THEN 1
                        END as detector
                FROM
                        (
                                rule as rj
                                LEFT JOIN policy AS pj(polid,name,
                                    exception) ON rj.pid=pj.polid
                        ) AS r
                        LEFT JOIN time as t ON t.pid=r.pid
                WHERE
                        1=%35 AND r.mid=%1 AND (%2 BETWEEN r.begin AND r.
                            end) AND (t.dow=extract(dow FROM TIMESTAMP
                            %2) OR t.dow IS NULL) AND ((%2=r.end) OR (t.
                            dow=extract(dow FROM TIMESTAMP %2) AND %2
                            BETWEEN start AND stop))
        ) as abc
GROUP BY mid
) UNION (
```

```
        SELECT
                mid,'0' as originator, CASE
                        WHEN r.begin::time=(%2::timestamp + interval '%41 minute'
                            )::time AND tid IS NULL THEN 1
                        WHEN r.begin::time=(%2::timestamp + interval '%41 minute'
                            )::time OR start=(%2::timestamp + interval '%41
                            minute')::time THEN 5
                        WHEN r.end::time=(%2::timestamp + interval '%41 minute')
                            ::time THEN 5
                        ELSE 5
                END as action, polid as attribute
        FROM
                (
                        rule as rj
                        LEFT JOIN policy AS pj(polid,name,exception) ON rj.pid=pj
                            .polid
                ) AS r
                LEFT JOIN time as t ON t.pid=r.pid
        WHERE
                1=%30 AND r.exception=true AND r.mid=%1 AND ((%2::timestamp +
                    interval '%41 minute')::timestamp BETWEEN r.begin AND r.end)
                    AND (t.dow=extract(dow FROM TIMESTAMP %2) OR t.dow IS NULL)
        ) UNION (
        SELECT
                mid,'2' as originator, CASE
                        WHEN stop::time=(%2::timestamp + interval '%41 minute')::
                            time THEN 1
                        ELSE 5
                END as action, polid as attribute
        FROM
                (
                        rule as rj
                        LEFT JOIN policy AS pj(polid,name,exception) ON rj.pid=pj
                            .polid
                ) AS r
                LEFT JOIN time as t ON t.pid=r.pid
        WHERE
                1=%32 AND r.exception=false AND r.mid=%1 AND ((%2::timestamp +
                    interval '%41 minute') BETWEEN r.begin AND r.end) AND (t.
                    start=(%2::timestamp + interval '%41 minute')::time OR t.stop
                    =(%2::timestamp + interval '%41 minute')::time) AND t.dow=
                    extract(dow FROM TIMESTAMP %2)
        )
) as xy WHERE NOT action=5 ORDER BY originator ASC,action DESC
```

Der Query aus Listing 5.18 liefert die auszuführende Aktion für einen bestimmten Rechner zu einem bestimmten Zeitpunkt, unter Berücksichtigung aller Grundbetriebszeiten, Heuristiken und Ausnahmen. Dazu enthält der Query noch einige Platzhalter, die zuerst ersetzt werden müssen. Tabelle 5.2 enthält eine Liste aller Platzhalter des Querys. Abgesehen von der mid und dem Timestamp werden alle Platzhalter durch Systemeinstellungen ersetzt. Eine Erläuterung der Systemeinstellungen befindet sich im Anhang B.5.

Die Bibliothek aus dem vorherigen Kapitel ermöglicht die unkomplizierte Steuerung der Clients. Listing 5.18 verarbeitet alle Grundbetriebszeiten, Heuristiken und Ausnahmen zu einer einzigen Anweisung. Mit diesen zwei Werkzeugen ausgestattet fehlt unserem Automat nur noch die aktuelle Uhrzeit und eine Liste aller dem Powerman bekannten Clients. Da jeder Computer einen Zeitgeber in irgendeiner Form besitzt, ist die Uhrzeit direkt vom System beziehbar. Im Kapitel 4.7.1 beschäftigen wir uns mit dem Layout der Datenbank. Zentrum dieses Layouts ist die Tabelle machine, welche alle dem Powerman bekannten Clients enthält. Mittels eines kleinen Querys lässt sich schnell eine Liste aller Clients beziehen. Der Automat iteriert vollständig über die Liste der Clients und findet dabei mit Hilfe des oben gezeigten Querys für jeden Client die jetzt anzuwendende

Platzhalter	Ersetzt durch
%1	mid des Rechners
%2	aktueller Timestamp
%30	Ausnahmen deaktivieren/aktivieren
%31	Heuristiken deaktivieren/aktivieren
%32	Grundbetriebszeiten deaktivieren/aktivieren
%35	Softenforcement deaktivieren/aktivieren
%41	Vorlaufzeit in Minuten für Herunterfahren

Tabelle 5.2: Platzhalter des Querys

Aktion. Eine Aktion entspricht einer Funktion der Bibliothek, die der Automat aufruft um unser abstraktes Konzept zur Steuerung der Clients in die Tat umzusetzen.

Die Iteration über die Clientliste muss regelmäßig erfolgen, da wir in Abhängigkeit der Zeit arbeiten. Wir haben uns für ein fünfminütiges Intervall enschieden. Diese Größe scheint uns sinnvoll, weil für unsere Zwecke die Zeitmessung nicht exakt sein muss. Mit 5 Minuten gehen wir einen Mittelweg zwischen Auslastung des Servers und der zeitlichen Auflösung bei Betriebszeiten und Heuristiken. Zu beachten ist auch, dass eine Iteration über die Clientliste relativ zeitintensiv ist, aufgrund der TCP-Verbindung, die zu jedem Rechner auf und abgebaut werden muss. Dazu kommt noch die Zeit, die der Client zwischen Anfrage des Servers und Senden der Antwort benötigt. Eine vereinfachte Darstellung des Automaten zeigt folgender Pseudocode:

```
while TRUE do
    elapsedTime = 0;
    foreach client ∈ db.machine do
        action = eval(query(client,now));
        library.run(action,client);
    end
    wait until elapsedTime = 5 minutes;
end
```

5.2.3 SNMP Informationsbeschaffung

Bisher haben wir in diesem Kapitel eine Bibliothek zur Umsetzung unseres Konzepts in Steuersignale entwickelt. Außerdem haben wir einen kompakten Automaten beschrieben, der mit Hilfe der Bibliothek das Steuern von Clients übernimmt, indem er die Datenbank auswertet. Der letzte zu entwickelnde Teil der CCU übernimmt die Aufgabe die Datenbank mit Informationen anzureichern, damit der beschriebene Automatismus funktionieren kann.

Grundbetriebszeiten, Heuristiken und Ausnahmen werden vom Benutzer mit Hilfe des Frontends (Kapitel A) eingetragen. Außerdem können nur Benutzer im Frontend Clients aus dem System löschen. Die CCU übernimmt das Hinzufügen von neuen Clients und aktualisiert die Liste bekannter Clients mit dem Ist-Zustand (fährt herauf, fährt herunter, an, aus) und sammelt die Informationen, die für die Auswertung der Heuristiken benötigt werden.

Das Aktualisieren der Clientliste passiert an verschiedenen Stellen der CCU. Ein Client, der mit installierter

Powerman-Clientsoftware gestartet wird, schickt während des Bootvorgangs einen SNMP-Trap an den Server (vgl. Kapitel 4.4). Der Trap löst ein Skript aus, welches zuerst anhand der MAC-Adresse der Quelle des Traps bestimmt, ob der Client bereits bekannt ist, oder sich zum ersten Mal am Server anmeldet. Ist die MAC-Adresse dem Server bekannt, so wird lediglich der Zustand „an“ des Clients registriert. Ist die MAC-Adresse dem Server bisher unbekannt, so wird ein neuer Eintrag für die Tabelle „machine“ der Datenbank erstellt und dieser hinzugefügt. Die Zustände „fährt herauf“ und „fährt herunter“ werden von der Steuerbibliothek gesetzt, nachdem das entsprechende Steuersignal gesendet wurde.

Um die benötigten Daten zum Auswerten von Heuristiken zu erhalten, erweitern wir unseren Automaten. Vor der Auswertung der Datenbank werden zunächst alle relevanten Informationen abgefragt und in der Datenbank gespeichert. Dabei betrachten wir den Ist-Zustand. Ein Rechner der aus ist (Trap nicht gesendet) muss nicht abgefragt werden, da er nicht antworten kann. Umgekehrt folgern wir aus einem Rechner, der laut Datenbank an sein sollte jedoch nicht antwortet, dass dieser Rechner aus sein muss und aktualisieren die Liste der bekannten Clients.

Der erweiterete Automat als Algorithmus in Pseudocode:

```
while TRUE do
    elapsedTime = 0;
    foreach client ∈ db.machine do
        if client.status != 'aus' and client.status != 'fährt herunter' then
            data = poll(client);
            if data == FALSE then
                client.status = 'aus';
            end
            else
                db.monitor.push(data);
            end
        end
        action = eval(query(client,now));
        library.run(action,client);
    end
    wait until elapsedTime = 5 minutes;
end
```

5.3 Web-Frontend

Die Hauptaufgabe des Powerman-Systems ist es, in einer größeren Umgebung die Kontrolle über das Ein- und Ausschalten von Servern zu zentralisieren und zu automatisieren. Dadurch erreicht man eine bessere Kontrolle über seine Server und deren Stromverbrauch.

Direkte Kontrolle über die Computer steht dem Benutzer im Powerman Frontend zur Verfügung. Im Frontend hat der Benutzer außerdem die Kontrolle über die automatische Steuerung. Das automatisierte Steuern von Computern richtet sich nach drei Kontrollschichten: den Grundbetriebszeiten als unterste Schicht, den Heuristiken und den Ausnahmen, welche die höchste Kontrollschicht darstellen.

5.3.1 Rollen

Es soll nicht jedem Benutzer die volle Funktion des Powerman zur Verfügung stehen. So haben wir zwei Rollen für die Benutzung des Frontends vorgesehen:

- Benutzer
- Administrator

Der normale Benutzer kann Rechner nur direkt steuern (ein-, ausschalten, powercycle, quickbooking). Im fertigen Frontend bedeutet das, dass dem normalen Benutzer nur die Funktionen der Startseite zur Verfügung stehen. Eine kleine Ausnahme ist das Setzen der Systemnachricht. Diese Funktion ist den normalen Benutzern ebenfalls zugänglich. Ansonsten kann der normale Benutzer nur das Logprotokoll und das Monitoring einsehen. Dem Administrator hingegen steht die volle Mächtigkeit des Powerman zur Verfügung.

Diese Aufteilung ist vor allem dazu da, den Mitarbeitern die Möglichkeit zu geben, mit Rechnern zu arbeiten, unabhängig von Grundbetriebszeiten etc., denn der Administrator kann nicht auf jeden Wunsch im Detail eingehen.

Aktion	**Administrator**	**Benutzer**
Client ein/ausschalten	✓	✓
Client powercyclen	✓	✓
Client Systemnachricht zuweisen	✓	✓
Client Grundbetriebszeit zuweisen	✓	
Client Heuristik zuweisen	✓	
Client Ausnahme zuweisen	✓	
Client Management-Komponente einstellen	✓	
Client Charakteristik zuweisen	✓	
Client entfernen	✓	
Quickbooking	✓	✓
Logprotokoll einsehen	✓	✓
Lokprotokoll löschen (archivieren)	✓	✓
Monitoring-Graphen einsehen	✓	✓
Grundbetriebszeiten verwalten (CRUD*)	✓	
Benutzer verwalten (CRUD*)	✓	
Steckdosen verwalten (CRUD*)	✓	
Systemeinstellungen vornehmen	✓	
Client-Software herunterladen	✓	

* CRUD steht für Create, Read, Update, Delete

Tabelle 5.3: Berechtigung von Administratoren und Benutzer im Überblick

5.3.2 Übersicht Startseite

Nach dem erfolgreichen Anmelden kommt man auf die Übersichtsseite. Hier bekommt der Benutzer eine kurze Übersicht über die dem Powerman-Server bekannten Rechner, zusammen mit ein paar Informationen und Funktionen, gegliedert in vier Blöcke: Systemnachrichten Clients, Quickbooking und Statisik. Einen detailierten Überblick befindet sich im Anhang, unter Kapitel A.1.

Für den Benutzer ist es sehr wichtig die einzelnen Rechner voneinander unterscheiden zu können. Intern benutzt der Powerman die MAC-Adresse, um Clients unterscheiden zu können. Dies ist jedoch für Benutzer nicht sehr komfortabel. Im Frontend greifen wir daher auf den Hostname zur Bennenung von Clients zurück. Intern wird weiterhin die MAC-Adresse benutzt, da ein Hostname nicht notwendiger Weise eindeutig ist. Innerhalb eines Netzwerks sollte der Hostname einen Rechner eindeutig identifizieren. Ist dem nicht so, so ergibt sich daraus lediglich ein Problem bei der Verwaltung im Frontend, der Powerman-Server ist weiterhin in der Lage die Rechner zu unterscheiden. Der Wert wird per SNMP vom Client geholt, jedoch nicht von unserer SNMP-Erweiterung, sondern aus der MIB2 unter dem Pfad 1.3.6.1.2.1.1.5.0. Dies ist lediglich der Hostname, den das System bereit stellt und deckt sich nicht notwendigerweise mit dem DNS-Eintrag, da dies zwei völlig unterschiedliche Dinge sind. Es liegt voll und ganz im Ermessen des Administrators, in wieweit Hostname und DNS-Eintrag zusammenhängen.

Auf der Startseite hat der Benutzer die Möglichkeit Rechner zu starten, bzw. herunter zu fahren, ein Quickbooking zu erstellen bzw. zu löschen oder einen Powercycle durchzuführen. Die Funktion zum Einschalten von

Computern wird sofort ausgeführt. Das System schickt an den betreffenden Rechner auf jeden Fall ein Wake-On-Lan-Paket (WOL), um den Rechner aufzuwecken. Zusätzlich dazu besteht die Möglichkeit dem Rechner eine LOM oder eine schaltbare Steckdose zuzuweisen. Verfügt das System über solch externe Werkzeuge zum Steuern eines Rechners, so werden diese implizit vom System mitbenutzt.

Im Gegensatz zum Einschalten erfolgt das Ausschalten verzögert. Es kann durchaus sein, dass gerade jemand an einem Computer arbeitet, wenn dieser vom Powerman Server den Befehl zum Herunterfahren bekommt. Um zu vermeiden, dass dem Benutzer ein Rechner während der Arbeit abgeschaltet wird, erfolgt das Herunterfahren zeitverzögert. Sobald ein Computer heruntergefahren werden soll, wird auf dem betroffenen Rechner eine Warnung mit der verbleibenden Zeit bis zum effektiven Herunterfahren angezeigt. Wie lange die Verzögerung ist hängt von den Systemeinstellungen 5.3.9 shutdownDelay und shutdownDelayExtra ab. Der effektive Herunterfahrzeitpunkt ist der Zeitpunkt zu dem der Benutzer auf „abschalten“ klickt + (shutdownDelay + shutdownDelayExtra) in Minuten, was normalerweise 18 Minuten nach dem klick ist. Weitere socketoutletOffDelay (normalerweise 6) Minuten später schaltet auch die erweiterte Steuerung, d.h. dass der Computer 6 Minuten Zeit hat herunter zu fahren, danach schaltet sich die Steckdose aus, bzw. wird an die LOM das Kommando „poweroff“ geschickt.

Fährt ein Rechner gerade hoch oder runter, so sind die Funktionen „anschalten“ und „ausschalten“ nicht verfügbar. An ihre stelle tritt ein graues Symbol, ohne Funktion.

Reagiert ein Computer nicht mehr, so bleibt einem meist nichts anderes übrig als ihn aus und wieder ein zu schalten. Diese beiden Schritte vereinen wir in der Funktion „powercycle“. Echte Funktionalität bietet diese Funktion nur für Computer mit externer Steuermöglichkeit und schaltet diese direkt aus, ohne dem Computer den Befehl zum Herunterfahren zu schicken, wartet dann einen kurzen Moment und schaltet den Rechner daraufhin wieder ein. Verfügt der Powerman Server über keine externe Steuermöglichkeit für einen Rechner, so hat diese Funktion keinen Effekt auf den Rechner.

Möchte der Benutzer einen Computer kurzfristig nutzen, oder ihn vom Herunterfahren abhalten, so macht er dies mit der Funktion Quickbooking. Die Aktion in der Liste der Clients startet einen Computer für eine Stunde. Mit dem Quickbooking wird der Computer sofort gestartet, oder ein bevorstehendes Herunterfahren abgebrochen. Allerdings wird der Recher normalerweise nach dem Ablauf dieser Stunde nicht heruntergefahren. Dies geschieht nur wenn die Systemkonfiguratiosvariable (5.3.9) softEnforcement wahr ist. Vor dieser Einstellung möchten wir auch an dieser Stelle ausdrücklich warnen, da sie unerwünschte Nebeneffekte haben kann.

Ein vorhandenes Quickbooking kann vom Benutzer wieder frei gegeben werden, indem er es mit der Funktion „Quickbooking aufheben“ wieder löscht. Mit der Aufhebung eines Quickbookings übernehmen wieder Grundbetriebszeiten A.3 und Heuristiken A die Kontrolle über den Computer.

5.3.3 Clients

Unter dem Menüpunkt „Clients“ findet man eine detaillierte Liste aller dem Powerman-Server bekannten Rechner. Diese Liste zeigt gibt Auskunft über den Status, den Hostname, das Betriebssystem, Mac- und IP-Adresse, sowie die Steuermöglichkeiten, die dem Powerman System für den jeweiligen Rechner zur Verfügung stehen. Diese Seite ist im Anhang auf Seite 65 dargestellt.

Information über das Betriebssystem ist für den Powerman Server unwichtig, dem Benutzer soll sie bei der Verwaltung der Clients helfen, indem der Benutzer eine Vielzahl von Betriebssystemen einsetzen kann, sich jedoch nicht merken muss, welches Betriebssystem auf welchem Computer eingesetzt wird.

Die Mac-Adresse ist dagegen wichtig für das Funktionieren des Powerman Systems. Meldet sich ein Client am Powerman Server, so überprüft dieser, ob ihm die Mac bereits bekannt ist. Nur eine dem Server unbekannte Mac wird als neuer Rechner dem System hinzugefügt. Kennt der Server die Mac bereits, so aktualisiert er die Informationen, die er zu der Mac speichert. Das hießt, dass weder Betriebssystem, noch IP-Adresse, noch andere Informationen zum Identifizieren eines Computers benutzt werden. Hierzu dient lediglich die Mac-Adresse. Ist es dem Server unmöglich die Mac-Adresse eines Clients (weder per SNMP noch aus seinem ARP-Cache) zu bestimmen, so wird die IP-Adresse des Clients in hexadezimaler Darstellung als Mac-Adresse gespeichert, da der Server eine MAC-Adresse braucht, um die Clients voneinander unterscheiden zu können.

Das nächste Feld „IP-Adresse“ zeigt die letzte dem Powerman Server zur Mac-Adresse bekannten IP-Adresse. Die Adresse, die in diesem Feld gespeichert ist, wird benutzt um Statusinformationen von dem betreffenden Client zu erhalten. Antwortet diese Adresse nicht, so wird dieser Rechner vom System als aus erkannt und der Status wird dementsprechend geändert.

Die Felder „Management“ und „Steuerung“ zeigen die Möglichkeiten des Powerman Servers einen Client zu steuern. Prinzipiell benutzt unser System SNMP um mit dem Client zu kommunizieren. Deswegen wird ein Client mindestens mit Hilfe von SNMP gesteuert, was dem Benutzer im Feld „Management“ auch angezeigt wird. Darüber hinaus enthält das Powerman System Module um eine bessere Kontrolle über die Clients zu erhalten. Zur zeit existieren genau zwei dieser Module, eines zur Kommunikation mit den Sun LOMs per Telnet und eines zum Steuern der Pearl Gembird Steckdose. Mit diesen Modulen erweitert der Server seine Steuermöglichkeiten um die Möglichkeit einen Rechner mit Strom zu versorgen bzw. einen Rechner vom Stromnetz zu trennen. Dies hat mehrere Vorteile. Zum Einen kann mit Hilfe der Gembird Steckerleiste zusätzlich Strom gespart werden, da der Strombedarf des Computers während der Standby-Zeit wegfällt und zum Anderen bleibt dem Powerman Server immer noch die Möglichkeit einen Rechner neu zu starten, wenn dieser nicht mehr reagiert. Außerdem haben praktische Tests gezeigt, dass sich nicht alle Computer per WOL fern starten lassen. So bieten diese Module eine verlässlichere Methode Rechner zu starten. Bei der Gembird Steckerleiste muss man die entsprechenden Computer im BIOS so konfigurieren, dass sie automatisch starten, wenn sie am Stromnetz hängen. Dies hat zur Folge, dass sie sofort starten, wenn die Steckdose vom Powerman Server auf „an“ geschaltet wird, wodurch man unabhängig von der WOL-Funktion geworden ist. Die Sun LOMs benutzen den Befehl „poweron“, um einen Computer zu starten. Mit Hilfe des entsprechenden Moduls kann sich der Powerman per Telnet auf einer LOM anmelden und das „poweron“ Kommando absetzen. So erreicht man auch hier Unabhängigkeit von der WOL-Funktion. Wird ein Computer mit der Hilfe eines solchen Moduls gesteuert, so zeigt sich dies an zwei Stellen. Im Feld „Management“ wird zusätzlich zu SNMP noch „Steckdose“ oder „LOM“ angezeigt, je nach benutztem Modul. Das Feld „Steuerung“ enthält genauere Details. Hier steht der Name der Steuereinheit und hinter dem ≫:≪ der Anschluss, über den der Computer gesteuert wird. Die von uns verwendete Gembird Steckerleiste besitzt vier schaltbare Steckdosen, somit es nicht ausreichend einem Rechner eine Steckerleiste zuzuweisen, es ist außerdem noch notwendig zu bestimmen, an welcher Steckdose der Computer hängt. Die Steckdosen einer Steckerleiste sind von 1 bis n (in diesem Fall 4) durchnummeriert. Eine Steckerleiste hat einen vom Benutzer definierbaren Namen (siehe Kapitel 5.3.6 für Details). Für einen Computer an der ersten Steckdose der Steckerleiste mit dem Namen „ImSchrankGanzObenLinks“ angeschlossen ist, ergibt sich für das Feld „Steuerung“ also der Eintrag „ImSchrankGanzObenLinks:1“. Für die LOMs benötigt man nicht Steckerleisten und Steckdosennummern, sondern IP-Adressen und Portnummern. IP-Adressen werden hinter vom Benutzer frei definierbaren Namen versteckt, die Portnummern werden so wie sie sind weiterverwendet. Der Eintrag für das Feld „Steuerung“ ergibt sich dann analog zu Steckerleisten.

In der letzten Spalte der Liste werden dem Benutzer Funktionen zur Verfügung gestellt, da diese hier etwas umfangreicher sind, sind einige von ihnen auf Unterseiten zusammen gefasst.

Zeit zuweisen

Die Hauptaufgabe des Powerman Systems ist das Kontrollieren der Betriebszeiten von Clients. Alle dies betreffenden Einstellungen befinden sich unter dem Punkt „Zeit zuweisen“. Diese Seite enthält alle Werkzeuge zur Verwaltung der drei Kontrollschichten (Grundbetriebsteigen, Heuristiken und Ausnahmen,vgl. Kapitel A) für den Client. Damit der Benutzer einen Überblick über die momentanen Einstellungen bekommt, gibt es auf dieser Seite einen Monatskalender, der grob die Betriebszeiten des Clients anzeigt. Jeder Tag ist ein Rechteck, Zeiten werden als Linien von links nach rechts dargestellt, wobei der linke Rand eines Rechtecks 0 Uhr und der rechte Rand 24 Uhr (also nächster Tag 0 Uhr) darstellt. Die unterste (blaue) Linie eines Tages zeigt an, wann ein Computer den Einstellungen entsprechend an sein sollte. Die darüberliegenden Linien zeigen den Gültigkeitsbereich von Grundbetriebszeiten an. Der Gültigkeitsbereich ist nur durch Start- und Enddatum definiert, weswegen die Darstellung eine durchgehende Linie von Start-, bis Enddatum ist. Die Linien sind farblich getrennt, damit dem Benutzer das Verfolgen von Linien, die sich über mehrere Zeilen (Wochen) erstrecken, leichter fällt.

Grundbetriebszeiten An dieser Stelle hat der Benutzer die Möglichkeit, dem Client beliebig viele Grundbetriebszeiten zuzuweisen. Diese werden zentral unter dem Menüpunkt Grundbetriebszeiten (siehe Kapitel A.3) verwaltet. Hier erfolgt nur die Zuweisung und die Festlegung von Start- und Enddatum. Außerdem können bereits erfolgte Zuweisungen bearbeitet und gelöscht werden.

Heuristik Der Link unter dem Punkt „Heuristik“ führt den Benutzer zur Eingabemaske für Heuristiken, wo der Benutzer die Möglichkeit hat die Heuristiken aus A zu konfigurieren.

Ausnahme Analog zu Grundbetriebszeiten werden hier die Ausnahmen für diesen Client konfiguriert.

Einstellungen

Bei der Erläuterung der Übersichtsseite sprachen wir über spezielle Managementkomponenten, die dem Powerman-Server mehr Gewalt über die Computer bieten. Auf der Seite „Einstellungen“ nimmt man die Zuweisung von Managementmodul zu Client vor. Mit Hilfe der Radiobuttons entscheidet der Benutzer, ob und welche spezielle Steuerung für den Client aktiviert werden soll. Außerdem muss das Modul konfiguriert werden.

Für Steckerleisten gibt es eine Liste, aus der der Benutzer eine Steckerleiste und Steckdose auswählen muss. Dabei gilt, dass jede Steckdose systemweit nur einmal verwendet werden darf. Die Elemente der Liste setzen sich zusammen aus dem Namen der Steckerleiste und einer Zahl, welche der aufgedruckten Nummer der Steckdose an der Steckerleiste entspricht.

Das andere zur Verfügung stehende Modul ist für die Sun LOM, welche per Telnet gesteuert wird. Dem Modul muss also IP-Adresse und Port mitgegeben werden. Die IP-Adresse wird wie bereits erwähnt hinter einem Bezeichner versteckt. Der Benutzer muss aus einer Liste, ähnlich wie bei Steckerleisten, die gewünschte IP bzw. den Bezeichner auswählen. Prinzipiell kann eine Telnetverbindung zu jedem TCP-Port aufgemacht werden. Würde man die Auswahl analog zu Steckerleisten mit einer Liste mit allen möglichen Kombinationen aus Bezeichner und Portnummer realisieren, so würde diese Liste aus 65536 Einträgen pro Bezeichner bestehen. Da dies nicht mehr komfortabel zu nutzen ist, wählt der Benutzer bei den LOMs nur den Bezeichner, also die IP-Adresse aus einer Liste aus und der TCP-Port wird im Eingabefeld daneben angegeben.

Die zweite Funktion auf dieser Seite ist das Setzen einer Systemnachricht. Die Idee hinter einer Systemnachricht ist dem Benutzer die Möglichkeit zu geben kurze Nachrichten auf den Clients zu hinterlassen, die als Information für ihn oder andere Benutzer dienen. Unter Windows wird die Systemnachricht permanent als Hintergrundbild angezeigt, während die Systemnachricht unter Solaris bei jedem Anmelden auf der Konsole angezeigt wird.

Löschen

Die letzte dem Benutzer zur Verfügung stehende Funktion ist das Löschen von Clients aus der Liste. Dies ist notwendig, wenn ein Client nie wieder benutzt werden soll, oder wenn er eine neue Netzwerkkarte bekommt und sich somit seine Mac-Adresse ändert. Da Mac-Adressen das Erkennungsmerkmal von Computern im Netzwerk sind, ist das Powerman System davon abhängig. Eine neue Mac-Adresse wird als neuer Client im System eingetragen, ungeachtet des Hostnames oder des darauf installieren Betriebssystems. Nach dem Aufruf der Löschenfunktion muss der Benutzer den Vorgang noch einmal bestätigen. Danach wird der Client und alle zugehörigen Daten aus dem System gelöscht.

5.3.4 Grundbetriebszeiten

Nun folgt der Menüpunkt „Grundbetriebszeiten“. Hier werden Grundbetriebszeiten verwaltet, damit sie Clients zugewiesen werden können. Auf der Eingangseite erhält der Benutzer eine vollständige Liste aller Grundbetriebszeiten und den dazugehörigen Betriebszeiten. Der Name einer Grundbetriebszeit ist frei wählbar und

dient der Übersicht, wenn Grundbetriebszeiten den Rechnern zugewiesen werden. Die einzelnen Zeiten werden nach Tag getrennt für jede Grundbetriebszeit seperat und nach Startzeit aufsteigend sortiert angezeigt. Die Darstellung ist immer <Startzeit>-<Stopzeit>. Zur Startzeit wird ein Computer hochgefahren und zur Stopzeit wieder heruntergefahren, natürlich nur, wenn das mit allen anderen definierten Regeln vereinbar ist. Dazu wird dem Benutzer in der Spalte „zug." angezeigt, wievielen Rechnern eine Grundbetriebszeit zugeordnet ist. Mit den Funktionen aus der „Aktion" Spalte können Grundbetriebszeiten bearbeitet und gelöscht werden.

Erstellen/Bearbeiten

Die Eingabemasken zum Erstellen und Bearbeiten von Grundbetriebszeiten sind gleich, bis auf die Tatsache, dass beim Bearbeiten von Grundbetriebszeiten schon ein Name und Zeiten angegeben sind, während die Felder beim Erstellen einer neuen Grundbetriebszeit leer sind. Zuerst muss der Benutzer einen eindeutigen Namen für die Grundbetriebszeit angeben. Hat er dies getan, so folgt die zweite Seite. Diese ist in zwei Sektionen unterteilt.

In dem oberen Kästchen „Neuen Zeitpunkt festlegen" fügt der Benutzer Zeitspannen, in denen ein Client an sein soll, zur Grundbetriebszeit hinzu. Dazu wählt er einen Wochentag, eine Start- und eine Stoppzeit aus und fügt sie der Liste „Eingetragene Zeiten", dem zweiten Kästchen, hinzu, indem er auf den Diskettenbutton klickt.

Im unteren Kästchen befindet sich die Liste mit allen zur Grundbetriebszeit gehörenden Zeitspannen. Durch den Link neben jedem Eintrag kann der Benutzer die entsprechende Zeitspanne aus der Grundbetriebszeit entfernen.
Ist der Benutzer fertig mit der Definition von Betriebszeiten, so muss er die von ihm getroffenen Einstellungen noch abspeichern, damit die Änderungen durchgeführt werden.

Löschen

Wird eine Grundbetriebszeit nicht mehr benötigt, so kann sie aus dem System entfernt werden, damit die Liste der Grundbetriebszeiten übersichtlich bleibt. Löschfunktionen sind über das ganze Frontend hinweg einheitlich. Jeder Eintrag (in diesem Fall jede Grundbetriebszeit) hat einen eigenen Löschbutton. Wird dieser ausgelöst, so muss der Benutzer das Löschen noch einmal bestätigen, bevor die Änderungen am System vorgenommen werden.

5.3.5 Benutzer

Um das Powerman System konfigurieren zu können, muss man sich zunächst am Frontend anmelden. Dies geschieht durch eine einfache Passwortabfrage. Nach einer erfolgreichen Authentifizierung wird dem Benutzer Zugang zum Frontend gewährt. Dabei orientiert sich das Frontend an der Priorität des Benutzers, welche ihm zur Erstellzeit seines Zugangs zugewiesen wurde. Die Priorität ist wichtig, um eine Art Hierarchie unter den Benutzern zu schaffen. Die Priorität wird durch eine ganze Zahl beschrieben, wobei 1 die höchste Priorität ist und die Priorität eines Benutzer geringer wird, je höher der Zahlenwert. Die Unterschiede in der Priorität machen sich an zwei Stellen bemerkbar. Zum einen dürfen nur Benutzer mit einer Priorität kleiner oder gleich 10 Benutzer verwalten, wozu auch das Ändern der Priorität gehört. Zum anderen können Ausnahmen nur von Benutzern mit gleicher oder höherer Priorität verändert oder gelöscht werden. Indem man jedem Benutzer einen eigenen Zugang einrichtet erreicht man außerdem größere Transparenz, da im Protokoll (siehe 5.3.7 für Details) der Benutzername eingetragen wird.

Hinzufügen/Bearbeiten

Die Eingabemaske zum Hinzufügen und Bearbeiten von Benutzern ist identisch. Beim Hinzufügen sind alle Felder Pflichtfelder, beim Bearbeiten sind die Passwortfelder nur Pflichtfelder, wenn das Passwort geändert werden soll. Der Benutzername muss im System eindeutig sein, da er zur Identifizierung benutzt wird. Die

Email-Adresse bleibt im Frontend unbenutzt. Ihre einzige Aufgabe ist es, dem Benutzer mitzuteilen, wer der Ansprechpartner zu einem Zugang ist.

Löschen

Analog zu allen anderen Löschfunktionen, die aus Übersichten heraus aufrufbar sind muss der Benutzer den Vorgang noch einmal bestätigen, bevor Änderungen am System vorgenommen werden.

5.3.6 Steckdosen

Ursprünglich nur zur Übersicht über alle am Powerman Server angeschlossenen Gembird Steckdosen gedacht, bietet die Seite „Steckdosen“ nun eine Übersicht über alle Komponenten, für Managementmodule. Das Prinzip eines Managementmoduls ist, dass es außer dem Powerman Server und der Software auf dem Client eine dritte Komponente gibt, die das äquivalent zum Ein- und Ausstecken eines gewöhnlichen Arbeitsplatzcomputers realisiert und vom Powerman Server bedienbar ist. In der vorliegenden Version ist das entweder eine Gembird Steckdose, oder die LOM-Einheit eines Sun Servers. Alle Einträge in der Übersicht über alle dem System bekannten Managementkomponenten sind nach Typ und Name sortiert. Bei den Gembird Steckdosen wird im Statusfeld mit roten und grünen Kugeln von links nach rechts entsprechend der Nummerierung der Steckdosen angezeigt welche Steckdosen gerade an sind. Ist eine Managementkomponente erst im System eingetragen, so können die Details nicht mehr verändert werden. Nur der Name kann unter der Aktion „bearbeiten“ verändert werden.

Hinzufügen

Unterhalb der Übersicht befindet sich für jede Art von Managementkomponente ein extra Dialog um Komponenten ins System einzutragen.

Gembird Steckerleisten besitzen kein Merkmal, anhand dessen zwei baugleiche Steckerleisten unterschieden werden können. Deswegen werden sie über ihre Position am USB-Bus identifiziert, was eine komfortable Handhabung schwierig macht. Das System kann nicht erkennen, wenn man eine Steckerleiste umsteckt, oder einen USB-Hub vorschaltet. Um die Benutzung einfach zu halten, bietet der Powerman eine Funktion, die automatisch neue Steckerleisten erkennt und zum System hinzufügt. Zeitgleich werden alle eingetragenen Steckerleisten, die nicht mehr an der erwarteten Stelle am USB-Bus gefunden werden gelöscht.

Sun LOM-Einheiten werden per Telnet gesteuert, also über eine gewöhnliche TCP/IP-Verbindung. Dafür benötigt das System die Angabe einer IP-Adresse und eines TCP-Ports. Diese Angaben sind zweigeteilt. Hier gibt man lediglich die IP-Adresse an. Der TCP-Port wird in 5.3.3 direkt einem Client zugewiesen. Das hat zum Einen den Vorteil, dass man mehrere Sun Server über einen Router mit dessen IP-Adresse steuern kann und somit nicht für jede LOM-Einheit einen extra Eintrag erstellen muss, zum Andern ist dieses Steuerkonzept deckungsgleich mit dem der Gembird Steckdosen, nämlich es gibt ein Steuergerät (Steckerleiste/LOM-Einheit bzw. Router) und es gibt an dem Steuergerät eine Schnittstelle zu dem Client (Steckdose/TCP-Port). So ist die Bedienung im Frontend einheitlich, unabhängig von der Wahl des Mangementmoduls.

Löschen

Analog zu allen anderen Löschfunktionen, die aus Übersichten heraus aufrufbar sind muss der Benotest den Vorgang noch einmal bestätigen, bevor Änderungen am System vorgenommen werden.

5.3.7 Logprotokoll

Das Logprotokoll bietet keine Funktionen, die der Benutzer aktiv aufrufen kann. Dafür kann man hier dem System ein wenig bei der Arbeit über die Schulter schauen. Während die Informationen, die man über die Aktivitäten seiner Mitbenutzer bekommt weniger aussagekräftig sind, sind die Einträge des Benutzers „System“

recht informativ. Der Benutzer System hat keine Zugang zum Frontend. Diese Nachrichten sind vom Powerman Server selbst, der seine Entscheidungen hier mitprotokolliert, damit man seine Aktionen nachvollziehen kann.

5.3.8 Monitoring

Der Powerman Server sammelt kontinuierlich Daten von den Clients. Diese Daten werden intern gespeichert und können hier vom Benutzer eingesehen werden. Dazu werden die gemessenen Daten in Graphen optisch aufbereitet. Um Informationen abrufen zu können, muss der Benutzer zunächst das Gerät auswählen, über das Information angezeigt werden soll. Zur Auswahl stehen alle dem System bekannten Rechner und Temperatursensoren. Als nächstes wählt der Benutzer den Zeitraum, über den er Information abrufen möchte. Zur Auswahl stehen Tag, Monat,Woche und Jahr. Die Angabe des Zeitraums ist immer relativ zu dem Zeitpunkt, an dem der Benutzer die Information einsehen möchte. Darum zeigt „Tag“ immer die Statistik der letzten 24 Stunden an. Analog dazu zeigen „Woche“, „Monat“ und „Jahr“ die Statistik der letzten 168, 720 und 8760 Stunden[1] respektive an.

Allen Graphen ist gemein, dass sie auf der x-Achse die Zeit und auf der y-Achse die gemessenen Werte abgetragen haben. Bei den Graphen CPU-, Mem- und Net-Load ist die Auslastung in Prozent von 0 bis 100 angetragen, die Skala des Graphen der Temperatur dagegen in Grad Celsius von 0 bis 80 und die Skala der Anzahl der am System angemeldeten Benutzer reicht von 0 bis 10.

5.3.9 Systemeinstellungen

Einige Aspekte des Systems sind an einfache Zahlen gebunden. Diese beeinflussen die Performance und Bedienbarkeit des gesamten Systems. Diese einfachen, jedoch grundlegenden Zahlen speichert das Powerman System zentral, aus Gründen der Wartbarkeit und Fehleranalyse. Hier erhält der Benutzer die Möglichkeit, Einfluss auf das Verhalten des Systems auszuüben. Das System kann nur funktionieren, wenn alle Attribute des Systems zusammenpassen. Es gibt noch weitere Attribute, die direkt im Quelltext des Systems verankert sind und deren Wert nicht ohne weitere Modifikationen bedenkenlos verändert werden kann und daher nicht auf dieser Seite veränderbar sind. Eine kurze Erläuterung der Systemeinstellungen befindet sich im Anhang B.5.

5.4 CLI-Frontend

Zusätzlich zum Web-Frontend soll es eine Möglichkeit geben, die wichtigsten Funktionen des Powerman über die Konsole zu steuern, da nicht immer ein Zugriff auf das Web-Interface möglich ist. Bei amasol beispielsweise verbinden sich extern arbeitende Mitarbeiter via SSH mit dem Firmennetzwerk, um so Zugriff auf die Testrechner zu bekommen. Ein Zugriff auf das Webinterface wäre zwar mittels SSH-Tunneling möglich, ist aber nicht praxistauglich. Aus diesem Grund entwickeln wir ein kleines Kommandzeilen-Tool, das die wichtigsten Funktionen des Powerman nutzen kann. Die wichtigsten Funktionen sind das

- starten
- herunterfahren
- powercyclen
- abbrechen eines Shutdown-Befehls

eines Rechners. Das CLI wird vollständig in PHP entwickelt. Auf diese Weise lassen sich bereits bestehende Konzepte aus dem Frontend weiterverwenden und parallel nutzen. So muss für das CLI nicht separat ein Adapter für die Datenbankverbindung geschrieben werden und gewisse Funktionen (z.B. zur Benutzerüberprüfung) können ohne Änderung verwendet werden. Das CLI akzeptiert vier Parameter: Benutzername, Passwort, Client und Aktion. Alle Parameter sind optional, d.h. sie müssen nicht direkt in der Kommandzeile eingegeben

[1] Bei Monat schwankt es zwischen 672, 696, 720, 744 Stunden, bei Jahr zwischen 8760 und 8784 Stunden, je nach Monat und Schaltjahr

werden. Fehlende Parameter fragt das Tool in der Eingabeaufforderung ab. Sollte die Angabe des Rechnernamens fehlen, so erscheint eine Liste mit allen Rechner und Ihrem dazugehörigen Status. Der Benutzer kann dann mittels Pfeiltasten komfortabel durch die Liste browsen und den gewünschten Rechner auswählen.

Listing 5.19: Aufruf des Powerman CLI

```
php powerman-cli.php [--user=amasol] [--pass=secret] [--c=client] [start|stop|
    powercycle|abort]
```

5.5 Kennlinienerfassung

Ursprünglich war der Plan, den genauen Stromverbrauch jedes angeschlossenen Rechners zu erfassen, um so eine aktuelle IST-Situation zu protokollieren und Prognosen für die Zukunft zu erstellen. Wie aber bereits aus der Hardware-Marktanalyse (-verweis auf kapitel-) hervorgeht gibt es keine günstige Hardware, die verbrauchte Watt messen und in digitaler Form bereitstellen kann. Da im Rahmen des Projekts keine teure Hardware eingekauft werden kann, da sonst kein Einspareffekt vorhanden ist, muss für die Stromverbrauchsmessung eine Alternative gesucht werden.

Als günstige Alternative bot sich die sogenannte Kennlinienerfassung an. Hierbei wird im Vorfeld für jedes Rechnermodell eine charakteristische Stromverbrauchskennlinie erzeugt. Diese Kennlinie kann dann später dazu benutzt werden, aus den aufgezeichneten Daten (u.a. CPU-Auslastung) auf den momentanen Sromverbrauch zu schließen. Natürlich erreicht man damit nur eine Näherung an den tatsächlichen Stromverbrauch, aber für eine erste Abschätzung ist das Verfahren ausreichend.

Um eine Kennlinienerfassung durchzuführen, ist ein präzises Leistungsmessgerät erforderlich. Das für das Vorher / Nachher-Abbild verwendete ELV EM-600 ist dabei nicht ausreichend wie sich in Tests gezeigt hat. Durch die permanenten Leitungsschwankungen war es nicht möglich exakte Messpunkte zu definieren. Durch die Partnerschaft unserer Universität mit dem Leibniz Rechenzentrum (LRZ) war es uns möglich, eines der qualitativ sehr hochwertigen Strommessgeräte auszuleihen und für unsere Messung zu benutzen.

5.5.1 Messgerät

Abbildung 5.3: Leistungsmessgeräte Serie 2xx, Quelle: Hersteller

Es handelt sich dabei das ein Leistungsmessgerät CLM221 von Christ Elektronik. Das Gerät beherrscht 14 Messmodi, darunter Wirkwiderstand, Scheinwiderstand, Spannung, Strom, Leistungsfaktor, Blindleistung, Wirkenergie. Der Neupreis beträgt €280. Über einen 3,5mm Klinkenstecker kann das Messgerät seine Messdaten an die serielle Schnittstelle eines PC's schicken. Leider ist dieses Kabel nicht im Standard-Lieferumfang enthalten und musste von uns zunächst noch nach Herstellerschaltplan gebaut werden.

Nach Anschluß an den PC sendet das Messgerät nun kontiunierlich im 1-Sekunden-Rhythmus alle gemessenen Parameter an den PC. Um die empfangenen Daten zu visualisieren genügt ein einfaches Konsolenprogramm (z.B. unter Windows „HyperTerminal").

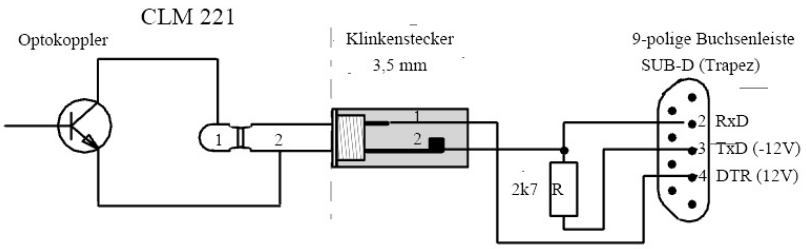


Abbildung 5.4: Schaltplan für den Anschluß an die serielle Schnittstelle, Quelle: Hersteller

Listing 5.20: Beispiel-Messdatenblock, der über die serielle Schnittstelle geschickt wird

```
W      00043.0
kWh    1.63176
var    00023.6
kvarh  1.57783
h      37.6986
VA     00049.0
kVAh   2.37114
cos    000.877
V      00226.9
A      0000.22
```

5.5.2 Test-Programm

Nachdem nun die Hardware einsatzbereit ist, muss noch eine Test-Software entwickelt werden. Dazu muss zunächst geklärt werden, welche Hardware-Komponenten an einem PC Einfluß auf den Strombedarf haben. Die nachfolgende Auflistung stellt eine Auswahl der größten Einflussfaktoren dar:

- CPU-Belastung
- Festplatten-Aktivität
- Netzwerk-Aktivität (LAN und WLAN)
- angeschlossene Peripherie
- Grafikanwendung

Im Vergleich mit allen aufgelisteten Punkten sorgt die CPU-Belastung für die größte Stromschwankung. Aus diesem Grund haben wir uns entschieden ausschließlich die CPU-Belastung und den dazugehörigen Strombedarf zu protokollieren und daraus eine charakteristische Kennlinie für den Rechner zu erzeugen. Das Test-Programm zur Erzeugung von CPU-Last verwendet zwei ineinandergeschachtelte Schleifen, die bis 2^{16} zählen. In der innersten Schleife wird eine Addition durchgeführt. Würde man das Programm allerdings so starten, würde es mit 100% CPU-Last laufen. Deshalb wird der Prozess nach der Addition für einige Millisekunden pausiert. Je nach Größe der Pause erreicht man damit eine konstante Last $< 100\%$. Die Wartezeit wird dem Programm als externer Parameter mitgegeben.

Listing 5.21: Load-Generator-Software

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <unistd.h>

int main (int argc, char** argv){
        unsigned int x,y,z,mult;
        if (argc < 2)
```

```
                return 0;

        mult = atoi(argv[1]);
        fprintf (stderr,"%s          ",argv[1]);
        mult = 200000 - mult*100;
        fprintf (stderr,"%d\n",mult);
        z = 0;
for (y = 0; y < pow(2,16); y++){
        for (x = 0; x < pow(2,16)-1; x++)
        {
                z++;
                printf("%d\n",z);
        }
        usleep(mult);
}
        return 0;
}
```

5.5.3 Messungen

Bei den Messungen wurde der zu testende PC über das zwischengeschaltete Strommessgerät an eine unterbrechungsfreie Stromversorgung angeschlossen, um etwaige Spannungs- und Netzschwankungen auszugleichen. Außerdem wurden möglichst alle Hintergrundprogramme auf dem PC gestoppt, sowie Peripheriekomponenten abgeschalten, um störende Einflüsse zu vermeiden. Nachdem der unbelastete Zustand (IDLE) notiert wurde, wurde das Last-Programm mit verschiedenen Pausierungszeiten gestartet. Parallel wurde auf einem zweiten Terminal / Fenster die CPU-Auslastung verfolgt. Wenn sich der Wert nach einigen Sekunden eingependelt hatte wurde der Messpunkt aufgezeichnet.

Core Duo-Laptop (zum Testen)

Bevor wir mit den Rechnern bei amasol begonnen haben, wurde zunächst ein Test mit einem eigenen Notebook durchgeführt. Es handelt sich dabei um ein Intel CoreDuo Notebook mit 2GHz CPU. Da es sich um eine Mehrkernarchitektur handelt, wurde der Test zweimal durchgeführt: Beim ersten Messdurchgang wurde das Last-Programm einmal gestartet, beim zweiten Durchgang zweimal, um beide Kerne auslasten zu können.

Sleep-Time	CPU1-Last	Watt	CPU2-Last	Watt
IDLE	0%	45,5	0%	65,4
199,9ms	8%	47	7,5%	66,2
140,0ms	11,7%	47,8	10,6%	66,6
80,0ms	17,2%	49	17,3%	67
40,0ms	29%	51,4	-	-
25,0ms	42%	54	-	-
20,0ms	47%	55,2	44%	68,9
10,0ms	63%	58,4	59,2%	71,4
7,5ms	69%	59,8	-	-
2,5ms	82,7%	63,7	-	-
0,5ms	92%	65,5	-	-
0,0ms	-	-	93,3%	70,3

Tabelle 5.4: Last-Messergebnis für CoreDuo Laptop

amasol Sun

Die SUN-Rechner von amasol zeigen im Gegensatz zu dem Notebook aus dem vorherigen Kapitel ein ganz anderes Verhalten. Egal wie hoch die CPU-Last der UltraSPARC IIe Prozessoren ist, verbrauchen sie immer konstant viel. Zudem war es schwierig, mit unserem Last-Programm die Auslastung über 85% zu bringen. Das liegt daran, dass Teile der CPU nicht für die Rechenoperationen benötig werden und brach liegen.

Sleep-Time	CPU-Last	Watt
IDLE	0%	43
1000ms	11%	42,8
199,9ms	36%	42,8
160,0ms	41%	42,7
140,0ms	44%	42,7
100,0ms	50%	42,7
80,0ms	56%	42,7
60,0ms	63%	42,8
40,0ms	71%	42,6
20,0ms	79,4%	42,8
10,0ms	85%	42,6
5,0ms	85%	42,7
0,1ms	85%	42,7
0,0ms	99,8%	42,7

Tabelle 5.5: Last-Messergebnis für SUN-Rechner von amasol

Für die SUN-Rechner ergibt unsere Kennlinie eine konstante Funktion $y = 43$

amasol PC

Die PC's sind im Gegensatz zu den SUN-Rechnern deutlich variabler im Stromverbrauch bei unterschiedlicher CPU-Auslastung. Zudem fällt der deutlich höhere Grundverbrauch auf.

Sleep-Time	CPU-Last	Watt
IDLE	0%	94,9
199,9ms	4,7%	96,6
160,0ms	11%	97,2
120,0ms	14,3%	94,5
80,0ms	19,9%	97,8
60,0ms	24,9%	99,1
40,0ms	33,2%	100,4
30,0ms	39,9%	102,4
20,0ms	49,8%	104,3
10,0ms	66,4%	107,3
5,0ms	66,4%	107,9
1,0ms	66,4%	107,3

Tabelle 5.6: Last-Messergebnis für PC's von amasol

Durch eine reine Addition in der Schleife des Last-Programms war es nicht möglich die Auslastung auf mehr als 66% zu erhöhen. Auch hier werden einige Bereiche der CPU durch unsere Addition nicht benutzt. Da wir aber möglichst nahe an 100% CPU-Last herankommen wollten, haben wir das Test-Programm um eine Multiplikation in der Schleife erweitert. Somit gelang es uns immerhin 93% der CPU zu beanspruchen.

Sleep-Time	CPU-Last	Watt
199,9ms	40,2%	106,1
80,0ms	63,8%	107,3
30,0ms	82,1%	111,3
10,0ms	86,4%	114
1,0ms	93%	112,5

Tabelle 5.7: Last-Messergebnis für PC's von amasol (Multiplikations-Verfahren)

Die charakteristische Kennlinie für die PC's bei amasol läßt sich nicht so einfach wie bei bei den SUN-Rechnern ermitteln. Um eine Formel für die Kennlinie zu erstellen haben wir verschiedene Polynom-Kurven über die Messdaten gelegt via „best-fit" die passendste Kurve ausgewählt. Somit lautet die Formel für die charakteristische Kennlinie: $y = 0,0022 * x^2 + 98$

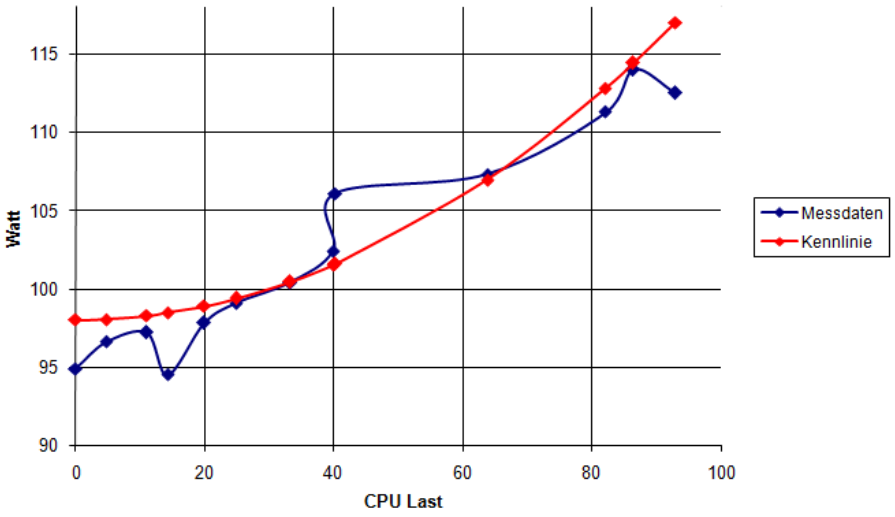


Abbildung 5.5: Messpunkte und Kennlinie des PCs

5.6 Temperatursensor

Wie bereits in 3.1.2 geschildert, handelt es sich bei dem 1Wire-Temperatursensor um ein Bastelprojekt aus dem Internet. Der Schaltplan besteht dabei nur aus 6 Komponenten. Dadurch lässt sich eine sehr kompakte Bauweise realisieren. Unsere Lochrasterplatine (Standardlochraster 2,54mm) hat gerade mal eine Größe von 2,5cm x 1,7cm.
Ein weiterer Vorteil des Dallas-Temperatursensors ist seine Kaskadierbarkeit. Man kann beliebig viele Sensoren parallel an die Schaltung klemmen. Jeder Sensor besitzt seine eigene Seriennummer und kann so individuell abgefragt werden.

Die Abfrage der Sensoren an der seriellen Schnittstelle erfolgt mit Hilfe der Open-Source-Software „digi-

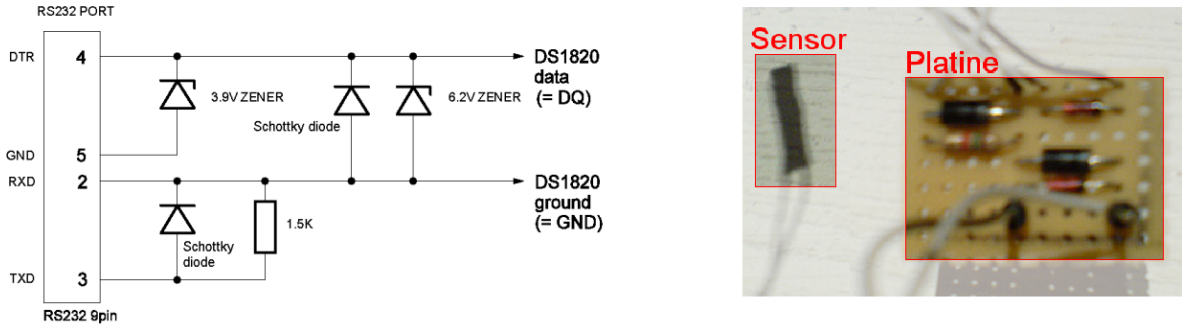


Abbildung 5.6: Fertig aufgebaute Schaltung mit Schaltplan

temp“. Vor der ersten Messung muss zunächst die Software einmal initialisiert werden. Das geschieht nach Eingabe des Kommunikationsports mit dem Chip weitgehend automatisch. Die gefunden Sensoren werden angezeigt. Durch Anpassen der erstellten Konfigurationsdatei kann die Ausgabe des Messdaten von digitemp angepasst werden.

Listing 5.22: Initialisierung des Temperatursensors

```
test-smithers:/var/powerman/digitemp # ./digitemp_DS9097 -s/dev/ttyUSB0 -i
DigiTemp v3.5.0 Copyright 1996-2007 by Brian C. Lane
GNU Public License v2.0 - http://www.digitemp.com
Turning off all DS2409 Couplers
.
Searching the 1-Wire LAN
10211D25010800C6 : DS1820/DS18S20/DS1920 Temperature Sensor
ROM #0 : 10211D25010800C6
Wrote .digitemprc
```

Listing 5.23: Aufbau der Konfigurationsdatei von Digitemp

```
test-smithers:/var/powerman/digitemp # cat .digitemprc
TTY /dev/ttyUSB0
READ_TIME 1000
LOG_TYPE 1
LOG_FORMAT "%b %d %H:%M:%S Sensor %s C: %.2C F: %.2F"
CNT_FORMAT "%b %d %H:%M:%S Sensor %s #%n %C"
HUM_FORMAT "%b %d %H:%M:%S Sensor %s C: %.2C F: %.2F H: %h%%"
SENSORS 1
ROM 0 0x10 0x21 0x1D 0x25 0x01 0x08 0x00 0xC6
test-smithers:/var/powerman/digitemp #
```

6 Fazit

Mit Spannung wurde nach der Implementierphase die tatsächliche Nutzung durch amasol erwartet. Schließlich hatte niemand eine genaue Vorstellung, ob und wieviel tatsächlich eingespart werden kann. Denn bereits unsere Anwesenheit in der Firma, als wir uns in die Topologie eingearbeitet haben, hat laut unserem Betreuer dazu geführt, dass sich die Angestellten zum ersten Mal der Thematik „Energiesparen“ bewusst wurden. Dadurch wurden bereits vor der ersten Stromverbrauchsmessung im Serverraum die Rechner plötzlich häufiger abgeschalten und nur bei Bedarf hochgefahren. Damals noch manuell am Gerät selbst, heute geschieht das meist durch unser Frontend.

6.1 Ergebnisse der Nachher-Messreihe

Messtag	gemessene Zeitspanne	Stromverbrauch
27.06.2007	52 h	13,53 kWh
04.07.2007	164 h	38,25 kWh
12.07.2007	185 h	36,77 kWh
18.07.2007	147 h	23,45 kWh
25.07.2007	167 h	14,32 kWh
01.08.2007	168 h	17,33 kWh
09.08.2007	190 h	23,15 kWh
16.08.2007	167 h	22,21 kWh
13.09.2007	672 h	99,32 kWh
19.09.2007	144 h	39,80 kWh
26.09.2007	171 h	34,09 kWh

Tabelle 6.1: Messergebnisse der Nachher-Situation

Laut amasol wurde am 11.09.2007 zusätzlich ein Router für den Kundenzugang im Serverraum installiert, der an die bestehende Messapparatur gehängt wurde. Da der Stromverbrauch des Routers nicht gemessen werden konnte, sind die ab diesem Zeitpunkt Werte schwer einzuordnen, da sich durch den Router beeinflusst wurden. Trotzdem konnte der mittlerweile Stromverbrauch pro Woche im Mittel von 53,71 kWh auf 29,71 kWh gesenkt werden. Das entspricht einer Einsparung von 45%. Auf das Jahr hochgerechnet, kann die Firma in der aktuellen Konfiguration (und wenn man den Messfehler durch den Router nicht berücksichtigt) €187,39 an Stromkosten einsparen. Anders ausgedrückt zahlt die amasol AG nur noch für sieben Monate im Jahr Stromkosten für die Testserver, die verbleibenden fünf sind kostenlos.

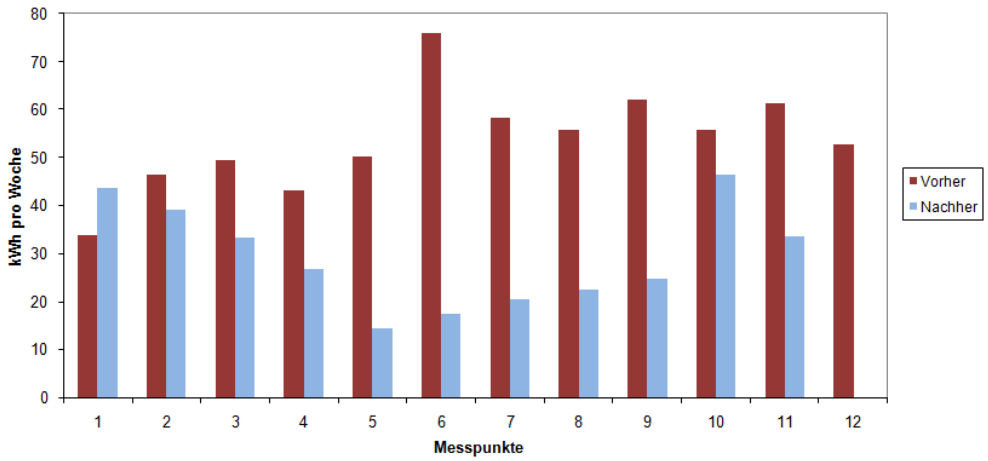


Abbildung 6.1: Vorher-Nacher Messung im direkten Vergleich

6.2 Abschlussbewertung

Aus Entwicklersicht handelt es sich um ein erfolgreiches Projekt. Alle Anforderungen der Auftraggeber konnten erfolgreich umgesetzt werden, wie ein Blick auf die Anforderungsliste aus Kapitel 2 zeigt. Dazu zählt in erster Linie die Entwicklung der Kernkomponenten (modulare zentrale Managementeinheit, policybasierte Steuerung), die Programmierung eines Web-GUI, die Bereitstellung eines CLI aber auch die Evaluierung des Marktes nach günstigen Hardware-Lösungen für Steckdosensteuerung und Temperaturmessung.

Lediglich bei Anforderungs-ID 02 („Simples Benutzermanagement“) und ID 14 („Kompaktes Benutzermanagement“) mussten wir einen Kompromiss finden, da sie die beiden Anforderungen widersprochen haben. Auf der einen Seite wurde ein einfaches Authentifizierungssystem gefordert. In diesem System sollte genau eine Rolle mit vollen Rechten vorhanden sein. Auf der anderen Seite gab es die Anforderung nach mehreren Rollen mit unterschiedlichen Rechten (Administratoren und normale Benutzer). Nach Rücksprache mit den Auftraggebern wurde entschieden, Anforderungs-ID 14 umzusetzen und das simple Benutzermanagement zu verwerfen.

Trotz erfolgreichem Projektabschluss ist das Powerman System keinesfalls perfekt und es gibt noch zahlreiche Gelegenheiten zum Verbessern und Weiterentwickeln. Durch die Modularität des System sollte das für zukünftige Entwickler kein Problem darstellen. Wir selbst haben bereits einige Ideen, um das System noch zu erweitern. Beispielsweise könnte man auch Virtual Machines vom Powerman System starten und herunterfahren lassen. Hierzu muss lediglich ein kleines Modul geschrieben werden, dass als Managementkomponente ins System eingebunden wird.

6.3 Eingliederung der Arbeit in GreenIT

Gerade in den letzten Monaten gab es vermehrt öffentliche Diskussionen über Energieverbrauch und Emissionen, die unter dem Stichwort CO_2-Debatte in der Tagespresse zu finden sind. Die Industriezweige, denen man hohe CO_2-Emissionen nachsagt, versuchen durch geeignete Maßnahmen umweltfreundlichere Geräte zu entwickeln. In der Automobilindustrie heißen diese Maßnahmen „Efficient Dynamics“ oder „BlueMotion“, abhängig vom Hersteller. In der IT-Branche gibt es dafür den einheitlichen Begriff „GreenIT“.

Unter dem Schlagwort „GreenIT“ versteckt sich ein breit gefächertes Gebiet, das sich ganz allgemein mit den ökologischen Aspekten der Computer- und Kommunikatinosbranche auseinandersetzt. Denn nicht nur in anderen Branchen, wie z.B. der Automobilbranche, werden die Umweltaspekte immer wichtiger, sondern

auch im IT-Bereich. Rechenzentren werden immer größer und leistungsfähiger, genau so wie deren Strom- und Kühlungsbedarf, was letzendlich zu einer höheren Umweltbelastung führt. GreenIT versucht eine

- effiziente Nutzung der Hardware
- Verwendung besonders energieeffizienter Hardware
- energieeffiziente Kühlung
- sinnvolle Nutzung der Abwärme
- umweltgerechte Entsorgung von Altgeräten

zu propagieren und durchzusetzen. Führende IT-Konzerne wie IBM, AMD, Dell, Intel, Sun investieren bereits erhebliche Summen in umweltschonende Initivativen wie energieeffiziente Server, Strom- und Kühllösungen für Rechenzentren . [WEB 5] Denn immer mehr Firmen achten beim Einkauf Ihrer Hardware darauf, energiesparende bzw. energieeffiziente System einzukaufen.

Neben reinen Hardware-Ansätzen entwickeln einige Hersteller bereits auch an Software-Konzepten um unter anderem das Energiemanagement zu verbessern. Intel arbeitet dabei an einer Technik namens „Active Management Technology“ (Intel AMT). Mittels dieser Technik lassen sich u.a. PCs remote ein/ausschalten bzw. resetten, BIOS-Einstellungen vornehmen, Informationen über das System abrufen und remote Zugriff auf den Computer zu nehmen (Übernahme von Maus/Tastatur und Bildschirm). Das System kann dabei via HTTP-Schnittstelle auch im ausgeschalteten Zustand erreicht und kontrolliert werden. [WEB 6] Intel AMT gliedert sich dabei aber in eine größere Komponenten-Lösung ein, die von Intel „vPro“ bezeichnet wird und sich in allen Business-PCs wiederfinden soll.

Laut Intel soll die Ein/Ausschaltfunktion über AMT langfristig Wake-On-Lan - und die damit verbundenen Probleme über Routergrenzen hinweg - ablösen. Allerdings sind die zum Teil sehr guten Konzepte von Intel nicht ISO-standardisiert und funktionieren damit nur mit Intel-Hardware. Somit müssen Unternehmen auch in naher Zukunft noch auf Fremdlösungen beim Energiemanagement setzen. Mit dem von uns entwickelten Powerman-System haben wir gezeigt, dass bereits eine kleine Lösung genügt um Energiekosten einzusparen. Somit konnte das Powerman System einen kleinen Beitrag auf dem Weg zur GreenIT leisten.

A Dokumentation für Benutzer

Die Hauptaufgabe des Powerman Systems ist es, in einer größeren Umgebung die Kontrolle über das Ein- und Ausschalten von Servern zu zentralisieren und zu automatisieren. Dadurch erreicht man eine bessere Kontrolle über seine Server und deren Stromverbrauch. Der Hauptteil des Systems läuft zentral auf dem Powerman-Server, die zu verwaltenden Rechner sind die Clients.

Im ersten Ansatz wird mit Hilfe des Powerman Systems der Zugang zu den Rechnern und die Übersicht verbessert, so dass das manuelle Energiemanagement leichter fällt. Außerdem kann das System das Energiemanagement automatisieren. Dabei richtet es sich nach drei Kontrollschichten, den Grundbetriebszeiten als unterste Schicht, den Heuristiken und den Ausnahmen, welche die höchste Kontrollschicht darstellen.

Grundbetriebszeiten sind ein Wochenplan, mit einer Liste von Zeitspannen, zu denen ein Computer an sein soll. Grundbetriebszeiten werden getrennt von Computern verwaltet, d.h. damit ein Computer einer Grundbetriebszeit ensprechend gesteuert wird, muss dem Computer diese Grundbetriebszeit explizit zugewiesen werden.

Der Powerman-Server sammelt Informationen über die Clients. Diese Informationen sind die Entscheidungsgrundlage der zweiten Kontrollschicht (Heuristiken) zur Steuerung der Clients. Heuristiken bestehen immer aus einem Prädikat (eine Aussage, die zu jedem Zeitpunkt entweder wahr oder falsch ist), zusammen mit der Angabe eines Zeitraums und einer Auswirkung. Ist dieses Prädikat für einen bestimmten Zeitraum wahr (aktiv), so hat dies eine Auswirkung auf die Entscheidung der Steuerlogik. Dem Benutzer stehen alle Möglichkeiten aus Tabelle A für Heuristiken zur Verfügung.

Tabelle A.1: Heuristiken im Frontend, ohne Zeitattribut

Merkmal	**Herunterfahren bei**			**Neustarten bei**			**Aktionen blocken bei**		
Benutzer	unter	X	eingeloggt	über unter	X	eingeloggt	über	X	eingeloggt
CPU-Load	unter	X	%	über unter	X	%	über	X	%
Net-Load	unter	X	%	über unter	X	%	über	X	%
Temperatur	unter	X	Grad	über unter	X	Grad			
Zombie Prozesse	über	X		über	X				
$< Prozessname >$	läuft	nicht		läuft	nicht				

Die höchste Kontrollschicht sind Ausnahmen. Sie setzen sich sowohl über Grundbetriebszeiten, als auch Heuristiken hinweg und beschreiben einen Zeitraum, in dem ein zu verwaltender Recher durchgehend an oder aus sein soll.

Das Powerman System besteht aus vier Hauptkomponenten. Die am meisten sichtbare Komponente ist das Frontend. Dazu kommen die Software auf den Clients, die Software auf dem Server und die Managementkomponente. Im vorliegenden Fall ist die Managementkomponente entweder eine Sun LOM (Lights Out Management), oder eine schaltbare Gembird Steckerleiste.

Direkte Kontrolle über die Computer steht dem Benutzer im Powerman Frontend zur Verfügung. Im Frontend hat der Benutzer ausserdem die Kontrolle über die Automatisierung. Die Clientsoftware macht das System reaktiv. Sie stellt Informationen, die vom Server abgerufen werden, bereit und nimmt Befehle vom Server entgegen. Die Serversoftware ist der aktivste Teil des Systems. Sie sammelt Informationen, schickt Befehle und übernimmt die Automatisierung des Energiemanagements. Unterstützt wird sie dabei von den Managementkomponenten, die dem System mehr Gewalt über die Clients geben.

A.1 Übersicht — Startseite

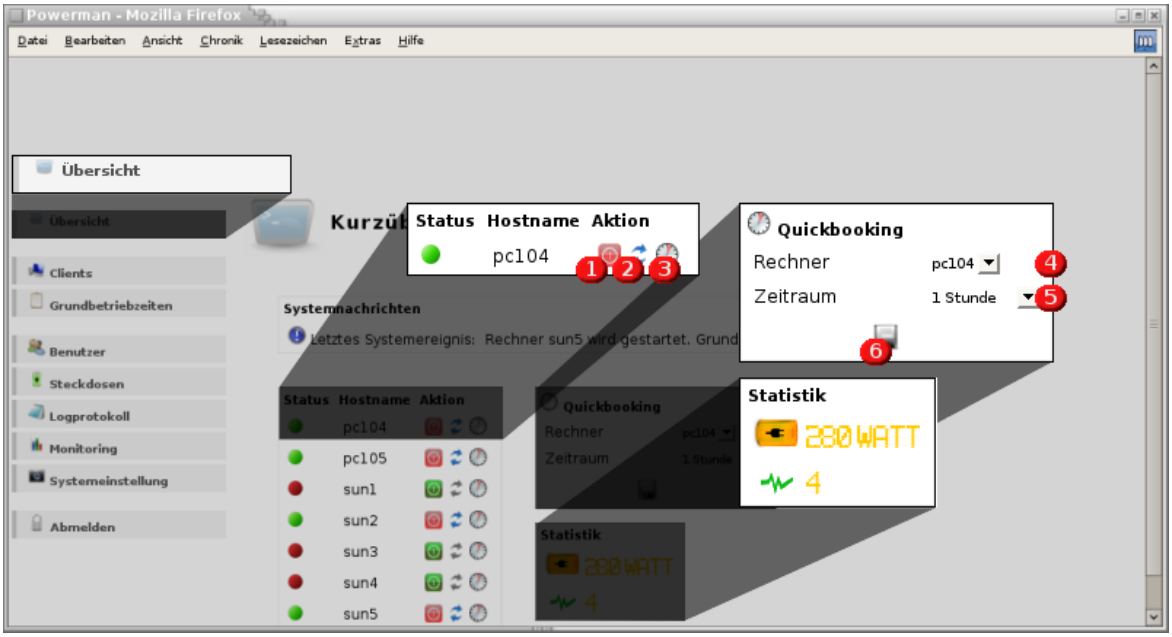


Abbildung A.1: Übersichtseite

Nach dem erfolgreichen Anmelden kommt man auf die übersichtsseite. Hier bekommt der Benutzer eine kurze übersicht über die dem Powerman-Server bekannten Rechnern, zusammen mit ein paar Informationen und Funktionen, gegliedert in vier Blöcke.

A.1.1 Beschreibung

A.1.1.1 Systemnachrichten

Der oberste Block der Übersichtsseite ist „Systemnachrichten“ . Dieser Block beinhaltet zwei Informationen. Zum einen teilt das System dem Benutzer hier mit, ob und wie viele neue Rechner sich am Powerman-Server angemeldet haben. Außerdem kann man hier die letzte eingetragene Systemnachricht sehen. Eine chronologisch geordnete Liste aller Systemnachrichten gibt es unter Kapitel A.6.

A.1.1.2 Clients

Der nächste Block ist eine Liste aller dem Powerman-Server bekannten Rechnern. Die Spalte Status gibt Auskunft über den Zustand des jeweiligen Clients. Diese Spalte benutzt nur Icons um Informationen darzustellen. Eine Liste aller Symbole findet sich in Tabelle A.2.

Tabelle A.2: Symbole in der Spalte Status

Symbol	Bedeutung
	Rechner ist an
	Rechner fährt hoch
	Rechner fährt herunter
	Rechner ist aus
	Rechner ist nicht konfiguriert

A.1.1.3 Quickbooking

Um kurzfristig einen Client zu benutzen, kann man mit einem Quickbooking schnell eine Ausnahme für den Client definieren, so dass man sich für den Zeitraum der Nutzung keine Gedanken über Grundbetriebszeiten und Heuristiken machen muss.

A.1.1.4 Statistik

Das Feld „Statistik“ zeigt dem Benutzer den geschätzten gegenwärtigen Stromverbrauch in Watt und die Anzahl der verwalteten hochgefahrenen Clients.

Des Weiteren gibt es eine Anzeige, die eine Schätzung, wieviel Strom im aktuellen Monat gespart wurde, enthält. Wir gehen davon aus, dass ohne das Powerman-System jeder Rechner zu den Geschäftszeiten durchgehend an ist. Die tatsächliche Betriebsdauer und Belastung der Rechner ist dem Powerman-Server bekannt. Der daraus errechnete tatsächliche Stromverbrauch wird mit dem Dauerbetrieb verrechnet. Die Differenz ist die Ersparnis, die durch den Powerman ermöglicht wurde. Diese Ersparnis wird zusätzlich direkt in einen Geldbetrag umgerechnet. Mit dieser Anzeige kann man leicht den gewonnen Vorteil durch die Benutzung des Powerman-Systems direkt ablesen. Da diese Anzeige nachträglich eingefügt wurde, befindet sie sich nicht auf dem Bildschirmfoto.

A.1.2 Funktionen

Die dem Benutzer hier zugänglichen Funktionen und Symbole werden in Tabelle A.3 erläutert. Einige Funktionen sind in Abbildung A.1 zu sehen und dort mit Markierungen versehen, welche sich auch in der Tabelle wieder finden.

A.2 Clients

Unter dem Menüpunkt „Clients“ findet man eine detailierte Liste aller dem Powerman-Server bekannten Rechner, was das Identifizieren von Rechnern vereinfachen soll. Zusätzlich zu den Informationen, die auch auf der Startseite angezeigt werden, zeigt diese Liste auch Informationen über das Betriebssystem, Mac- und IP-Adresse, sowie die dem Client zugeordnete Managementkomponente. Die Funktionen auf dieser Seite (Abbildung A.2) sind in der Tabelle A.2 aufgeführt.

A.2.1 Zeit zuweisen

Hier befinden sich alle clientspezifischen Funktionen zur Steuerung der drei Kontrollschichten. Abbildung A.3 zeigt die Liste eines Rechners, für den keine Einstellungen gemacht wurden. Die Funktionen dieser Sei-

Tabelle A.3: Symbole und Funktionen

Markierung	Symbol	Bedeutung
-		Rechner anschalten
1		Rechner ausschalten
-		(keine Funktion)
-		Vorgang abbrechen
2		Rechner powercyclen
3		Quickbooking für eine Stunde
-		Quickbooking aufheben
4	-	Rechner für Quickbooking auswählen
5	-	Dauer des Quickbookings auswählen
6		Quickbooking speichern

Tabelle A.4: Funktionen der Clientliste

Markierung	Symbol	Bedeutung
1		A.2.1 Zeit zuweisen (nur priviligierte Benutzer)
2		A.2.2 Einstellungen
3		A.2.3 Rechner löschen (nur priviligierte Benutzer)

te sollen nun Schritt für Schritt erläutert werden, indem wir einem imaginären Beispielrechner zuerst eine Grundbetriebszeit, danach Heuristiken und zum Schluss eine Ausnahme zuweisen.

A.2.1.1 Grundbetriebszeit hinzufügen

Folgt man dem Link „Grundbetriebszeit hinzufügen“ (Abbildung A.3 Markierung 1), so gelangt man zur Eingabemaske aus Abbildung A.4. Zuerst wählen wir eine Grundbetriebszeit, welche wir zuvor erstellt haben (siehe Kapitel A.3) aus, bestimmen dann den Zeitraum für den die Grundbetriebszeit bei diesem Client Anwendung findet und klicken abschließend den Speichern-Button[1] um die Zuweisung zu speichern.

A.2.1.2 Heuristiken

Der Link, der in Abbildung A.3 durch Markierung 2 angezeigt wird, führt zum Einstellungsbildschirm für Heuristiken. Die Eingabemaske in Abbildung A.5 ist eine Implementierung der schematischen Darstellung aus Tabelle A. In der ersten Spalte (Markierung 1) wählt der Benutzer die Merkmale aus, für die eine oder mehrere Heuristik(en) gelten sollen. Die Spalten zwei bis vier sind die Aktionen, die ausgelöst werden können. Um eine Heuristik zu definieren, muss der Benutzer im Feld (Merkmal, Aktion) die Bedingung und den Zeitraum, für den diese Bedingung gelten soll spezifizieren. In Abbildung A.5 zeigen die Markierungen 2 (Bedingung) und

[1] Die meisten Funktionen haben Speichern- und Zurück-Buttons, wie hier bei den Markierungen 4 und 5, und haben auch immer die selbe Funktion, weshalb wir sie nur hier exemplarisch für alle Funktionen erläutern

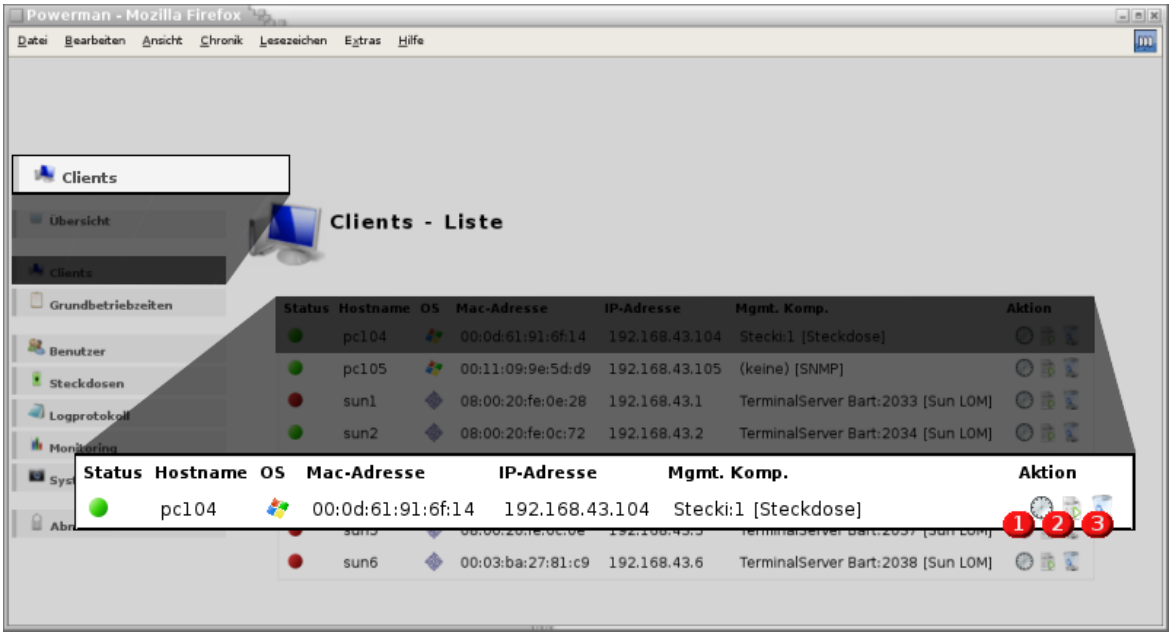
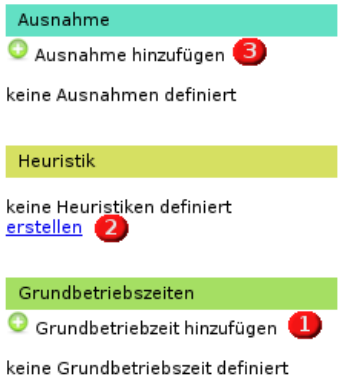


Abbildung A.2: Clients Liste



Markierung	Bedeutung
1	A.2.1.1 Grundbetriebszeit hinzufügen
2	A.2.1.2 Heuristik definieren
3	A.2.1.3 Rechner löschen

Abbildung A.3: Ansicht bei einem unkonfigurierten Rechner

3 (Zeitraum) das Feld an, in dem eine Heuristik für das Merkmal Benutzer definiert werden kann, welche den Computer bei Bedarf herunter fährt.

Des Weiteren unterstützen die Heuristiken auch Prozessmonitoring. Dieses muss zunächst aktiviert werden (Markierung 4). Nun kann man Prozessnamen als Merkmal eintragen. Von links nach rechts trägt man der Reihe nach den Prozessnamen, die Zeitspanne und die Aktion, die ausgeführt werden soll, wenn der Prozess für die angegebene Zeitspanne nicht läuft, ein. Um den Eintrag zu speichern, muss der Benutzer auf das Diskettensymbol (Markierung 6) klicken.

Alle Einstellungen finden erst Anwendung und werden erst endgültig im System gespeichert, wenn der Benutzer auf den Speichern-Button klickt, auf dessen Abbildung wir hier verzichtet haben.

Die Abbildung in Abbildung A.6 zeigt drei Heuristiken, die wir unserem Beispielrechner zuweisen wollen. Heuristiken, die Aktionen blocken stehen in der Hierarchie höher als alle anderen Heuristiken, d.h. solange eine Heuristik Aktionen blockt, sind Grundbetriebszeiten und andere Heuristiken für die Logik irrelevant. Markierung 4 soll lediglich auf den Button aufmerksam machen, mit dem man einzelne Prozesseinträge löschen kann.

Markierung	Bedeutung
1	Auswahlliste aller Grundbetriebszeiten
2	Eingabefelder für Start- und Enddatum des Gültigkeitszeitraums der Grundbetriebszeit. Anstatt das Datum per Hand einzutragen, kann auch der Kalender-Button benutzt werden
3	Anstatt sich auf ein Enddatum fest zu legen, kann man die Markierung der Checkbox entfernen. Dadurch wird der Gültigkeitszeitraum nicht nach oben beschränkt
4	Der Speichern-Button übernimmt die Einstellungen in das System und führt den Benutzer zurück zur vorherigen Anzeige
5	Der Zurück-Button führt den Benutzer zurück zur vorherigen Anzeige, ohne die Einstellungen zu speichern

Abbildung A.4: Grundbetriebszeit hinzufügen

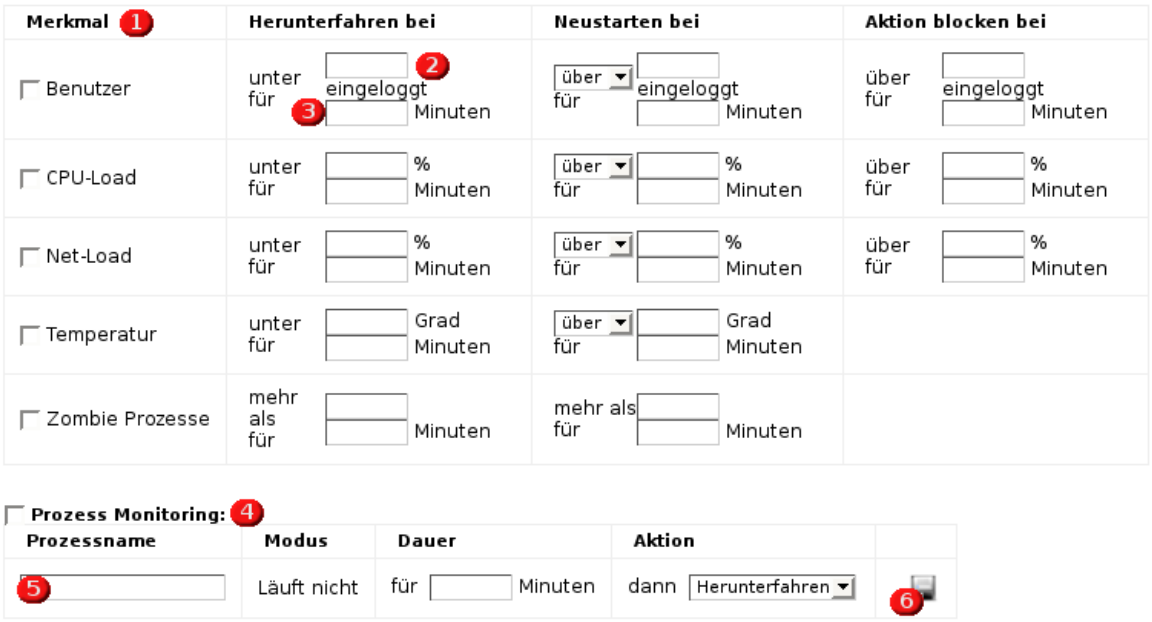
A.2.1.3 Ausnahmen

Als letzte Station richten wir noch eine Ausnahme für unseren Testrechner ein. Zu der zugehörigen Eingabemaske gelangen wir, indem wir dem Link aus Abbildung A.3 Markierung 3 folgen. Die Eingabemaske ähnelt sehr dem Zuweisen von Grundbetriebszeiten. Die Bezeichnung der Ausnahme ist frei wählbar (in unserem Fall „SimonSaysOff“). Ebenso wie bei Grundbetriebszeiten müssen ein Start- und ein Enddatum angegeben werden. Zusätzlich muss hier noch eine Uhrzeit angeben werden. Außerdem gibt es noch das Auswahlmenü „Modus“ (Abbildung A.7 Markierung 1) aus der man auswählen kann, ob der Client zwischen dem Start- und Endzeitpunkt immer an, oder immer aus sein soll. Die Abbildung A.7 zeigt auch den Kalender (Markierung 2), der zur Auswahl des Start- und Enddatums benutzt werden kann. Aufgerufen wird der Kalender durch einen Klick auf den Kalender-Button.

Nach all unseren Änderungen präsentiert sich die Liste der Zeit-zuweisen-Seite wie in Abbildung A.8 dargestellt. Die Markierungen zeigen die Bearbeiten- und Löschen-Symbole für zugewiesene Grundbetriebszeiten und eingerichtete Ausnahmen. Diese Symbole sind wie die Speichern- und Zurück-Buttons systemweit gleich und werden an dieser Stelle exemplarisch benannt. Die Eingabemasken der Bearbeiten-Funktion sind die selben wie beim Hinzufügen bzw. Zuweisen, jedoch mit bereits ausgefüllten Feldern. Löschenfunktionen fragen prinzipiell nach einer erneuten Bestätigung durch den Benutzer, bevor sie etwas auslösen.

A.2.2 Einstellungen

Diese Seite enthält alle clientspezifischen Einstellungen, die nichts mit der Zeitverwaltung zu tun haben. Neu am System angemeldete Rechner werden als unkonfiguriert markiert. Unkonfiguriert deswegen, weil das System noch nicht so viel mit ihnen anfangen kann. Erst nachdem ein Benutzer auf dieser Seite Einstellungen vornimmt und speichert, wird ein Rechner nicht mehr als unkonfiguriert markiert.



Markierung	Bedeutung
1	Auswahlliste aller Merkmale
2	Eingabefeld für Bedingung
3	Zeitspanne für die die Bedingung erfüllt sein muss, damit die Aktion durchgeführt wird
4	Prozessmonitoring (de)aktivieren
5	Eingabezeile für Prozessmonitoring
6	Prozessmonitoring speichern

Abbildung A.5: Unausgefüllte Heuristikeneingabemaske

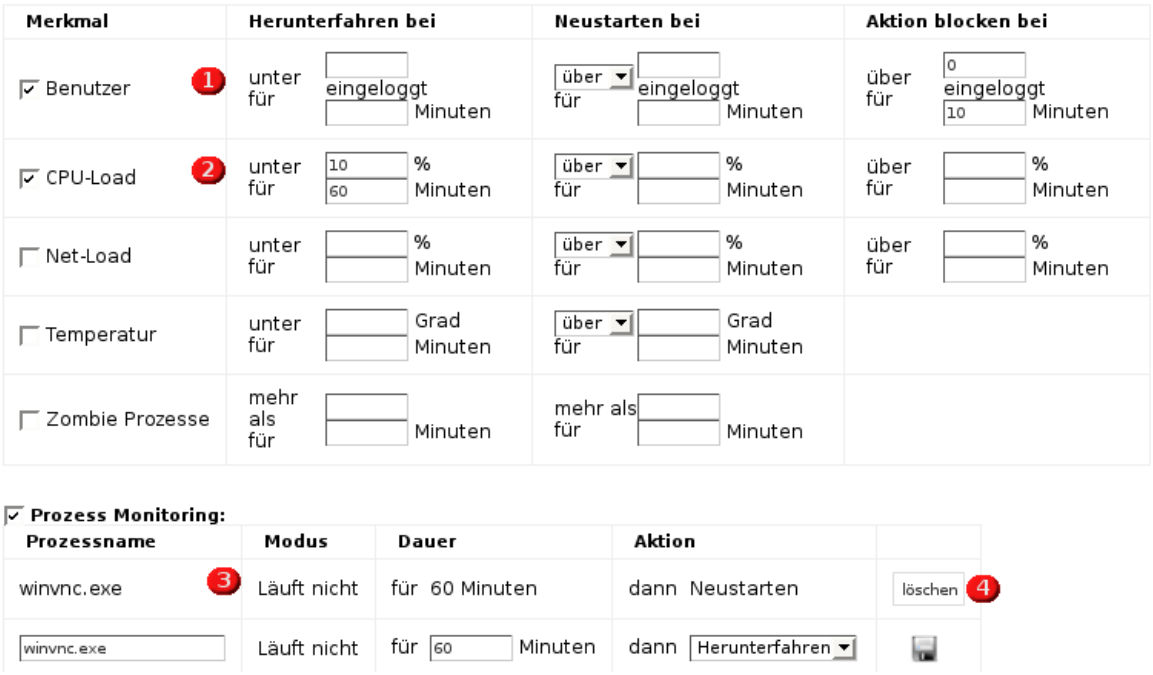
A.2.2.1 Systemnachricht

Dieses Textfeld steht allen Benutzern zur Verfügung. Die Systemnachricht soll das Arbeiten mit dem Powerman System vereinfachen. Sie ist eine Art virtuelle Post-It Notitz, die man einem Client zuordnet. Jeder der sich danach an diesem Rechner anmeldet bekommt die Systemnachricht zu sehen. Unter Windows ist die Systemnachricht in das Hintergrundbild eingebettet, auf Unix-Systemen wird sie beim Anmelden auf der Konsole ausgegeben. Außerdem interpretiert das System die dem Client zugewiesenen Grundbetriebszeiten und hängt selbstständig noch einen kleinen Wochenkalender mit den Zeiten, zu denen der Rechner vorraussichtlich eingeschaltet ist, an die Systemnachricht an.

A.2.2.2 Managementkomponente

Hier kann der Benutzer dem Client eine Managementkomponente zuweisen. Mit den runden Radioboxen wählt der Benutzer zunächst die Art der Managementkomponente, die er dem Client zuweisen möchte. Er hat dabei die Wahl zwischen keine, Gembird Steckerleiste oder Sun LOM. Die Verwaltung von Managementkomponenten wird in Kapitel A.5 behandelt. Es kann nie zwei Rechnern die gleiche Komponente zugewiesen werden.

Das erste Auswahlmenü, ist eine Liste aller dem System bekannten Gembird Steckerleisten. Jede Steckerleiste ist mit jeder ihrer Steckdose einmal in dieser Liste vertreten. Die Einträge in der Liste sind zusammengesetzt aus der Bezeichnung der Steckdose und der Nummer der Steckdose, die auch auf dem Gerät selber aufgedruckt



Markierung	Bedeutung
1	Blocke alle Aktionen sobald es mindestens einen Benutzer gibt, der seit mindestens 10 Minuten angemeldet ist.
2	Fahre den Computer herunter, wenn die Auslastung der CPU innerhalb der letzten 60 Minuten nie höher als 10% war.
3	Starte den Computer neu, wenn innerhalb der letzten 60 Minuten kein Prozess winvnc.exe gelaufen ist.
4	Prozessmonitoringzeile löschen

Abbildung A.6: Beispielheuristiken

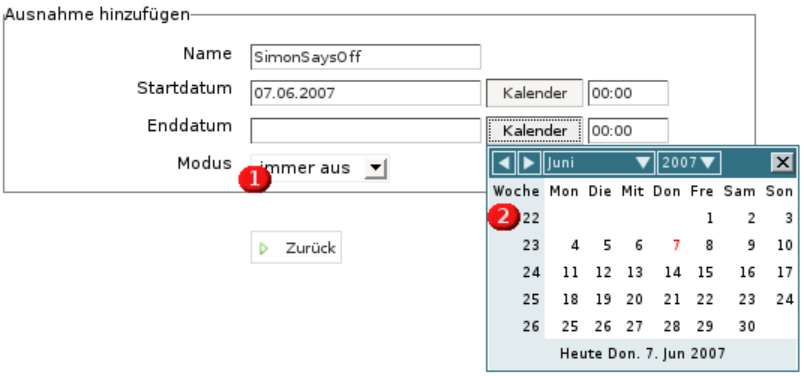


Abbildung A.7: Ausnahme für Client einrichten

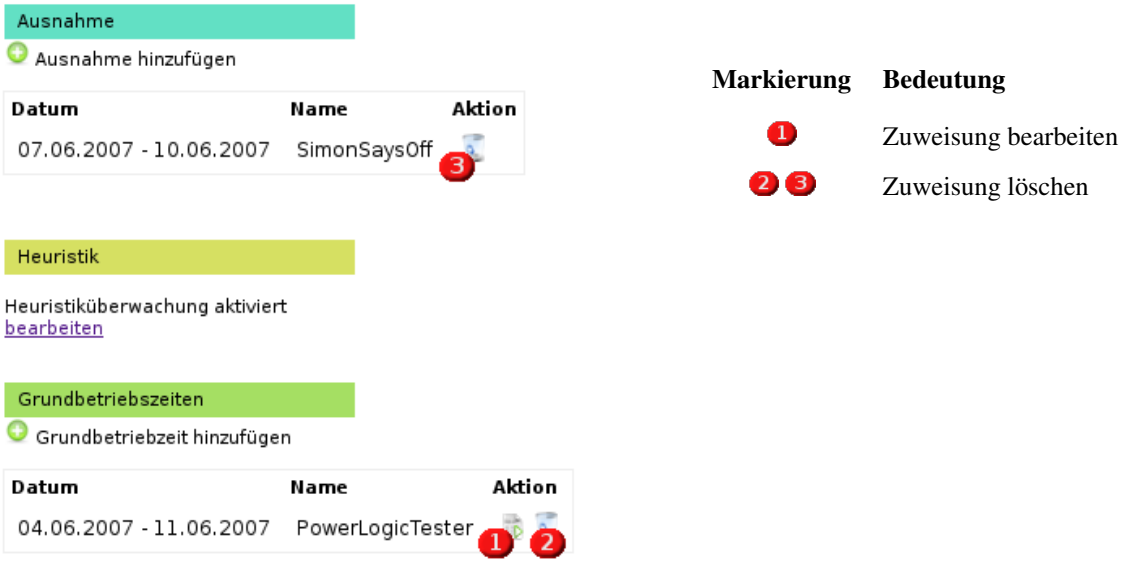


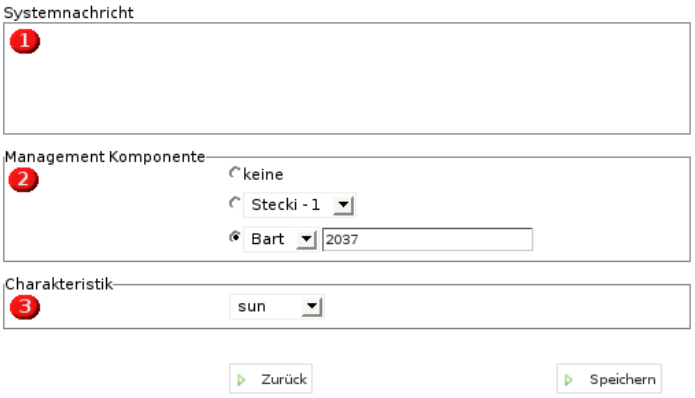
Abbildung A.8: Ansicht bei einem konfigurierten Rechner

ist. Eine Steckerleiste, im folgenden Beispiel „Stecki“, genannt, vier Steckdosen erscheint also vier mal in der Liste, mit den Einträgen „Stecki - 1“, „Stecki - 2“, „Stecki - 3“ und „Stecki - 4“ .

Das zweite Auswahlmenü, mit dem Textfeld daneben, ist eine Liste aller dem System bekannten Schnittstellen zu den Sun LOMs. Wir gehen davon aus, dass eine Sun LOM immer per TCP/IP erreichbar ist und über Telnet gesteuert wird. Das Auswahlmenü ist also nur ein Alias für eine IP-Adresse. In das Feld daneben muss der Benutzer den TCP-Port eintragen, auf den sich der Powerman Server verbinden soll, wenn er eine Sun LOM bedienen möchte.

A.2.2.3 Charakteristik

Auf der Übersichtsseite wird der ungefähre momentane Stromverbrauch angezeigt. Damit der Powerman-Server den Stromverbrauch abschätzen kann, sind in der Datenbank Charakteristiken des Stromverbrauchs der unterschiedlichen Clients hinterlegt. Da die Charakteristik nur die Auslastung der CPU berücksichtigt sind im Moment nur zwei Charakteristiken im System hinterlegt. Eine Charakteristik für PCs und eine für Suns.



Mark.	Bedeutung	
1	A.2.2.1	Textfeld zur Bearbeitung der Systemnachricht
2	A.2.2.2	Managementkomponente zuweisen (nur priviligierte Benutzer)
3	A.2.2.3	Stromverbrauchscharakteristik zuordnen (nur priviligierte Benutzer)

Abbildung A.9: Einstellungen für einen einzelnen Client

A.2.3 Rechner löschen

Funktionen zum Löschen von Objekten aus dem System fragen immer zuerst den Benutzer, ob er sich sicher ist, dass das Objekt gelöscht werden soll und bietet dem Benutzer zwei Buttons an. Der Abbrechen-Button führt den Benutzer zurück zur vorherigen Seite, ohne dass Änderungen am System vorgenommen werden. Der Löschen-Button ist die Bestätigung zum Löschen des Objekts.

A.3 Grundbetriebszeiten

Abbildung A.10: übersicht Grundbetriebszeiten

Diese Seite bietet eine Übersicht der Grundbetriebszeiten, welche beliebig vielen Clients zugewiesen werden können. Außer dem Plan an sich kann man hier auch ablesen, wievielen Rechnern eine Grundbetriebszeit zugeordnet ist. Die Funktionen auf dieser Seite (Abbildung A.10) sind in der Tabelle A.3 aufgeführt.

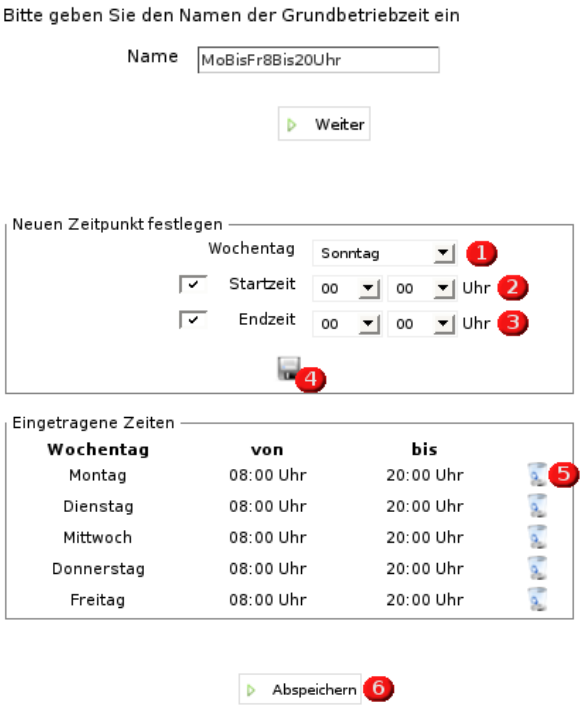
Tabelle A.5: Funktionen der Grundbetriebszeitenliste

Markierung	Symbol	Bedeutung
1		A.3.1 Grundbetriebszeit anlegen
2		A.3.2 Grundbetriebszeit bearbeiten
3		A.3.3 Grundbetriebszeit löschen

A.3.1 Grundbetriebszeit hinzufügen

Die Eingabemaske zum Hinzufügen von Grundbetriebszeiten und die Eingabemaske zum Bearbeiten einer Grundbetriebszeit sind identisch. Beim Hinzufügen sind das Namensfeld und die Liste der eingetragenen Zeiten leer. Da die Bedeutung der Felder an einem Beispiel leichter verständlich ist, wird die Eingabemaske in A.3.2 beschrieben.

A.3.2 Grundbetriebszeit bearbeiten



Markierung	Bedeutung
1	Wochentag
2	Zeitpunkt zu dem der Client hochgefahren werden soll
3	Zeitpunkt zu dem der Client heruntergefahren werden soll
4	Zeit eintragen
5	Zeit aus Liste löschen
6	Eingetragene Zeiten speichern

Abbildung A.11: Grundbetriebszeit bearbeiten

Abbildung A.11 zeigt die Seite zum Bearbeiten der Grundbetriebszeit MoBisFr8Bis20Uhr aus Abbildung A.10. Der Bearbeitungsvorgang ist auf zwei Seiten aufgeteilt. Zuerst wird dem Benutzer eine Eingabemaske zum Verändern des Namens der Grundbetriebszeit gezeigt (Abbildung A.11 links oben). Um zur zweiten

Seite zu gelangen muss der Benutzer auf den Weiter-Button klicken. Die zweite Seite (Abbildung A.11 links unten) bietet die Möglichkeit neue Betriebszeiten zur Grundbetriebszeit hinzuzufügen und beliebige vorhandene Betriebszeiten zu löschen. Ist man mit der Bearbeitung fertig, so benutzt man den Abspeichern-Button um die Liste der eingetragenen Zeiten zu speichern. Eine Betriebszeit muss nicht aus Start- und Stoppzeit bestehen. Entfernt man den Haken aus der Checkbox, so wird der Zeitpunkt nicht gespeichert. Dadurch ist es möglich Betriebszeiten ohne Start- bzw. Stoppzeit zu erstellen. Damit lassen sich Regeln definieren, die einen Rechner ausschalten, wenn er zu einer bestimmten Zeit an ist bzw. anschalten, wenn er aus ist. Die zweite Grundbetriebszeit aus Abbildung A.10 enthält ein Beispiel für eine Betriebszeit ohne Startzeitpunkt. In diesem Beispiel wird jeder Rechner, dem nur diese Grundbetriebszeit zugewiesen ist, jeden Donnerstag um 19:00 Uhr heruntergefahren, wird jedoch nie automatisch gestartet.

A.3.3 Grundbetriebszeit löschen

(siehe Kapitel A.2.3)

A.4 Benutzer

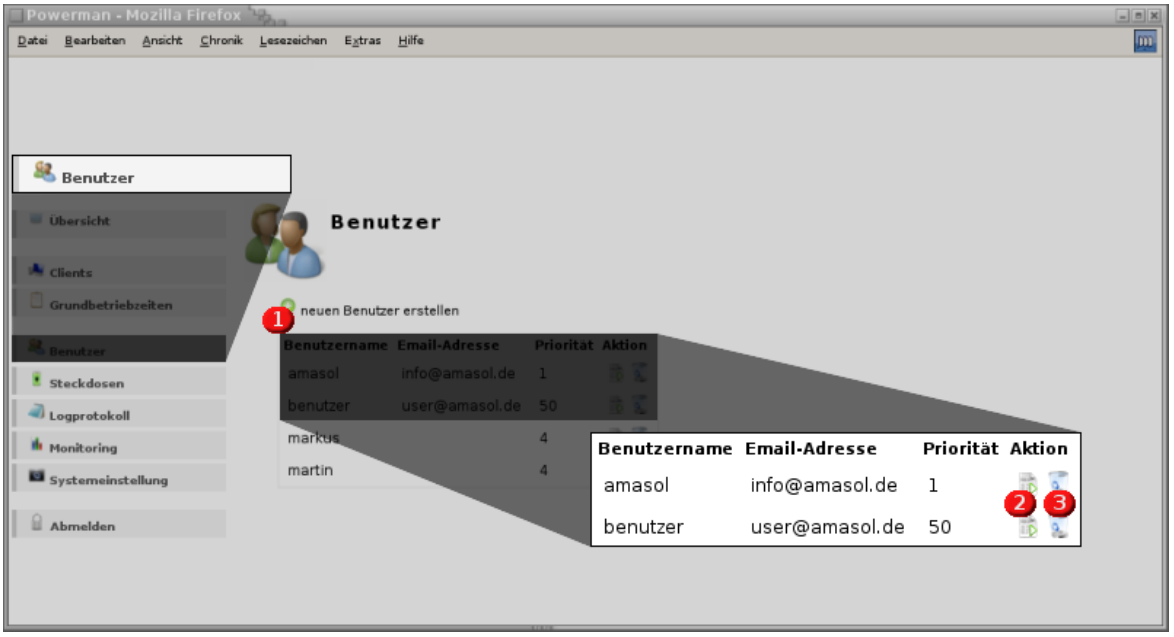
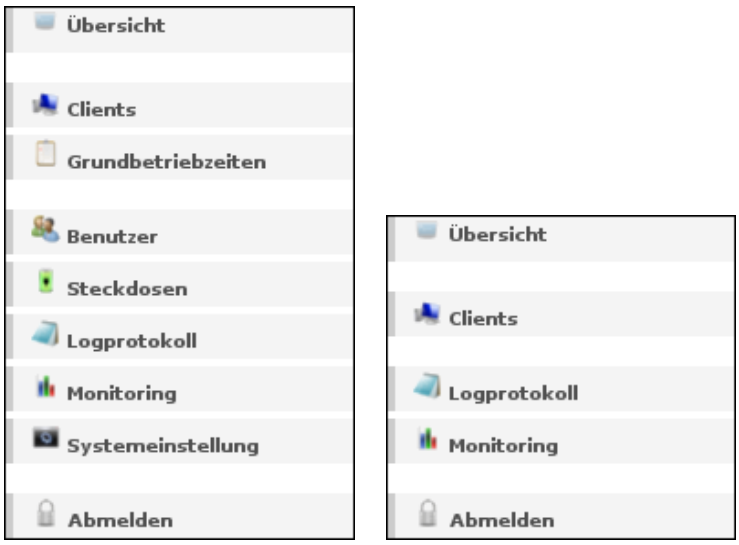


Abbildung A.12: Benutzerübersicht

Das Powerman-Frontend verwendet eine eigene Benutzerverwaltung zur Zugangssteuerung. Prinzipiell unterscheiden wir zwischen normalen Benutzern und Adminbenutzern. Während Adminbenutzern die volle hier beschriebene Mächtigkeit des Powerman Frontends zur Verfügung steht, sind die Möglichkeiten des normalen Benutzers sehr beschränkt. Dem normalen Benutzer stehen nur alle Möglichkeiten der Übersichtsseite (Kapitel A.1) und das Setzen der Systemnachricht der Clientseite (Kapitel A.2) zur Verfügung. Das Logprotokoll (Kapitel A.6) und Monitoring (Kapitel A.8) besitzen ohnehin keine Möglichkeiten für Einstellungen.

Jedem Benutzer ist eine Priorität zugeordnet. Die Position in der Hierarchie ist umgekehrt zum Zahlwert, d.h. je höher der Zahlwert der Priorität, desto weiter unten steht der Benutzer in der Benutzerhierarchie. Dadurch ist die höchste Priorität die ein Benutzer haben kann 0. Die Priorität hat zwei Funktionen. Zum Einen ist jeder Benutzer mit einer Priorität zwischen 0 und 10 ein Adminbenutzer und kann damit das gesamte Frontend nutzen, zum Anderen steuert die Priorität das überschreiben von Ausnahmen. Ein Benutzer kann eine Ausnahme nur dann überschreiben oder löschen, wenn seine Priorität mindestens der Priorität des Benutzer entspricht, der die Ausnahme gesetzt hat.



Menü der Adminbenutzer Menü der normalen Benutzer

Abbildung A.13: (Admin-)Benutzermenü

Tabelle A.6: Funktionen der Benutzerliste

Markierung	Symbol	Bedeutung
1		A.4.1 Benutzer hinzufügen
2		A.4.2 Benutzer bearbeiten
3		A.4.3 Benutzer löschen

A.4.1 Benutzer hinzufügen

Analog zu Grundbetriebszeiten, sind die Eingabemasken zum Hinzufügen und zum Bearbeiten von Benutzern identisch und werden in Kapitel A.4.2 erläutert. Die Bedeutung der Felder geht direkt aus der Beschriftung hervor. Die Email-Adresse ist das einzige Feld, welches nicht ausgefüllt werden muss. Sind alle relevanten Felder ausgefüllt, so wird der neue Benutzer durch einen Klick auf den Speichern-Button angelegt.

A.4.2 Benutzer bearbeiten

Das Beispiel in Abbildung A.14 zeigt den Benutzer „amasol“, für den eine Email-Adresse eingetragen ist und dessen Priorität 1 ist. Beim Bearbeiten wird das aktuelle Passwort nicht angezeigt, auch nicht als Kette von *. Benutzername, Email-Adresse und Priorität sind immer änderbar. Die Passwortfelder werden vom System nur berücksichtigt, wenn im Feld Passwort etwas eingetragen wurde. In diesem Fall muss der Inhalt beider Felder gleich und nicht leer sein. Auch der Benutzername ist jederzeit änderbar. Das funktioniert, da Änderungen erst beim nächsten Einloggen des betroffenen Benutzers berücksichtigt werden, d.h. auch wenn der Benutzername von „amasol“ nach „Amasol“ geändert wird, so können alle Benutzer, die im Moment als „amasol“ eingeloggt sind problemlos weiterarbeiten. Erst beim nächsten Login muss als Benutzername „Amasol“ geschrieben werden.

A.4.3 Benutzer löschen

(siehe A.2.3)

Markierung	Bedeutung
1	Angabe der Email-Adresse ist optional
2	Die Priorität ist absteigend, $0 \mathrel{\hat{=}}$ höchster Priorität, Prioritäten $0 - 10$ sind Adminbenutzer
3	änderungen werden erst mit dem nächsten Einloggen des betroffenen Benutzers spürbar

Abbildung A.14: Benutzer bearbeiten

A.5 Steckdosen

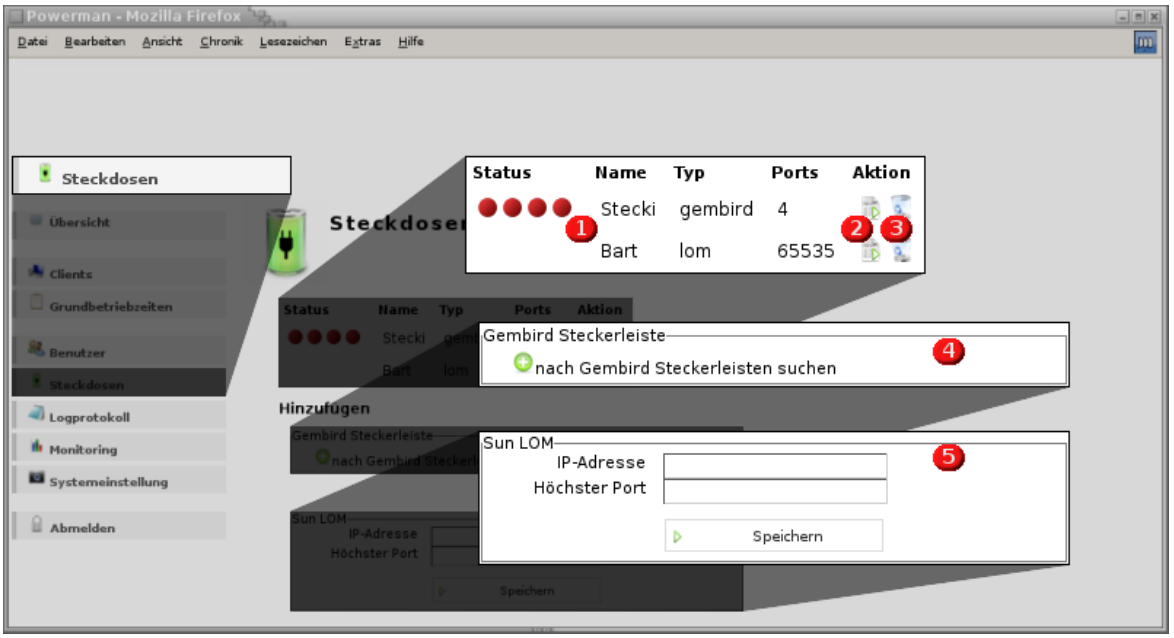


Abbildung A.15: Steckdosenübersicht und hinzufügen

Hinter dem Menüpunkt „Steckdosen“ versteckt sich die Verwaltung der Mangementkomponenten. Jede Art von Managementkomponente bringt ihre eigene Eingabemaske um Komponenten zum System hinzuzufügen mit und da dies im Moment genau zwei sind, platzieren wir diese gleich auf der Übersichtsseite. Entfernt man eine Komponente aus dem System, so werden automatisch die Zuweisungen dieser Komponente aus dem System entfernt. Die Funktionen in der Liste aller dem System bekannten Komponenten unterscheidet zwischen Gembird Steckerleisten und Sun LOMs. Bei Gembird Steckerleisten wird der Zustand (an oder aus) jeder einzelnen Steckdose durch eine grüne, bzw. rote Kugel dargestellt (Abbildung A.15 Markierung 1). Eine Kugel entspricht einer Steckdose, von 1 bis 4, von links nach rechts, entsprechend der aufgedruckten Nummerierung. Durch Klicken auf eine der Kugeln wird der Zustand dieser Dose verändert (ein- bzw. ausgeschalten). Da das Ausschalten von Steckdosen durchaus kritisch ist, muss der Benutzer das Abschalten einer Steckdose erst bestätigen, während das Einschalten einer Steckdose, ohne Aufforderung zur Bestätigung, direkt ausgeführt wird.

Die Spalte „Ports“ in der Übersicht hat ebenfalls eine spezielle Bedeutung. Bei Gembird Steckerleisten ist es schlicht die Anzahl der Steckdosen auf der Leiste. Bei Sun LOMs ist es der höchste erlaubte Port, d.h. wenn man beim Hinzufügen einer SunLOM als höchsten Port 22 einträgt, so kann bei der Zuweisung zu Rechnern

Tabelle A.7: Steckdosenübersicht

Markierung	Bedeutung	
1		Anzeige und Steuerung von Gembird Steckdosen
2	A.5.1	Bezeichnung ändern
3	A.5.2	Komponente löschen
4	A.5.3	Den USB-Bus nach Gembird Steckerleisten durchsuchen
5	A.5.4	Eine Sun LOM zum System hinzufügen

einer der TCP-Ports 0 bis 22 benutzt werden.

A.5.1 Komponente bearbeiten

Die Managementkomponenten sind tief im System verwurzelt und für das Arbeiten des Systems überaus kritisch. Deswegen ist es nur möglich die Bezeichnung der Komponente im System zu verändern. Die Eingabemaske dafür ist ähnlich der Abbildung oben links in Abbildung A.11.

A.5.2 Komponente löschen

(siehe A.2.3)

A.5.3 Gembird hinzufügen

Die Verwaltung von Gembird Steckerleisten ist für das System recht aufwändig. Dementsprechend wenig Einfluss kann der Benutzer darauf nehmen. Die Funktion „nach Gembird Steckerleisten suchen“ ist sehr mächtig. Sie durchsucht den USB-Bus nach Gembird Steckerleisten und fügt diese automatisch dem System hinzu. Jede neu erkannte Gembird Steckerleiste erhält den Namen „NeueSteckerleisteN“, wobei N bei 0 beginnt und für jede gefundene Leiste um eins erhöht wird. Die Reihenfolge in der Steckerleisten erkannt werden hängt vom Betriebssystem und ggf. der Reihenfolge, in der die Steckerleisten an das System angeschlossen wurden ab. Die Anzahl der Steckdosen auf der Steckerleiste wird automatisch bestimmt. Beim Durchsuchen des USB-Busses berücksichtigt das System bereits bekannte Steckerleisten und fügt diese nicht erneut dem System hinzu. Andererseits löscht das System auch sofort Steckerleisten, die es nicht mehr am USB-Bus findet.

A.5.4 Sun LOM hinzufügen

Das Hinzufügen von Sun LOMs erfolgt im Gegensatz zu Gembird Steckerleisten nicht automatisch. Hier muss per Hand die IP-Adresse und der höchste erlaubte Port eingeben werden. Mit dem Speichern-Button fügt man die neue Komponente dem System hinzu. Nun kann diese IP-Adresse einem Client als Managementkomponente zugewiesen werden.

A.6 Logprotokoll

Im Logprotokoll werden effektive Änderungen am System zusammen mit der Uhrzeit und dem Benutzer, der die Änderung durchgeführt hat, gespeichert. Die Liste ist nach dem Zeitpunkt absteigend sortiert, d.h. die neuesten Nachrichten stehen oben. „www“„system“ und „cli“ sind interne Benutzer, ohne Loginmöglichkeit.

Abbildung A.16: Ansicht Logprotokoll

Tabelle A.8: Funktionen aus Abbildung A.16

Markierung	Bedeutung
1	Alle Einträge die älter als einen Monat sind löschen

Nachrichten vom Benutzer www sind Meldungen, die das Frontend produziert. Dem Benutzer cli sind alle Meldungen, die das Command Line Interface (CLI) produziert zugeordnet und alle Nachrichten, die die Powermanlogik produziert sind dem Benutzer system zugeordnet. Die Nachrichten werden durch den Löschen-Button nicht endgültig gelöscht. Die zu löschenden Nachrichten werden in einer HTML-Datei gespeichert, so dass sie auch nach dem Löschen bei Bedarf noch eingesehen werden können.

A.7 Monitoring

Das Monitoring ermöglicht das Betrachten der Leistungsdaten von Rechnern über die Zeit. Dazu werden die Messdaten, die der Powerman-Server sammelt, in Graphen visualisiert. Tabelle A.9 erklärt die Funktionen der Seite.

A.8 Systemeinstellung

Einige Aspekte des Systems sind an einfache Zahlen gebunden. Diese beeinflussen die Performance und Bedienbarkeit des gesamten Systems. Aus Gründen der Wartbarkeit und Fehleranalyse speichert das Powerman System diese einfachen, jedoch kritischen Einstellungen zentral in der Datenbank. Hier erhält der Benutzer die Möglichkeit Einfluss auf das Verhalten des Systems auszuüben. Das System kann nur funktionieren wenn alle Attribute des Systems zusammen passen. Die verschiedenen Datentypen werden in Kapitel A.8.1 eingeführt und die Bedeutung der Attribute in Kapitel A.8.2.

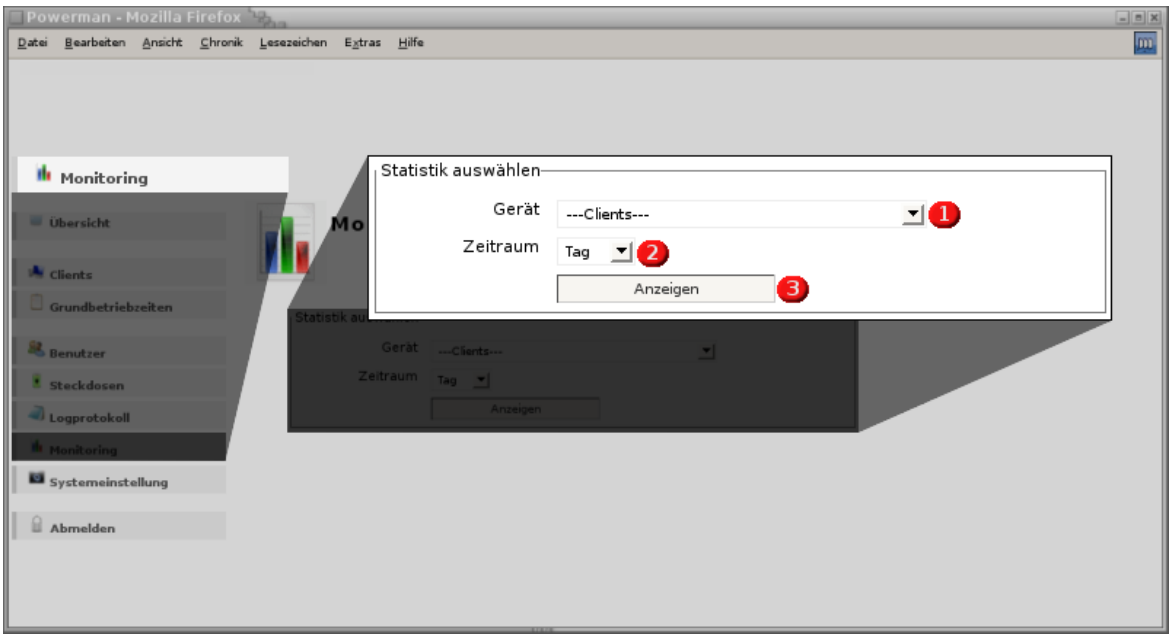


Abbildung A.17: Auswahlbildschirm für das Monitoring

Tabelle A.9: Funktionen aus Abbildung A.17

Markierung	Bedeutung
1	Auswahl zu welchem Rechner die Graphen erstellt werden sollen. Außer den Clients stehen hier auch evtl. angeschlossene Sensoren zur Verfügung
2	Zeitraum, den der Graph abdecken soll
3	Löst das Generieren der Graphen aus. Bei einem Zeitraum über einer Woche kann dies, je nach Auslastung des Servers, einige Minuten in Anspruch nehmen

A.8.1 Datentypen

Die Attribute auf dieser Seite sind einfache Werte aus einem bestimmten Definitionsbereich. Zur Integritätsbestimmung kennt das Powerman System den Definitionsbereich eines jeden Attributs, sodass eine jede änderung eines Werts das System syntaktisch korrekt hält.

A.8.1.1 Boolean

Prinzipiell kennen Felder des Datentyps Boolean nur zwei Zustände, die im Allgemeinen als „Wahr“ und „Falsch“ bzw „an“ und „aus“ oder auch „aktiviert“ und „deaktiviert“ bezeichnet werden. Unter Umständen

Tabelle A.10: Funktionen aus Abbildung A.18

Markierung	Bedeutung
1	Wert des Attributs verändern

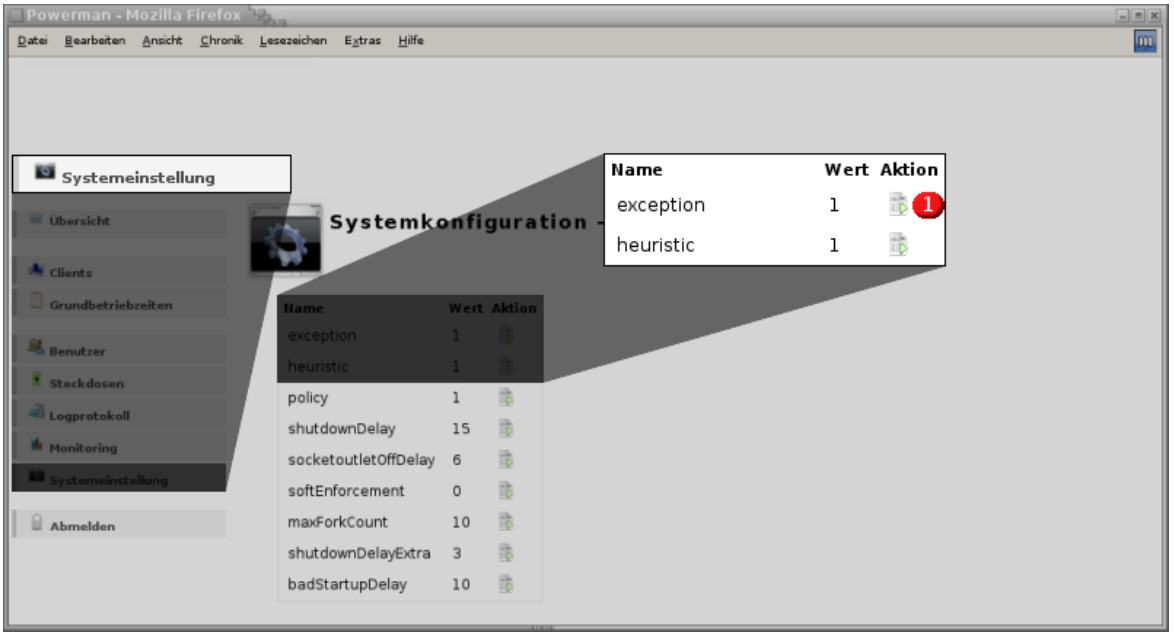


Abbildung A.18: übersicht Systemkonfiguration

kann die Eingabemaske von der Norm abweichen. Das Frontend akzeptiert die Eingaben „true“, „TRUE“ und „1“ als „Wahr“ bzw. „an“ und die Eingaben „false“, „FALSE“ und „0“ als „Falsch“ bzw.„aus“ .

A.8.1.2 setN

SetN beschreibt die Menge der Natürlichen Zahlen, einschließlich der 0. Das Frontend wird für diesen Datentyp ein einfaches Textfeld zur Verfügung stellen.

A.8.2 Attribute

Dies sind die Attribute, die der Benutzer mit Hilfe des Frontends verändern kann. Manche von ihnen unterliegen strengeren Vorschriften, als es ihr Datentyp vermuten lassen möchte.

A.8.2.1 exception — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Ausnahmen für Clients betrifft. Ist dieses Attribut aktiviert, so werden die definierten Ausnahmen und das Quickbooking beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert so werden Ausnahmen nicht berücksichtigt, jedoch nicht aus dem System gelöscht.

A.8.2.2 heuristic — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Heuristiken für Clients betrifft. Ist dieses Attribut aktiviert, so werden die definierten Heuristiken beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert so werden Heuristiken nicht berücksichtigt, jedoch nicht aus dem System gelöscht.

A.8.2.3 policy — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Grundbetriebszeiten für Clients betrifft. Ist dieses Attribut aktiviert , so werden die zugewiesenen Grundbetriebszeiten beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert, so werden Grundbetriebszeiten nicht berücksichtigt, jedoch werden Zuweisungen nicht gelöscht.

A.8.2.4 shutdownDelay — setN

Ein Client erhält niemals den Befehl sich sofort abzuschalten, sondern erst nach einer bestimmten Vorwarnzeit. Dieses Attribut gibt die Vorwarnzeit in Minuten an und muss ein Vielfaches von fünf sein.

A.8.2.5 shutdownDelayExtra — setN

Heuristiken haben alle fünf Minuten die Möglichkeit, einen Herunterfahrenvorgang abzubrechen. Das Attribut shutdownDelayExtra ist eine zusätzliche Vorwarnzeit, damit die Heuristiken noch ein letztes Mal überprüft werden, bevor ein Client heruntergefahren wird.

A.8.2.6 socketoutletOffDelay — setN

Manche Managementkomponenten bzw. Steckdosen müssen abgeschaltet werden. Der Powerman veranlasst daher socketoutletOffDelay Minuten nach dem effektiven Herunterfahren die dem Client zugewiesene Mangementkomponente auf „aus“ zu schalten.

A.8.2.7 softEnforcement — Boolean

Ein Schalter zur Steuerung der Powerman Logik. Ausnahmen und Grundbetriebszeiten sind zwei getrennte Kontrollschichten [2]. Das bedeutet, dass Ausnahmen Grundbetriebszeiten überschreiben, aber nicht umgekehrt. Das Ende einer Ausnahme ist automatisch ein Zeitpunkt, an dem ein Client heruntergefahren werden soll. Fällt die Verzögerungszeit in eine Grundbetriebszeit, also zwischen Start- und Stoppzeitpunkt, wird ein Rechner heruntergefahren, auch wenn die Grundbetriebszeit das zu diesem Zeitpunkt nicht vorsieht. Ist softEnforcement aktiviert kann die Logik die Entscheidung treffen, den Client in dieser Situation an zu lassen, jedoch nur mit dem Nebeneffekt, dass die Logik einen Rechner immer, nicht nur zur Startzeit, anschaltet, sobald es eine Ausnahme oder Grundbetriebszeit gibt, die dieses vorsieht.

A.8.2.8 maxForkCount — setN

Da der Server viel Zeit mit warten verbringt, eignet sich das Sammeln von Informationen sehr gut für Paralellisierung. Dabei erzeugt der Serverprozess Kopien von sich selbst, um mehrere Rechner mehr oder weniger gleichzeitig abfragen zu können. Hier wird die maximale Anzahl der gleichzeitig vorhandenen Kopien des Serverprozesses nach oben beschränkt, da Prozesskopien relativ hohe Systemlast mit sich bringen.

A.8.2.9 badStartupDelay — setN

Tritt nun auf dem Client ein Fehler auf, so dass es nicht zum Start der Clientsoftware kommt, kann das System nicht richtig funktionieren. Hat sich ein Client nach badStartupDelay Minuten nicht am System angemeldet, so wird der Client intern wieder auf „aus“

[2] vgl. Kapitel A.2

A.8.2.10 workingHours — setN

Für die Anzeige über die gesparte Energie wird ein Vergleichswert zum aktuellen Stromverbrauch benötigt. Als Vergleichswert nehmen wir an, dass alle Rechner für mehrere Stunden pro Tag durchgehend an sind und dabei minimal ausgelastet sind. Diese Variable ist die Anzahl der Betriebsstunden pro Tag, an 5 Tagen der Woche.

A.8.2.11 kWhCost — setN

Der Preis für eine Kilowattstunde in Cent. Dieser Wert bei der Ersparnisanzeige zum Umrechnen von Kilowattstunden in Euro benutzt.

A.9 Powerman Command Line Interface

Für die geskriptete Benutzung des Powerman, oder zur schnellen Bedienung, ohne Webfrontend, existiert auch ein Kommandozeilenwerkzeug (PCLI). Seine Funktionen beschränken sich auf das Ein- und Ausschalten, sowie das neustarten von Computern und das Abbrechen dieser Vorgänge.

Listing A.1: Command Line Interface Usage

```
# powerman --help

USAGE: /usr/local/bin/powerman [--user=amasol] [--pass=secret] [--c=client] [
    start|stop|powercycle|abort]
```

Auch bei Benutzung des PCLI muss der Benutzer sich mit Benutzername(–user=) und Passwort(–pass=) authentifizieren. Ist der Benutzer authentifiziert, so so wird die Aktion(start—stop—powercycle—abort) auf den Client(–c=) angewendet. Das Anwenden von Aktionen auf eine Mehrzahl von Clients wird nicht unterstützt. Es sind nicht zwingend alle vier Angaben erforderlich. Angaben, die nicht beim Aufruf mit angegeben wurden, werden einzeln vom PCLI abgefragt.

B Dokumentation für Administratoren

B.1 Installierung des Powerman Servers

Der Powerman Server besteht aus einer Vielzahl von Softwarekomponenten, die für die Arbeit des Systems benötigt werden. Dieses Kapitel soll eine Anleitung zur Installation eines Powerman-Servers beinhalten. Tabelle B.1 ist eine Liste der Abhängigkeiten des Powerman Servers. Diese Pakete müssen alle installiert sein, damit der Powerman funktionieren kann. Alle Pakete die allgemein benötigt werden, müssen auf jeden Fall

Paket	**Funktion**	**benötigt für**
Apache	Webserver	Frontend
Postgresql	Datenbank	allgemein
PHP mit cli	Basispaket	allgemein
PHP-postgres	PHP Postgresqlerweiterung	allgemein
PHP-gd	PHP Grafikbibliothek	Frontend
PHP-socket	PHP Socketerweiterung	Sun LOM
PHP-snmp	PHP SNMP-Erweiterung	allgemein
PHP-pcntl	PHP Prozess-Erweiterung	allgemein
PHP-ncurses	PHP NCurses-Erweiterung	cli
PEAR-DB	Datenbankabstraktionsbiliothek für PHP	allgemein
digitemp	Temperatur auslesen	1-Wire Sensoren
sispmctl	Steckdosensteuerung	Gembird Steckerleisten
cron	Daemon	allgemein
Net-SNMP	Trap-Daemon	allgemein
Dateisystem usbfs	USB Information	Gembird Steckerleisten

Tabelle B.1: Vom Powerman Server benötigte Pakete

installiert werden, die restlichen Pakete nur, wenn man die entsprechende Funktion auch nutzen möchte.

Das Programm sispmctl benötigt einen speziellen Patch für das Powerman-System. Deshalb muss diese Software selnst übersetzt werden und kann nicht aus einem Paketverwaltungsprogramm heraus installiert werden.

Als alternative für den Apache Webserver eignet sich jeder Webserver, der PHP-Skripe aufrufen kann. Da das System über eine Benutzerautentifizierung verfügt, bietet es sich von der Sicherheit her an, den Zugriff auf das Frontend auf https zu beschränken. Als alternative fpr den Net-SNMP Trap-Daemon eignet sich jeder Trap-Daemon, der externe Skripte aufrufen kann.

Datei bzw. Ordner	Verwendung
/var/powerman/cli	Enthält Powerman Command Line Interface
/var/powerman/www	Enthält Powerman Webinterface
/var/powerman/common/config/config.php	Powerman Konfigurationsdatei
/var/powerman/misc/sispmctl/sispmctl_powerman.patch	Patch für sispmctl Version 2.4a
/var/powerman/misc/snmp/conf/snmptrapd.conf	Konfigurationsdatei für Net-SNMP Trapdaemon
/var/powerman/misc/system/crontab	Neuer Eintrag für /etc/crontab
/var/powerman/misc/system/fstab	Neuer Eintrag für /etc/fstab
/var/powerman/misc/setup.sql	Initialisierungsskript für Postgresql

Tabelle B.2: Serverinstallationsrelevante Dateien und Ordner

B.1.1 Powerman Archiv

Das Powerman Archiv muss nach /var/powerman entpackt werden. Es enthält alle Dateien, die in irgendeiner Form für den Powerman relevant sind, inklusive der Powermanpakete für Client Computer. Die für die Installation wichtigen Dateien und Ordner sind in Tabelle B.2 aufgelistet.

Listing B.1: Powerman Archiv entpacken

```
1 # mkdir -p /var/powerman
2 # tar -xzf powerman.tgz -C /var/powerman
```

B.1.2 PHP einrichten

Bei manchen Distributionen ist das php-cli ebenso wie alle anderen Zusatzmodule je ein eigenständiges Paket und wird standardmäßig nicht mitinstalliert. Der Powerman benötigt es jedoch, für das Powerman-CLI und für den SNMP-Trapdaemon. Das Einrichten von PHP mit Standardwerten ist für den Powerman ausreichend. Bei einer speziellen Konfiguration von PHP muss darauf geachtet werden, dass

- Outputbuffer erlaubt sind
- SNMP v2 verwendet wird
- TCP für SNMP erlaubt ist
- die PNG-Unterstützung der GD-Bibliothek zur Verfügung steht

Des Weiteren benutzt der Powerman die PEAR DB-Bibliothek für Datenbankzugriffe. Diese muss ebenfalls installiert sein und sich im standard Includepath von PHP befinden.

B.1.3 Webserver einrichten

Für das Webinterface des Powermans muss zunächst ein Webserver konfiguriert werden. Der Webserver muss Dateien mit der Endung .php mit PHP assoziieren können. Die Startdatei des Frontends ist index.php und normalerweise im Verwezichnis /var/powerman/www zu finden. Diese Datei bindet Dateien aus ../common/ und ../ccu/ ein. Deswegen muss für das Frontend die Ordnerstruktur aus /var/powerman beibehalten oder zu mindest simuliert werden und die rechte dementsprechend gesetzt sein.

B.1.4 Postgresql einrichten

Die Datei /var/powerman/misc/setup.sql enthält das Datenbanklayout für den Powerman. Es besteht aus Tabellen-, Funktions-, Trigger- und Domänendefinitionen, zusammen mit ein paar Initialdaten. Die normale Power-

man Installation geht davon aus, dass für den Powerman eine extra Datenbank angelegt wird, deswegen benutzt der Powerman das Schema public. Möchte man die Powerman Datenbank in ein anderes Schema bannen, so muss man /var/powerman/misc/setup.sql anpassen, indem man in der ersten Zeile den Suchpfad von public auf den neuen Namen anpasst. Außerdem muss man in der Datei /var/powerman/common/sqlconnect.sql noch den Kommentar von der Zeile welche den Befehl „ SET SEARCH PATH TO “entfernen und public durch den neuen Namen ersetzen (vgl. Listing B.2).

Listing B.2: Auskommentierte Zeile zum Auswählen des Schemas aus sqlconnect.php

```
1 // Schema in Postgresql
2 // $_dbHandle->query("SET SEARCH PATH TO \"public\"");
```

B.1.5 Digitemp installieren

Das Programm Digitemp wird so, wie es ausgeliefert wird, vom Powerman benutzt. Bei der Installation ist lediglich darauf zu achten, dass der Powerman es aufrufen kann. Dazu muss es im Pfad, der in der config.php als BINDIR gespeichert ist liegen und das Binary muss digitemp heißen.

B.1.6 Sispmctl installieren

Im Gegensatz zu Digitemp kann Sispmctl nicht als vorkompiliertes Paket installiert werden, sondern muss aus den Quellen übersetzt werden. Der Grund hierfür ist, dass es Probleme bei der Verwaltung von Gembird Steckerleisten gibt, da diese kein Merkmal haben, welches sie eindeutig identifiziert. Um das Problem zu lösen, werden die Steckerleisten über ihre Position am USB-Bus identifiziert. Sispmctl bietet diese Methode der Identifizierung nicht, dafür ist der kleine Patch aus /var/powerman/misc/sispmctl nötig. Dieser erweitert die src/main.c um die Möglichkeit eine Steckdose per Bus- und Devicenummer zu identifizieren.

Listing B.3: sispmctl kompilieren und installieren

```
1 # cd
2 # wget http://mesh.dl.sourceforge.net/sourceforge/sispmctl/sispmctl-2.4a.tar.gz
3 # cd /usr/local/src
4 # tar -xzf ~/sispmctl-2.4a.tar.gz
5 # cd sispmctl-2.4a
6 # patch src/main.c /var/powerman/misc/sispmctl/sispmctl_powerman.patch
7 # ./configure --enable-webless
8 # make
9 # cp src/sispmctl /usr/local/bin/
```

B.1.7 Powerman Command Line Interface installieren

Das PCLI ist einsatzbereit, wenn es das Webfrontend auch ist. Der Ordner /var/powerman/cli enthält lediglich die Datei cli.php. Zur einfacheren Benutzung sollte das PCLI im Pfad liegen. Dazu erstellen wir von Hand einen symbolischen Link in /usr/local/bin.

Listing B.4: symbolischen Link erstellen

```
1 # ln -s /var/powerman/cli/cli.php /usr/local/bin/powerman
```

B.1.8 Powerman konfigurieren

Das Frontend, das PCLI und die Steuerlogik des Powerman benutzen alle die selbe Konfigurationsdatei. Sie befindet sich in /var/powerman/common/config. Die Datenbankeinstellungen sind als Array $_confDB gespeichert. Abgesehen davon sind alle Einstellungen als Konstanten definiert.Tabelle B.3 ist eine Liste aller

Möglichkeiten der /var/powerman/common/config/config.php. Die setup.sql welche zum initialisieren der Datenbank benutzt wird (vgl Kapitel B.1.4), richtet außerdem auch den Adminbenutzer *amasol* mit Passwort *amasol* ein. Einige Einstellungen können online im System verändert werden. Für eine kurze Beschreibung der Systemeinstellungen siehe Benutzerdokumentation.

B.1.9 SNMP-Trapdaemon einrichten

Der Powerman Server sucht niemals nach Clients. Immer wenn ein Client startet schickt er einen SNMP-Trap an den Server um sich an diesem anzumelden. Deswegen muss auf dem Powerman Server ein SNMP-Trapdaemon auf dem TCP-Port 162 lauschen. Für den Trap benutzen die Clients die Community pow und die oid des Traps ist 1.3.6.1.4.1.19518.4123.1.1.0. Dieser Trap muss /var/powerman/ccu/trap.php mit der IP-Adresse des Trapsenders als Argument ausführen. Der Net-SNMP kann dieses nicht direkt, sodass wir ein Wrapper-Skript /var/powermam/ccu/trap dafür benutzen. Soll der Net-SNMP verwendet werden und ist dieser noch unkonfiguriert, kann man /var/powerman/misc/snmp/conf/snmptrapd.conf nach /etc/snmp/ kopieren um den Trapdaemon zu konfigurieren. Soll der Powerman zu einer bestehenden Konfiguration hinzugefügt werden, so sind die beiden Zeilen aus Listing B.5 zur momentanen /etc/snmp/snmptrapd.conf hinzuzufügen.

Listing B.5: sispmctl kompilieren und installieren

```
1 authCommunity execute pow
2 traphandle 1.3.6.1.4.1.19518.4123.1.1.0 /var/powerman/ccu/trap
```

B.1.10 Cron einrichten

Die Powerman-Logik ist kein eigenständiger Dienst, sondern lediglich ein php-Skript. Das regelmäßige Ausführen des Skriptes übernimmt der cron-Daemon für uns. Die Powermanlogik /var/powerman/ccu/papa.php muss alle 5 Minuten ausgeführt werden. Dazu muss die Zeile aus /var/powerman/misc/system/crontab (siehe Listing B.6) zu /etc/crontab hinzugefügt werden.

Listing B.6: Powermanlogik 5 Minuteneintrag in /etc/crontab

```
1 0-59/5 * * * *  root    php /var/powerman/ccu/papa.php
```

B.1.11 Tmpfs einrichten (optional)

Um die das Webfrontend zu beschleunigen kann man das tmpfs-Dateisystem benutzen. Dazu muss man ein tmpfs in /var/powerman/www/tmp einhängen. Dadurch folgen aus Zugriffen auf dieses Verzeichnis keine Festplattenzugriffe, sondern die temporären Dateien werden im RAM abgelegt, was den Geschwindigkeitsvorteil bringt. Zur Vereinfachung enthält /var/powerman/misc/system/fstab bzw. Listing B.7 einen passenden Eintrag für /etc/fstab.

Listing B.7: Powermanlogik 5 Minuteneintrag in /etc/crontab

```
1 tmpfs   /var/powerman/www/tmp   tmpfs   size=256M       0 0
```

B.1.12 Installationsabschluss

Um die Installation abzuschließen müssen nun noch die konfigurierten Dienste gestartet werden. Der Powerman Server ist nun einsatzbereit.

B.2 Installieren des Powerman Clients

Die Powerman Clientsoftware liegt sowohl für Windows als auch Solaris in /var/powerman/misc/powerman-Deploy.

B.2.1 Installation unter Solaris

Der Net-SNMP gehört unter Solaris nicht zu den Standardpaketen. Deswegen ist die Installation in zwei Stufen aufgeteilt. Zuerst soll der Net-SNMP installiert werden und danach die Powermanerweiterung und die init-Skripte. Dazu muss zunächst das Tar-Archiv nach /tmp entpackt werden, denn nach der Installtion wird dessen Inhalt nicht mehr benötigt.

B.2.1.1 Net-SNMP installieren

Um den Net-SNMP zu installieren muss man das Skript install_netsnmp.sh aus /tmp/powermaninstaller ausführen. Dieses benutzt pkg-get um Net-SNMP zu installieren. Für den Fall, dass pkg-get nicht auf dem Zielsystem installiert ist, beinhaltet der Installer das pkg-get Binary. Die Installation erfordert ggf. Benutzerinteraktion, wenn Ordner angelegt werden sollen, die bisher noch nicht existieren. Als Version des Net-SNMP soll mindestens 5.3.1 benutzt werden.

B.2.1.2 Powermanerweiterung installieren

Die Datei powermaninstaller/install_powerman.sh ist analog zum Installieren des Net-SNMP das Skript zum Installieren der Powerman-Erweiterung. Bevor man dieses Ausführt muss sichergestellt sein, dass die Binaries des Net-SNMP im Pfad enthalten sind. Dabei kommt es auf die Dateien net-snmp-config, snmpd und snmptrap an. Sind diese Dateien nicht im Pfad zu finden, so kann install_powerman.sh nicht korrekt arbeiten. Das Skript installiert sich in das Verzeichnis, in dem der Net-SNMP seine persistenten Daten speichert (normalerweise /var/net-snmp). Dort legt es die Ordner powerman/lib und powerman/conf an, kopiert seine Dateien hinein und erstellt die Datei /etc/init.d/powerman. Zum Schluss fügt das Skript den Powerman noch zum Defaultrunlevel als S91powerman hinzu. Gestartet wird der Dienst mittels /etc/init.d/powerman start.

Listing B.8: Powerman auf Solaris installieren und starten

```
1 # gunzip powerman_installer.tgz | tar -xf - -C /tmp
2 # cd /tmp/powermaninstaller
3 # ./install_netsnmp.sh
4 [...Net-SNMP nach /usr/local/ installieren lassen...]
5 # export PATH=/usr/local/sbin:/usr/local/bin:$PATH
6 # ./install_powerman.sh
7 # /etc/init.d/powerman start
```

B.2.2 Installation unter Windows

Die Installation geschieht mittels mitgeliefertem Installationsskript und kann entweder durch Doppeklick auf die Datei „install.vbs“ erfolgen oder durch Eingabe von „wscript install.vbs“ von der Kommandozeile aus. Das Skript fragt nur noch nach dem Installationspfad (standardmäßig c:\powerman), der Rest geschieht automatisch. Wichtig ist, dass das Skript mit vollen Administrator-Rechten gestartet wird, da einige Einstellungen vorgenommen werden müssen.

B.3 Logdateien

Die Benutzer haben im Frontend die Möglichkeit das Logprotokoll zu löschen. Wird diese Funktion benutzt, so verschwinden die Daten nicht endgültig. Bevor sie gelöscht werden, erstellt das Frontend aus den Daten noch eine einfache HTML-Datei, die eine Tabelle mit den Spalten „Timestamp“ , „Nachricht“ und „Benutzer“ beinhaltet. So kann man das Logprotokoll des Frontends kompakt und übersichtlich halten, es gibt jedoch außerhalb des Frontends jederzeit die Möglichkeit alte Logeinträge einzusehen. Die Dateinamen sind nach dem Schema $< Datum >$-logprotokoll-$< Uhrzeit >$.log aufgebaut. So entspricht eine Sortierung nach dem Dateinamen einer Sortierung nach dem Erstellungsdatum. In der Standardkonfiguration liegt in diesem Verzeichnis auch die Datei powerman.log, welche vom Serverprozess genutzt wird. Ein kurzes Beispiel der Ausgaben des Serverprozesses zeigt Listing B.9. Das Beispiel zeigt die Logeinträge des Serverprozesses bei einem aktiven Rechner.

Listing B.9: Beispiel Logeinträge eines Durchlaufs von papa.php

```
1  papa.php gestartet ...
2  2007-06-23 20:20:00 monitoring start
3                      eingeschaltet: 39
4  2007-06-23 20:20:00 kind 1 cpuload
5  2007-06-23 20:20:00 kind 1 memload
6  2007-06-23 20:20:00 kind 1 netload
7  2007-06-23 20:20:00 kind 1 temp
8  2007-06-23 20:20:00 kind 1 users
9  2007-06-23 20:20:00 kind 1 fertig
10 2007-06-23 20:20:00 alle kindprozesse haben terminiert
11 2007-06-23 20:20:00 monitoring fertig
12 2007-06-23 20:20:00 lese sensoren ... fertig
13 2007-06-23 20:20:00 [Hostname, ID, Aktion (leer, wenn keine)] {
14                        pc104 35
15                        pc105 42
16                        pc107 44
17                        sun1 38
18                        sun2 39
19                        sun3 40
20                        sun4 41
21                        sun5 34
22                        sun6 37
23 2007-06-23 20:20:00 } //fertig
24 2007-06-23 20:20:00 beginne detectStartupFailure ... fertig
25 2007-06-23 20:20:00 loesche abgelaufene Rules ... fertig
26 2007-06-23 20:20:00
27 papa.php fertig
```

B.4 Wartungsarbeiten

Da der Powerman eine Datenbank benutzt, gibt es relativ wenig, worum man sich regelmäßig kümmern muss.

- powerman/www/tmp überwachen
- Logdateien verwalten
- Prozessleichen entfernen

B.4.1 Tmp-Verzeichnis überwachen

Der Powerman löscht die Dateien, die er in /var/powerman/www/tmp erstellt, nicht automatisch. Volle Ordner beeinträchtigen die Laufzeit und so sollte der Ordner regelmäßig geleert werden. Da alle Dateien darin nur sehr

kurzzeitig benötigt werden, kann man ohne Probleme einfach den gesamten Ordnerinhalt löschen. Es besteht zwar die Möglichkeit, dass gerade benutzte Dateien gelöscht werden, dieses Problem lässt sich jedoch durch ein einfaches Neuladen der Seite (refresh) lösen. Diese Aufgabe ist besonders wichtig, wenn man sich für den optionalen Installationsschritt Tmpfs einrichten (Kapitel B.1.11) entschieden hat, da ein tmpfs in seiner größe viel beschränkter ist, als eine Festplatte.

B.4.2 Logdateien verwalten

Leert man das Logprotokoll des Frontends regelmäßig, so sammeln sich auch hier über die Zeit viele Dateien an, die unter Umständen sehr viel Speicherplatz in Anspruch nehmen können. Deswegen sollte man sich auch hier um das Entsorgen alter und unwichtiger Protokolle kümmern. Der Serverprozess generiert etwa ein Kilobyte an Logausgabe (bei nur einem aktiven Rechner) pro Durchlauf und somit über 256 Kilobyte pro Tag. Das heißt, dass diese Datei relativ schnell viel Speicherplatz benötigt und sollte daher regelmäßig gekürzt werden (z.B. mit logrotate).

B.4.3 Prozessleichen entfernen

Durch die Verwendung von PHP ergibt sich ein Problem bei der Prozesshandhabung. Wie es scheint, ist die Implementierung der POSIX-Funktionen nicht ganz sauber und so können Prozessleichen entstehen, welche die Systemleistung beeinträchtigen können. Der regelmäßige Aufruf von „killall php“ sorgt dafür, dass diese Leichen verschwinden.

B.4.4 Monitoringtabelle leeren

Die Antworten der Clients auf die regelmäßigen Anfragen werden in der Datenbank gespeichert. Das Frontend kann Überischtsgraphen für höchstens ein Jahr erstellen. Daten, die älter als ein Jahr sind, sind für den Powerman demnach nicht mehr relevant und können spätestens dann bedenkenlos gelöscht werden. Für diese Arbeit stellt das Frontend keine Funktion zur Verfügung, weshalb der Eingriff direkt per SQL an der Datenbank durchgeführt werden muss. Der Query aus Listing B.10 löscht alle Daten, die älter als ein Jahr sind.

Listing B.10: SQL-Query zum Löschen alter Monitoringdaten

```
1 DELETE FROM monitoring WHERE stamp < NOW() - interval '1 year';
```

B.5 Systemeinstellungen

Dieses Kapitel beschäftigt sich mit der Funktion „Systemeinstellungen“ des Frontends. Die dort vorhandenen Einstellungen sind sehr kritisch und werden deswegen hier in voller Länge erläutert.

Einige Aspekte des Systems sind an einfache Zahlen gebunden. Diese beeinflussen die Performance und Bedienbarkeit des gesamten Systems. Diese einfachen, jedoch grundlegenden Zahlen speichert das Powerman System zentral, aus Gründen der Wartbarkeit und Fehleranalyse. Hier erhält der Benutzer die Möglichkeit Einfluss auf das Verhalten des Systems zu üben. Das System kann nur funktionieren, wenn alle Attribute des Systems zusammen passen.

B.5.1 Datentypen

Die Attribute auf dieser Seite sind einfache Werte aus einem bestimmten Definitionsbereich. Zur Integritätsbestimmung kennt das Powerman System den Definitionsbereich eines jeden Attributs, sodass eine jede Änderung eines Werts das System syntaktisch korrekt hält. Alle hier vorgenommenen Einstellungen sind ohne

Einschränkung systemweit gültig. Sehr viele der Einstellungen betreffen direkt die Powerman Logik. Die Powerman Logik ist der Kern des Systems. Der Benutzer hat keinen direkten Zugang dazu. Hier entscheidet das System über die Aktionen, die für jedes einzelne System durchzuführen sind.

B.5.1.1 Boolean

Prinzipiell kennen Felder des Datentyps Boolean nur zwei Zustände, die im Allgemeinen als „Wahr“ und „Falsch“ bzw „an“ und „aus“ oder auch „aktiviert“ und „deaktiviert“ bezeichnet werden. Im Normalfall bietet einem das Frontend für jedes Datenfeld vom Typ Boolean ein Menü mit den beiden Möglichkeiten „an“ und „aus“, da diese Bezeichnungen im Kontext intuitiv verständlicher sind. Unter Umständen kann die Eingabemaske von der Norm abweichen. Das Frontend akzeptiert die Eingaben „ true“, „TRUE“und „1“als „Wahr“bzw. „an“und die Eingaben „false“, „FALSE“und „0“als „Falsch“bzw. „aus“.

B.5.1.2 setN

SetN beschreibt die Menge der natürlichen Zahlen, einschließlich der 0. Das Frontend wird für diesen Datentyp meistens ein einfaches Textfeld zur Verfügung stellen.

B.5.2 Attribute

Dies sind die Attribute, die der Benutzer mit Hilfe des Frontends verändern kann. Manche von ihnen unterliegen strengeren Vorschriften, als es ihr Datentyp vermuten lassen möchte.

B.5.2.1 exception — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Ausnahmen für Clients betrifft. Ist dieses Attribut aktiviert, so werden die definierten Ausnahmen und das Quickbooking beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert, so werden Ausnahmen nicht berücksichtigt, jedoch nicht aus dem System gelöscht.

B.5.2.2 heuristic — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Heuristiken für Clients betrifft. Ist dieses Attribut aktiviert, so werden die definierten Heuristiken beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert, so werden Heuristiken nicht berücksichtigt, jedoch nicht aus dem System gelöscht.

B.5.2.3 policy — Boolean

Ein Schalter zur Steuerung der Powerman Logik, der das Auswerten von Grundbetriebszeiten für Clients betrifft. Ist dieses Attribut aktiviert, so werden die zugewiesenen Grundbetriebszeiten beim Entscheidungsprozess berücksichtigt. Ist dieses Attribut deaktiviert, so werden Grundbetriebszeiten nicht berücksichtigt, jedoch werden Zuweisungen nicht gelöscht.

B.5.2.4 shutdownDelay — setN

Ein Client erhält niemals den Befehl sich sofort abzuschalten. Zum Einen muss eventuell am System arbeitenenden Benutzern die Chance gegeben werden, auf den Herunterfahrenbefehl zu reagieren, zum Anderen muss evtl. aktiven Heuristiken die Zeit gegeben werden, einen Herunterfahrenbefehl aufzuheben. Das führt dazu, dass der Powerman Clients im Normalfall zeitverzögert herunter fährt. Dieses Attribut gibt die Vorwarnzeit in Minuten an und muss ein vielfaches von fünf sein, sonst verliert der Powerman die Möglichkeit Rechner automatisch herunter zu fahren.

B.5.2.5 shutdownDelayExtra — setN

Heuristiken haben alle 5 Minuten die Möglichkeit einen Herunterfahrenvorgang abzubrechen. Da shutdownDelay ein vielfaches von 5 ist, fallen der Zeitpunkt des effektiven Herunterfahrens und der letzte Zeitpunkt, an dem für den Client Heuristiken ausgewertet werden können, zusammen. Somit nimmt man dem System die letzte Möglichkeit einfluss auf den Prozess zu nehmen. Um dieses Problem zu umgehen, schlägt das System nochmal zusätzliche Verzögerungszeit auf. Sinnvolle Werte für dieses Attribut sind alle zulässigen Zahlen kleiner 5. Setzt man dieses Attribut auf 0, so nimmt man dem System wieder die letzte Eingriffsmöglichkeit, dafür setzt das effektive Herunterfahren auch pünktlich ein. Ansonsten erhält man durch das Attribut shutdownDelayExtra eine leichte Unschärfe um den eingetragenen Wert. Der Zeitpunkt des effektiven Herutnerfahrens berrechnet sich also <Zeitpunkt aus der Grundbetriebszeit> + <shutdownDelayExtra>.

B.5.2.6 socketoutletOffDelay — setN

Sobald der Powerman Dienst auf den Clients abgeschaltet bzw. nicht mehr erreichbar ist, kann das System keine begründeten Aussagen mehr über den betroffenen Client treffen. Somit auch nicht mehr, ob das Herunterfahren erfolgreich war oder nicht. Ein System, welches sich nicht vollständig herunter fährt, kann sich nicht mehr automatisch erholen. Manche Managementkomponenten bzw. Steckdosen müssen abgeschaltet werden. Da wir keine Aussagen über den Fortschritt und Erfolg eines Herunterfahrenvorgangs treffen können, bleibt nur Zeit als Entscheidungsgrundlage. Der Powerman veranlasst daher socketoutletOffDelay Minuten nach dem Herunterfahren die dem Client zugewiesene Mangementkomponente auf „aus“ zu schalten. Der genaue Zeitpunkt dafür ist <Zeitpunkt aus der Grundbetriebszeit> + <shutdownDelayExtra> + <socketoutletOffDelay>. Der Client hat also genau socketoutletOffDelay Minuten Zeit zum Herunterfahren, danach schaltet ihn die Mangagementkomponente ab.

B.5.2.7 softEnforcement — Boolean

Ein Schalter zur Steuerung der Powerman Logik. Ausnahmen und Grundbetriebszeiten sind zwei getrennte Kontrollschichten. (vgl. Kapitel A.2) Das bedeutet, dass Ausnahmen Grundbetriebszeiten überschreiben, aber nicht umgekehrt. Das Ende einer Ausnahme ist automatisch ein Zeitpunkt, an dem ein Client heruntergefahren werden soll. Die Powerman Logik hat das Problem, dass diese Entscheidung auf der Ebene Ausnahme getroffen wird und somit wird das Herunterfahren passieren, unabhängig von Heuristiken und Grundbetriebszeiten. Der Vorgang wird nur aufgehalten, wenn innerhalb der Verzögerungszeit eine Grundbetriebszeit beginnt. Fällt die Verzögerungszeit in eine Grundbetriebszeit, also zwischen Start- und Stopzeitpunkt, wird ein Rechner heruntergefahren, auch wenn die Grundbetriebszeit das zu diesem Zeitpunkt nicht vorsieht. Das gewünschte Verhalten wäre wohl eher, dass die Logik erkennt, dass der Rechner nach Ende der Ausnahme an sein soll und deshalb auf das Senden des Herunterfahrenbefehls verzeichtet. Dies ist eine nicht triviale Entscheidung, da dazu Daten aus zwei Kontrollschichten herangezogen werden müssen. Mit softEnforcement auf „an“ kann die Logik diese Entscheidung treffen, jedoch nur mit dem Nebeneffekt, dass die Logik einen Rechner immer anschaltet, sobald es eine Ausnahme oder Grundbetreibszeit gibt, die dieses vorsieht. Dadurch verliert der Benutzer zeitweise die Möglichkeit einen Computer von Hand auszuschalten bzw. das Powerman System wird ihn bei nächster Gelgenheit wieder einschalten.

B.5.2.8 maxForkCount — setN

Der Powerman Server wird im regelmäßigen Abstand von 5 Minuten aktiv. Bei jedem Mal sammelt er Informationen von den Clients, entscheidet für jeden ihm bekannten Client, welche Aktion durchzuführen ist und bereinigt die Datenbank. Gerade das Sammeln von Informationen ist kritisch. Die Clientsoftware stellt erst bei der Anfrage durch den Server die gewünschten Informationen zusammen, was sehr zeitaufwändig ist. Diese Informationen beeinflussen den unmittelbar folgenden Entscheidungsprozess. Der Server muss also warten, bis alle Informationen eingeholt wurden, bevor er fortfahren kann. Intern führt der Server eine Liste von Clients, von denen er weiß, dass sie an sind. Die Tatsache dass ein Rechner aus ist erkennt der Server nur daran, dass die Anfragen unbeantwortet bleiben, dann streicht er diesen Rechner aus seiner Liste. Diese

Timeouts dürfen nicht zu knapp bemessen sein, denn durch hohe Systemauslastung ist es möglich, dass ein paar Sekunden vergehen, bis ein Client antworten kann.

Aus dem Zeitaufwand zum Zusammenstellen der Informationen und der Timeouts ergibt sich für den Server ein Skalierungsproblem, wenn er alle Clients sequenziell abfrägt, denn er hat nur ein 5 minütiges Zeitfenster um all seine Aufgaben zu bewältigen, denn danach soll er eigentlich von vorne mit der Abarbeitung seiner Aufgaben beginnen. Da der Server viel Zeit mit Warten verbringt, eignet sich das Sammeln von Informationen sehr gut für Paralellisierung. Dabei erzeugt der Serverprozess Kopien von sich selbst, um mehrere Rechner mehr oder weniger gleichzeitig abfragen zu können, was vor allem das Problem der Timeouts handhabbar macht. Dieser immense zeitliche Vorteil geht jedoch zu Lasten der Performance des Servers. Im günstigsten Fall erstellt der Serverprozess eine Kopie von sich für jeden Rechner aus seiner internen Liste, was zu einem Skalierungsproblem bei der Performance des Servers führt.

Um wiederum dieses Problem in den Griff zu bekommen, muss die maximale Anzahl der gleichzeitig vorhandenen Kopien des Serverprozesses nach oben beschränkt werden. Dies ist die Bedeutung des Attributs maxForkCount. Es gibt an, wieviele Kopien der Serverprozess von sich höchstens erstellen soll. Reicht das nicht, um alle aktiven Rechner abzufragen, wartet der Server solange, bis bereits gestartete Abfragen beendet sind und erstellt dann weitere Kopien von sich um auch die restlichen Clients abfragen zu können. Ist der Wert dieses Attributs eins, so werden alle Clients sequenziell abgefragt. Ist der Wert 0 so wird der Server in einer Endlosschleife gefangen.

B.5.2.9 badStartupDelay — setN

Nachdem einem Rechner der Befehl zum Starten geschickt wurde, speichert das System intern, dass sich der Rechner im Status „Hochfahren“befindet. Da in dieser Zeit die Clientsoftware noch nicht aktiv ist, macht es keinen Sinn diesen Rechner abzufragen und ein Timeput würde auch nicht suggerieren, dass der Rechner aus ist. Tritt nun auf dem Client ein Fehler auf, sodass es nicht zum Start der Clientsoftware kommt, wird der Rechner intern nie einen anderen Status als „Hochfahren“bekommen. Hier benutzen wir wieder einen Timeout zum Erkennen von Problemen. Hat sich ein Client nach badStartupDelay Minuten nicht am System angemeldet, so wird der Client intern wieder auf „aus“.

Name	Standardwert	Bedeutung
DEBUG	0	Debug-Modus, wenn $\neq$ 0 produziert das System viel Output)
VERSION	1	Version des Systems
LOGFILE	/var/powerman/log/powerman.log	Log-Datei, in die die Steuerlogik und das SNMP-Trap Skript schreiben.
POWERMAN_PORT	4123	TCP-Port, auf dem der Net-SNMP Dienst der Clients auf Anfragen vom Server wartet
SNMP_COMMUNITY	pow	SNMP Community, die zum Auslesen der Clients benutzt wird
SNMP_RW_COMMUNITY	powerman	SNMP Community, die zum Schreiben von Daten auf die Clients benutzt wird
SNMP_HOSTNAME	1.3.6.1.2.1.1.5.0	Hostname oid
SNMP_OS	1.3.6.1.4.1.19518.4123.1.2.0	Betriebssystems oid
SNMP_IPTOMACINDEX	1.3.6.1.2.1.4.20.1.2.	IF-MIB IP- zu MAC-Adressen zuweisung
SNMP_MAC	1.3.6.1.2.1.2.2.1.6.	oid-Prefix von Mac-Adressen
SNMP_POWERMAN	1.3.6.1.4.1.19518.4123	oid der Powermanerweiterung
USB_DEVICES	/proc/bus/usb/devices	Übersicht über alle angeschlossenen USB-Geräte
WOL_BROADCAST	192.168.43.255	IP-Adresse an die WOL-Pakete geschickt werden
WOL_PORT	40000	UDP-Port an den WOL-Pakete geschickt werden
POWERCYCLE_TIMEOUT	10	Anzahl Sekunden, die beim Powercyclen zwischen Aus- und Anschalten gewartet werden soll
LOM_PASSWD	powerman	Passwort zum Telnetzugang der Sun LOMs
DATE_INFINITE	01.01.2038	größtmögliches Datum
DATE_MAX	mktime(0,0,0,12,31,2037)	größtmöglicher Timestamp
BINDIR	/usr/local/bin/	Verzeichnis, in dem sispmctl und digitemp liegen
STATE_OFF	0	Bezeichnung des Zustandes eines Clients, der aus ist
STATE_UP	1	Bezeichnung des Zustandes eines Clients, der gerade herauffährt
STATE_DOWN	2	Bezeichnung des Zustandes eines Clients, der gerade herunterfährt
STATE_ON	3	Bezeichnung des Zustandes eines Clients, der gerade an ist
$_confDB['TYPE']	pgsql	Zu verwendender Datenbankkonnektor
$_confDB['USER']	powerman	Benutzername zum Anmelden an der Datenbank
$_confDB['PASS']	default	Passwort zum Anmelden an der Datenbank
$_confDB['PROTO']	tcp	Verbindungstyp (tcp/unix)
$_confDB['HOST']	localhost:54321	IP-Adresse/Hostname und Port bzw. Domainsocket
$_confDB['NAME']	powerman	Name der Datenbank, die benutzt werden soll

Tabelle B.3: Möglichkeiten der config.php

C Dokumentation für Entwickler

Diese Dokumentation soll sich mit dem Erweitern des Powerman Systems beschäftigen und zum Schluss werden wir noch kurz auf das Portieren des Systems auf ein anderes Datenbanksystem ansprechen.

C.1 Powerman erweitern

Je nach Anwendungsgebiet entsteht an verschiedenen Stellen der Bedarf das System zu erweitern.

C.1.1 Neue Stromverbrauchscharakteristik

Schafft man zum Beispiel eine neue Art von Computer an, möchte man für diesen eine eigene Stromverbrauchscharakteristik anlegen, damit die Abschätzung des Stromverbrauchs näher an den tatsächlichen Stromverbrauch herankommt.

Die Berechnung des Stromverbrauchs richtet sich in unserem Modell ausschließlich nach der CPU-Last. Für das Powerman System ist die Stromverbrauchscharakteristik nichts anderes als ein Polynom $p(x) = \sum_{i=0}^{n} a_i x^i$ beliebigen Grades n. Gespeichert werden lediglich die Koeffizienten $a_0 ... a_n$ als ;-getrennte Liste in der Tabelle „characteristic“. Die Tabelle besteht aus drei Feldern. Das Feld „cid“ wird automatisch gefüllt und ist der Primär Schlüssel. Das Feld „name“ ist frei belegbar und ist der im Frontend angezeigte Name der Charakteristik. Das dritte Feld „coeff“ enthält die ;-getrennte Liste der Koeffizienten des Polynoms.

Durch das Hinzufügen von neuen Datensätzen in diese Tabelle sind beliebig viele Charakteristiken im System hinterlegbar.

C.1.2 Neues Frontendmodul

Es ist auch möglich das ganze Frontend um neue Funktionalitäten zu erweitern. Ein Modul besteht aus einer php-Datei in /var/powerman/php/www/page und muss mindestens drei Funktionen implementieren: init(), validate() und run(). Ein Modul wird aufgerufen, indem man die Variable „page“ beim Seitenaufruf definiert. Die gewünschte Funktion des Moduls wird mit der Variablen „func“ definiert Wird das Modul aufgerufen, so bindet die index.php zuerst die Datei ein und führt dann der Reihe nach die Funktionen init(), validate() und run() aus. Bei der Implementierung sollte man sich an folgende Konventionen und Funktionen halten:

- SQL-Querys in common/sql.php ablegen
- regular expressions zum überprüfen von Daten in common/regex.php ablegen
- query() (definiert in common/sql.php) zum Absetzen von Querys benutzen
- match() (definiert in common/regex.php) zum überprüfen von übergebenen variablen benutzen
- msg() (definiert in common/msg.php) zur Ausgabe von Statusmeldungen benutzen.
- Debug-Code immer in if(DEBUG){/* CODE */} Blöcke schreiben
- $GLOBALS['func'] beinhaltet vom WWW-Client übergebene Anweisungen an das jeweilige Modul

Funktion	**vorgeschlagene Bedeutung**
init()	Initialisierungen von Variablen etc.
validate()	Überprüfen von übergebenen Variablen
run()	Ausführen von Aktionen, Anzeigen von Eingabemaske etc.

Tabelle C.1: Benötigte Funktionen eines Moduls im Frontend

Literaturverzeichnis

[HAN 99] HEGERING, H.-G., S. ABECK und B. NEUMAIR: *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[WEB 1] PRITCHARD, STEPHEN: *Stromfressende Server bereiten IT-Managern Sorgen*. T-Online Business Portal, 2007, http://www2.t-online-business.de/c/11/29/78/14/11297814.html .

[WEB 2] DUISBURG-ESSEN, UNIVERSITÄT: *Wissen Sie, was Ihr Rechner verbraucht?* Zentrum für Informations- und Mediendienste, 2007, http://www.uni-due.de/zim/info/energiesparen.shtml .

[WEB 3] BENJAMIN BENZ, THORSTEN THIELE: *Netz-Schalter*. c't Magazin, 2005, http://www.heise.de/ct/projekte/netz-schalter/ .

[WEB 4] LIPPMANN, RICHARD: *Temperaturmessung mit Linux*. Richard Lippmann, 2007, http://lena.franken.de/hardware/temperaturmessung.html .

[WEB 5] FRIEDRICH, DOROTHEA: *Forrester Research: Umwelt ist ein wichtiger Faktor in der IT*. computerwoche.de, 2007, http://www.computerwoche.de/nachrichten/mittelstand/592768/index.html .

[WEB 6] INTEL: *Built-in Manageability and Proactive Security for Business Desktop PCs*. intel.com, 2007, http://download.intel.com/business/vpro/pdfs/vpro_wp.pdf .