University of Zurich
Department of Informatics

DIPLOMARBEIT – Communication Systems Group, Prof. Dr. Burkhard Stiller

Wuala Monitoringsystem
—
Erfassen, Speichern und Darstellen von relevanten Ereignissen in einem grossen verteilten Speichersystem

Gregor Berther
Urdorf, Schweiz
Matrikelnummer: 01-702-245

Betreuer: Dominik Grolimund
Abgabe: 4. Januar 2008

Universität Zürich
Institut für Informatik (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Schweiz

ifi

Diplomarbeit
Communication Systems Group (CSG)
Institut für Informatik (IFI)
Universität Zürich
Binzmühlestrasse 14, CH-8050 Zürich, Schweiz
URL: http://www.csg.uzh.ch/

Zusammenfassung

Monitoring ist ein wichtiges Konzept zur Auswertung relevanter Ereignisse eines überwachten Systems, aufgrund welchem unerwünschte Entwicklungen frühzeitig erkannt und strategische Entscheidungen für zukünftige Entwicklungen getroffen werden. In verteilten Systemen ist Monitoring ein nicht triviales Unterfangen, da die gewünschten Daten und Ereignisse nicht zentral verfügbar, sondern auf den einzelnen Knoten des Netzes verstreut sind. Das System hat keinen einfach zu überprüfenden globalen Zustand, sondern besteht aus einer Vielzahl autonom interagierenden Einzelknoten. Wuala ist ein solches grosses verteiltes Speichersystem, welches die Daten der Benutzer in einem Peer-to-Peer Netz speichert. Diese Diplomarbeit beschreibt den Aufbau, die Umsetzung und die Einsatzmöglichkeiten des Wuala Monitoringsystems. Es stellt ein Framework zur schnellen und einfachen Reporterfassung in allen Teilen von Wuala zur Verfügung. Die gesammelten Daten werden in einer zentralen Datenbank gespeichert, wo sie mit verschiedenen Visualisierungsprogrammen dargestellt und ausgewertet werden können.

Abstract

Monitoring is an important concept to evaluate relevant events in a supervised system, which helps to recognize undesired developments and to make strategical decisions for further improvements. Especially in distributed systems, monitoring is not trivial, because the necessary data is not locally available, but scattered on all of the network's nodes. The system has no global state that can be simply supervised, but consists of lot of autonomous interacting nodes. Wuala is such a large-scale distributed storage system, which stores the user's data in a peer-to-peer network. This diploma thesis describes the design and the implementation of the Wuala monitoring system. It provides a framework for a quick and simple collection of reports in every single part of Wuala. The gathered data is stored on a centralized database, where it can be visualized and analyzed with various programs.

Danksagung

Ich möchte mich bei meinen beiden Betreuern von der Caleido AG, Dominik Grolimund und Luzius Meisser, und dem gesamten Wuala-Team bedanken für ihre Hilfe und Unterstützung und insbesonders für die Möglichkeit Teil eines aufstrebenden Startup Unternehmens mit einem grossartigen Produkt zu sein. Des Weiteren bedanke ich mich bei Prof. Dr. Burkhard Stiller, dass er mich dieses frei gewählte Thema als externe Diplomarbeit innerhalb der Communications Systems Group ausarbeiten liess.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die digitale Vernetzung nimmt in der heutigen Welt immer mehr zu. Das Internet ist für viele Leute nicht mehr aus ihrem Leben wegzudenken, sei es nun am Arbeitsplatz oder für den privaten Gebrauch. Somit ist es nicht erstaunlich, dass immer neue Applikationen auf den Markt kommen, welche die Grenzen des Internets aufweichen. Der Benutzer soll von überall her Zugriff auf seine Dateien haben und dies ohne grossen Aufwand oder vorgängige Planung.

Die Caleido AG entwickelt mit Wuala eine solche Applikation, die es dem Benutzer erlaubt, seine in Wuala gespeicherten Daten von jedem Computer aus abzurufen, auf welchem Wuala installiert ist.

Wuala [1] ist ein grosser verteilter Onlinespeicher. Das Ziel ist es, den Benutzern die Erreichbarkeit ihrer Dateien nicht durch Speicherung auf vielen Servern zu garantieren, sondern durch die Ausnutzung brachliegender Ressourcen jedes einzelnen Benutzers. Diese Ressourcen sind freier Festplattenspeicher sowie die Down- und Uploadbandbreite jedes einzelnen Benutzers. Wuala organisiert alle Teilnehmer in einem Peer-to-Peer Netzwerk.

Diese Diplomarbeit beschäftigte sich mit der Erarbeitung und Umsetzung eines Monitoringsystems, welches die speziellen Ansprüche und Anforderungen von Wuala als verteiltes Peer-to-Peer Netzwerk erfüllt.

1.1 Motivation

Das Feld vom verteilten Onlinespeicher und der damit verbundenen Applikation macht den Aufbau eines Monitoringsystems zu einem nicht trivialen Unterfangen. Die interessanten Ereignisse und Daten sind verstreut auf verschiedene Server und auf die Computer der Benutzer. Sie sind deshalb nicht immer und zu jeder Zeit verfügbar. Auch können Daten im Netz verloren gehen, wenn die Verbindung zwischen den kommunizierenden Knoten zusammenbricht oder nicht zustande kommt. Dies sind die Herausforderungen, die es beim Aufbau eines Monitoringsystems für Wuala zu beachten gilt.

1.2 Problemstellung und Ziele

Das Ziel dieser Arbeit ist es, ein lauffähiges und einsatzbereites Monitoringsystem für Wuala aufzubauen. Da sich Wuala zu der Zeit, in der diese Diplomarbeit entsteht, in der ersten geschlossenen Alphaphase befindet, ist ein System erforderlich, das schon nach relativ kurzer Zeit einsatzfähig ist. Damit schon in dieser frühen Phase des Produktes wichtige Ereignisse erfasst werden können und aufgrund deren Auswertung wichtige Design- und Marketingentscheidungen getroffen werden können. Das Monitoringsystem soll als Framework aufgebaut sein, das eine Erweiterung und Erstellung von neuen Monitoringaufgaben vereinfacht und beschleunigt.

Des Weiteren muss das Monitoringsystem auf die speziellen Anforderungen eines grossen verteilten Speichersystems eingehen. Die Erfassung der Daten muss einfach und unabhängig vom Ort des Auftretens des Ereignisses sein. Ebenso muss die Skalierung der Daten in Hinblick auf die erwarteten steigenden Benutzerzahlen berücksichtigt werden.

1.3 Aufbau der Arbeit

In Kapitel 2 wird Wuala und die dabei eingesetzten Technologien vorgestellt, damit verständlich wird in welchem Umfeld das Monitoringsystem eingesetzt wird. Kapitel 3 befasst sich mit Arbeiten, in welchen verwandte Lösungen im Gebiet des Monitorings von verteilten Systemen beschrieben werden. Danach werden in Kapitel 4 der konkrete Aufbau und die Besonderheiten des Wuala Monitoringsystems erklärt, gefolgt von Kapitel 5 mit der Beschreibung des Designs und der Implementierung des Monitoringsystems. Kapitel 6 gibt einige Beispiele für reale Einsatzmöglichkeiten und in Kapitel 7 wird das Monitoringsystem evaluiert. Kapitel 8 enthält die Schlussfolgerungen, die aus dieser Diplomarbeit gezogen werden können, sowie einen Ausblick auf mögliche Weiterentwicklungen, die zukünftig am Wuala Monitoringsystem vorgenommen werden könnten.

Kapitel 2

Ausgangslage

Dieses Kapitel beschreibt das Produkt und das Umfeld, in welchem die vorliegende Diplomarbeit zustande gekommen ist. Vor allem werden die eingesetzten Technologien und neuen Ansätze beschrieben, welche das Speichermodell von Wuala von Produkten anderer Anbieter unterscheidet. Die neue Art der Datenspeicherung und die Besonderheiten der Applikation sollen zeigen, welche Herausforderungen an ein Monitoringsystem für Wuala gestellt werden.

2.1 Wuala Überblick

Wuala ist eine von der Caleido AG entwickelte Javaapplikation für Windows, Mac und Linux [1] (Abbildung: 2.1). Nachdem der Benutzer eine kleine Applikation installiert und ein Konto eingerichtet hat (Benutzername und persönliches Passwort genügen), kann er im Wuala-Netzwerk teilnehmen. Wuala erlaubt es dem Benutzer seine Dateien auf einer Art Onlinefestplatte zu speichern. Das heisst, die Dateien werden mit der Wuala Applikation ins Wuala-Netzwerk geladen und können ab diesem Zeitpunkt von jedem Computer, auf dem Wuala installiert ist und der ans Internet angeschlossen ist, heruntergeladen werden. Dieser Anwendungsfall eignet sich vor allem für Backups oder für das Übermitteln von privaten Dateien, zum Beispiel vom Arbeitsplatz an den Computer zu Hause. Als nächster Schritt können die Dateien gezielt mit Freunden oder mit einer Gruppe geteilt werden. Dies ist zum Beispiel für Ferienfotos nützlich. Statt riesige E-Mails zu versenden, werden die Fotos einmal ins Wuala geladen und für die berechtigten Freunde freigegeben. Als dritte Option kann man die hoch geladenen Dateien auch veröffentlichen. Dann sind sie im öffentlichen Bereich der Applikation für alle Wuala-Benutzer sichtbar und können von diesen betrachtet, heruntergeladen oder sogar im eigenen privaten Wuala-Bereich gespeichert werden (als Link oder als Kopie der Datei).

Die Rechteverwaltung für dieses einfache dreistufige System - *Zugreifen, Teilen, Veröffentlichen (access, share, publish)* - wird über ein Ordnersystem verwaltet. Die Zugriffsrechte werden auf einem Ordner definiert und alle darin enthaltenen Dateien und Unterordner erben mindestens dasselbe Zugriffsrecht. Das heisst ein Ordner, der sich in einem Ordner

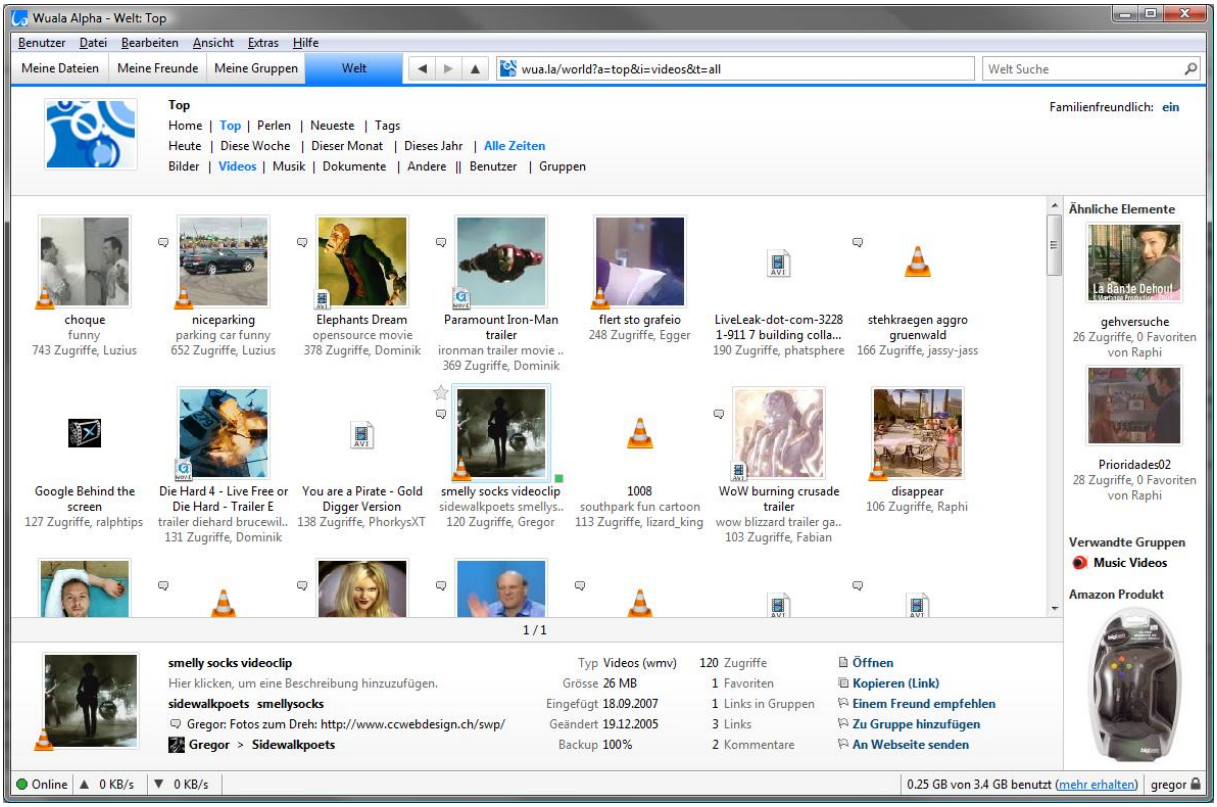

Abbildung 2.1: Wuala Screenshot

mit der Berechtigung *Teilen* befindet, kann nicht auf *Zugreifen* (also nur privat sichtbar) gesetzt werden. Das Setzen auf *Veröffentlichen* ist jedoch möglich, da die Zugriffsrechte dadurch erweitert werden.

Jeder Benutzer von Wuala erhält zu Beginn ein Gigabyte Speicherplatz, über welches er frei verfügen kann. Reicht dieser Platz nicht aus, so kann der Benutzer mehr Speicher dazuverdienen, wenn er bereit ist einen Teil seiner eigenen lokalen Festplatte für das Wuala-Netzwerk freizugeben. Dies ist jedoch nur für Computer möglich, die im Durchschnitt mindestens 17% pro Tag online sind (ca. 4 Stunden). Der neu dazuverdiente Speicher errechnet sich dann aus der Menge der lokal gespeicherten Daten mal der Onlinezeit (Beispiel: Bei einem Benutzer sind insgesamt 40 GB Wualadaten lokal gespeichert und die Wuala Applikation ist durchschnittlich 6 Stunden online pro Tag. Dies ergibt 40 x 25 % = 10 Gigabyte zusätzlich verfügbaren Onlinespeicher).

Das Ziel ist es hunderttausende von Benutzern von Wuala zu überzeugen und im hart umkämpften Markt der Onlinespeicheranbieter zu bestehen. Eine Auswahl von Anbietern sind Omnidrive [2], All My Data [3], Box.net [4] oder Omemo [5].

2.2 Wuala Technologien

Die oben erwähnte Möglichkeit, den lokalen Festplattenspeicher als Teil eines gesamten Netzes anzubieten, zeigt den neuen Ansatz, der hinter Wuala steckt. Während andere

Anbieter von Onlinespeicher auf riesige Serverfarmen setzen, reduziert Wuala den Einsatz von Servern durch den Einsatz einer Peer-to-Peer (P2P) Lösung. Der Grundgedanke hinter diesem Ansatz ist es, die brachliegenden Ressourcen (Festplattenspeicher und Bandbreite) von mit dem Internet verbundenen Computern auszunutzen und damit einen stabilen, persistenten und verteilten Onlinespeicher anzubieten. Jeder einzelne Teilnehmer des Netzwerks kann gleichzeitig auch Anbieter von Speicher sein. Da die Onlinezeiten, Zuverlässigkeit und Uploadgeschwindigkeit von normalen Benutzer-PCs und Notebooks natürlich weit hinter denen von Servern zurückliegen, mussten Lösungen gefunden werden, damit das Wuala-Netzwerk die Anforderungen erfüllen kann, die man an einen persistenten Speicher stellt.

2.2.1 Aufbau des Netzes

Alle Knoten des Wuala-Netzwerkes sind in einer verteilten Hashtabelle (DHT) organisiert. Die eingesetzte DHT wurde von den Wuala-Entwicklern an der ETH Zürich unter dem Namen Kangoo [6] entwickelt. Die Knoten enthalten die Routinginformationen, so dass jeder Knoten im Netz von jedem anderen Knoten aus möglichst effektiv und schnell angesprochen werden kann.

Die einzelnen Knoten im Netz werden in drei Kategorien unterschieden, in Supernodes (Superknoten), Storagenodes (Speicherknoten) und Clientnodes (Klientknoten). Die Unterscheidung der Typen erfolgt über die Güte und Qualität der Netzanbindung. Werden die Kriterien erfüllt, so kann der Knoten ein Supernode werden, der Informationen über die DHT enthält und als Weiterleitungspunkt im Routingverfahren agiert. Zum Storagenode wird ein Knoten, wenn er eingehende Verbindungen akzeptiert und der Benutzer zusätzlich bereit ist mit seinem lokalen Speicher ans Wuala-Netzwerk beizutragen. Die übrigen Knoten, die keinen Speicher teilen wollen oder die Anforderungen an die Konnektivität nicht erfüllen, sind Clientnodes. Das Ziel ist es, dass der P2P Ansatz dazu beiträgt, dass die Serverkosten auch bei riesigen Datenmengen in Grenzen gehalten werden können.

Für den Betrieb von Wuala werden zusätzlich noch Server eingesetzt. So werden die Metadaten zentral gespeichert um einen schnellen Zugriff zu gewährleisten. Ebenso werden kleine Dateien zentral gespeichert, da sich für diese der Aufwand einer P2P Verteilung nicht lohnt.

2.2.2 Gewährleistung der Verfügbarkeit

Die Verfügbarkeit von Daten die zentral auf einem Server gespeichert sind, steht und fällt mit der Erreichbarkeit des Servers. In einem P2P Netz kann die persistente Verfügbarkeit der Daten nur mit der Einführung von Redundanz gewährleistet werden, da sonst immer alle Knoten des Netzwerks online sein müssten. Die einzelnen Daten werden in n-Fragmente zerteilt und diese werden mittels Fehlercodes in m-Fragmente verschlüsselt, bis die Verfügbarkeit der Datei gewährleistet werden kann. Diese einzelnen Fragmente werden dann auf den Storagenodes gespeichert, die zum Zeitpunkt des Hochladens online sind. Der Effekt des Fehlercodes ist es nun, dass für das vollständige Herunterladen der Datei

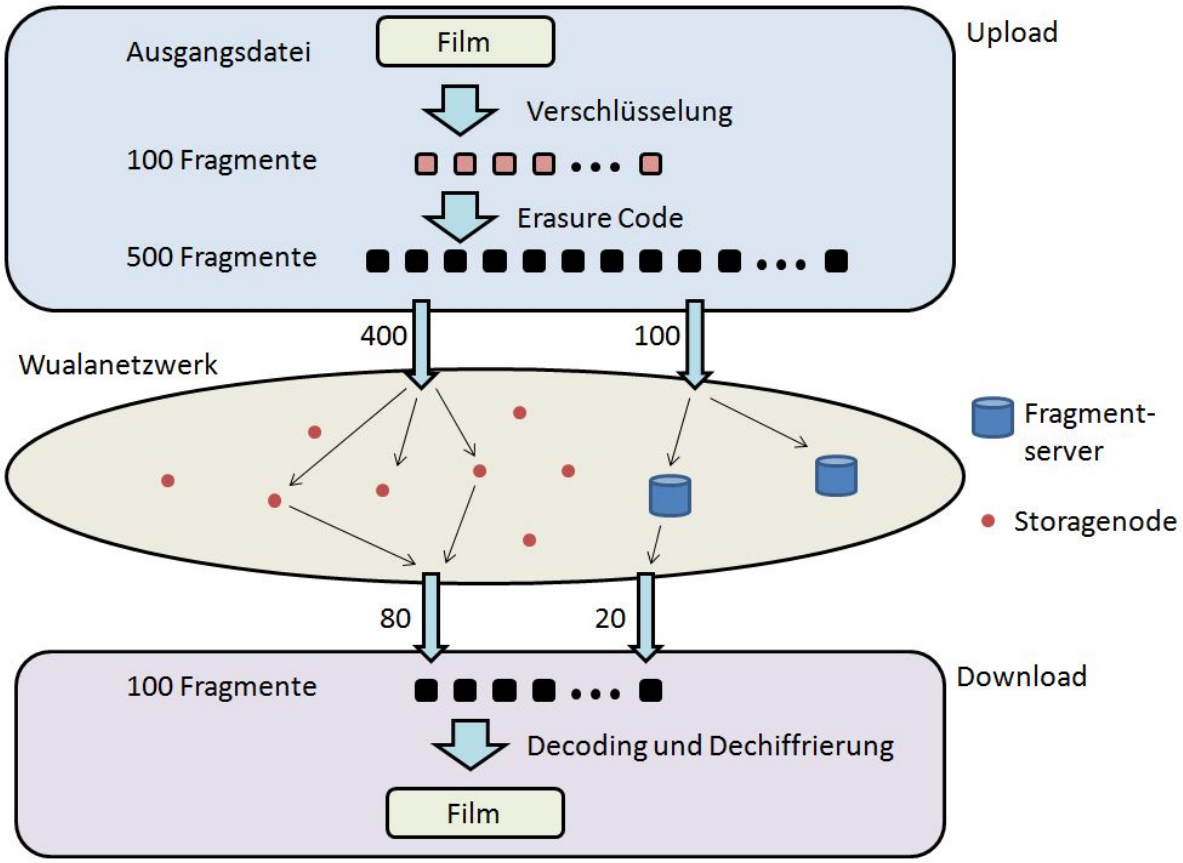

Abbildung 2.2: Beispiel für den Up- und Download einer Datei im Wuala-Netzwerk

nicht mehr alle m-Fragmente gefunden werden müssen. Es genügen n beliebige Fragmente aus m um die Datei wiederherstellen zu können. Abbildung 2.2 illustriert diesen Vorgang zur besseren Verständlichkeit.

Da sich die Anzahl und Verfügbarkeit der Storagenodes stetig ändert, werden die Dateien jedes Benutzers gewartet, wenn er in Wuala eingeloggt ist. Ist die Verfügbarkeit für eine Datei zu klein, werden zusätzliche Fragmente generiert, die auf den Storagenodes gespeichert werden. Die Verfügbarkeit eines Fragments beruht auf der Verfügbarkeit des Storagenodes auf dem es gespeichert ist. So müssen weniger Fragmente gespeichert werden, wenn die Onlinezeit der einzelnen Storagenodes hoch ist. Bei kleiner Onlinezeit werden entsprechend mehr Fragmente benötigt.

Als zusätzlicher Sicherheitsmechanismus werden in der Anfangsphase von Wuala die ersten n- verschlüsselten Fragmente einer Datei zusätzlich auf zentralen Servern gespeichert, so dass die Datei gefunden werden kann, auch wenn im schlechtesten Fall kein einziger Storagenode online ist.

2.2.3 Sicherheit und Fairness

Wuala legt einen hohen Stellenwert auf die Privatsphäre des Benutzers [7]. So wird garantiert, dass niemand auf die privaten Dateien zugreifen kann, wenn er nicht über die richtigen Kontoinformationen verfügt. Die privaten Dateien der Benutzer sind auch für die Wuala-Betreiber nicht sichtbar, da die Dateien auf dem eigenen Computer, bevor sie ins Wuala-Netzwerk geladen werden, verschlüsselt werden. Für die Organisation der Zugriffsrechte und der Schlüssel wird Cryptree [8] eingesetzt. Damit werden die unterschiedlichen Rechte, wie Gruppenzugriff und Teilen mit Freunden, gehandhabt.

Um die Benutzer dazu zu animieren selber Speicher bereit zu stellen, wird das Fairnesssystem Havelaar [9] eingesetzt. So werden Benutzer, die bereit sind lokalen Speicher zur Verfügung zu stellen, einerseits mit mehr Onlinespeicher belohnt (je länger die durchschnittliche Onlinezeit, desto grösser der Onlinespeicher) und andererseits werden ihre Downloads priorisiert, dass heisst sie bekommen mehr Bandbreite zur Verfügung gestellt.

Kapitel 3

Grundlagen

Für das Monitoring eines grossen verteilten Speichersystem gelten ähnliche Voraussetzungen wie für Cluster und Gridsysteme. [10] und [11] beschreiben Ansätze, die dafür das Simple Network Management Protocol (SNMP) einsetzen. [10] ist für den Einsatz in heterogenen Netzwerken geeignet und ist für die Zustandsmessungen der Teilsysteme verantwortlich, die zentral ausgewertet werden. [11] beschreibt eine Lösung für grosse Datenzentren. Ein Multicast Ansatz zum Monitoring von high-performance Computersystemen wird in [12] beschrieben, womit die Bandbreitenbelastung verringert werden kann.

Die meisten Arbeiten beschäftigen sich mit der Auswertung der Systemperformance der Teilsysteme und des Gesamtsystems, wie die Auslastung der Bandbreite, die CPU-Last, Internetkonnektivität oder Speicherplatz [13]. [14] setzt auf ein zweidimensionales Monitoring von Performancedaten; die vertikale Komponente erfasst die Performancedaten auf einem einzelnen Knoten, die horizontale Komponente enthält die Daten über die Netzperformance.

Ein Prinzip für die Verbreitung von Nachrichten in einem verteilten System ist das Gossip Protokoll [15]. Eine Nachricht lässt sich auf diese Weise mit O(log(N)) Runden (N = Anzahl Knoten) im gesamten Netzwerk verbreiten. Eine weitere herausfordernde Angelegenheit bezüglich der Datensammlung in verteilten Systemen ist die Möglichkeit von globalen Schnappschüssen (Snapshots) [16].

Die dezentrale Aggregation von Daten innerhalb des verteilten Systems ist der Ansatz von [17]. Nebst dem Einsatz des Gossip Protokolls wird das Netz in hierarchische Strukturen unterteilt (Zonen), in welchen die Daten separat aggregiert werden.

Ganz auf zentrale Strukturen verzichtet [18]. Hier wird ein Konzept für das Monitoring von P2P Netzen vorgestellt, das selbst als P2P Netz organisiert ist. Das System beinhaltet eine eigene deklarative Sprache P2PML, mit welcher die einzelnen Monitoringaufgaben definiert werden können.

All diese vorgestellten Arbeiten befassen sich mit der Problematik, dass Monitoring in einem verteilten System nicht trivial zu lösen ist, da die relevanten Daten und Ereignisse nicht von einem zentralen System oder Zugriffspunkt abgerufen werden können. Diesen

Übergang von verstreuten und nicht immer verfügbaren Ereignissen zu statistisch relevanten und einfach auszuwertenden Daten gilt es mit dem Wuala Monitoringsystem zu ermöglichen.

Kapitel 4

Monitoringsystem für Wuala

4.1 Wahl des zentralen Ansatzes

Keines der unter Kapitel 3 erwähnten Produkte und Ansätze kann ohne Änderungen oder Anpassungen für das Wuala Monitoringsystem eingesetzt werden. Einige Systeme sind sehr stark auf die Sammlung von Performancedaten fokussiert. Dies ist zwar auch ein Teilaspekt, der mit dem Wuala Monitoringsystem abgedeckt werden soll, jedoch wäre eine Einschränkung auf diesen zu stark. Die meisten der zu erfassenden Daten sind stark auf die Applikation bezogen und deshalb wird ein direkt im Wuala Sourcecode integrierter Ansatz verfolgt.

Der Einsatz von selbstaggregierenden Schnappschüssen bringt den Vorteil, dass die entstehende Netzlast gleichmässig verteilt ist, da jeder Knoten einen Teil der Rechenleistung beisteuert. Bei einer zentralen Aggregation wird der Zielknoten stark beansprucht, jedoch sind die Knoten von jeglichem zusätzlichen Rechenaufwand befreit. Bei der Netzaggregation ist auch das Missbrauchspotential grösser: Ein Knoten auf einer höheren Hierarchieebene kann einerseits die Daten für seinen Teil des Netzwerkes fälschen und weiterleiten oder andererseits die Daten für eigene Zwecke auswerten. Bei einer zentralen Anlaufstelle kann der Knoten immer noch seine eigenen Daten fälschen, aber der Einfluss auf den gesamten Schnappschuss bleibt minimal. Der Einsatz eines Gossip Protokolls enthält ebenfalls ein grösseres Missbrauchspotential, da die Knoten absichtlich falsche Daten im Netzwerk verbreiten könnten.

Diese Gründe führten zu der Wahl, dass das Wuala Monitoringsystem als zentrales System umgesetzt wurde. Für die Nachrichtenverbreitung können die Routingmechanismen der Wuala DHT benutzt werden. Die Skalierung des Systems beruht auf zwei Aspekten. Einerseits können die Anzahl der eintreffenden Nachrichten sinnvoll eingerichtet werden, das heisst die Anzahl von Reports können vom Entwickler selbst begrenzt werden (Abschnitt 5.1.3.3). Andererseits kann die Abarbeitung einer temporären Überlast mit der Verarbeitung in weniger intensiven Zeitabschnitten ausgeglichen werden. Die zusätzliche Netzbelastung durch das Monitoringsystem ist im Vergleich zur normalen Netzaktivität gering, so dass eine zentrale Lösung einen besseren Tradeoff von Aufwand und Nutzen bringt. Ausserdem führte der Einsatz dieser pragmatischen Lösung dazu, dass das Wuala

Monitoringsystem schon in einer frühen Phase für die Sammlung relevanter Daten eingesetzt werden konnte.

4.2 Anforderungen

Die Aufgabe ein Monitoringsystem für Wuala zu planen und zu implementieren, muss von verschiedenen Blickwinkeln aus betrachtet werden. Einerseits gilt es innert nützlicher Frist ein einsatzfähiges System zu bauen, das produktiv eingesetzt werden kann. Andererseits dürfen die Anforderungen nicht aus den Augen verloren werden, so dass das System auch bei vielen Benutzern noch skaliert und einsatzfähig bleibt. [19] unterscheidet drei unterschiedliche Dimensionen von Skalierung für verteilte Systeme: Eine numerische (Zunahme der Benutzer), eine geographische (Lokalisation der Knoten) und eine administrative (Aufwand für den Unterhalt) Komponente. Für das Monitoringsystem sind vor allem die numerische und die administrative Komponente zu beachten. Mit dem Anstieg der Benutzerzahlen werden die anfallenden Datenmengen zunehmen.

Der Aufwand für das Erfassen und Durchführen von Erhebungen im Netzwerk soll einfach gehalten werden, so dass das Monitoringsystem als unterstützende Applikation eingesetzt wird und den Betrieb von Wuala zu verbessern hilft. Die Bereitstellung eines Frameworks hilft, dass neue Monitoringaufgaben schnell erstellt und schon bald im realen System durchgeführt werden können.

Ebenfalls eine wichtige Anforderung an das Monitoringsystem ist eine gute Wahl des einzusetzenden Speichersystems. Es gilt die Vorteile und Nachteile bezüglich Speichergrösse und Eignung zur Auswertung und Visualisierung der gespeicherten Daten zu berücksichtigen. Ebenso müssen Fragen über die einfache Erweiterung, Kosten und Ausfallsicherheit in die Überlegungen miteinbezogen werden.

4.3 Aufbau des Monitoringsystems

Die Abbildung 4.1 zeigt den grundlegenden Datenfluss des Monitoringsystems. In allen Teilen des Wuala-Netzwerkes können Daten erfasst werden, welche an eine zentrale Serverinstanz gesendet werden. Die eintreffenden Daten werden vom Server aufbereitet und in eine Datenbank gespeichert. Die Datenauswertung und Visualisierung nutzt nun diese Daten.

Als Datenquellen kommen die einzelnen Computer des Wuala-Netzwerkes (die Knoten des Peer-to-Peer Netzes) und die Wuala eigenen Server in Frage.

Das Monitoringsystem für Wuala kann in zwei voneinander unabhängige Subsysteme gegliedert werden. Dies sind einerseits die Reporterfassung und Reportspeicherung und andererseits die Visualisierung der gesammelten Ereignisse. Diese Unterteilung wurde vorgenommen, um verschiedenen Aspekten des Monitoringsystems für Wuala gerecht zu werden. Durch die Trennung der beiden Teil erhöht sich die Flexibilität, indem unterschiedliche Implementierungen für beide Subsysteme eingesetzt werden können.

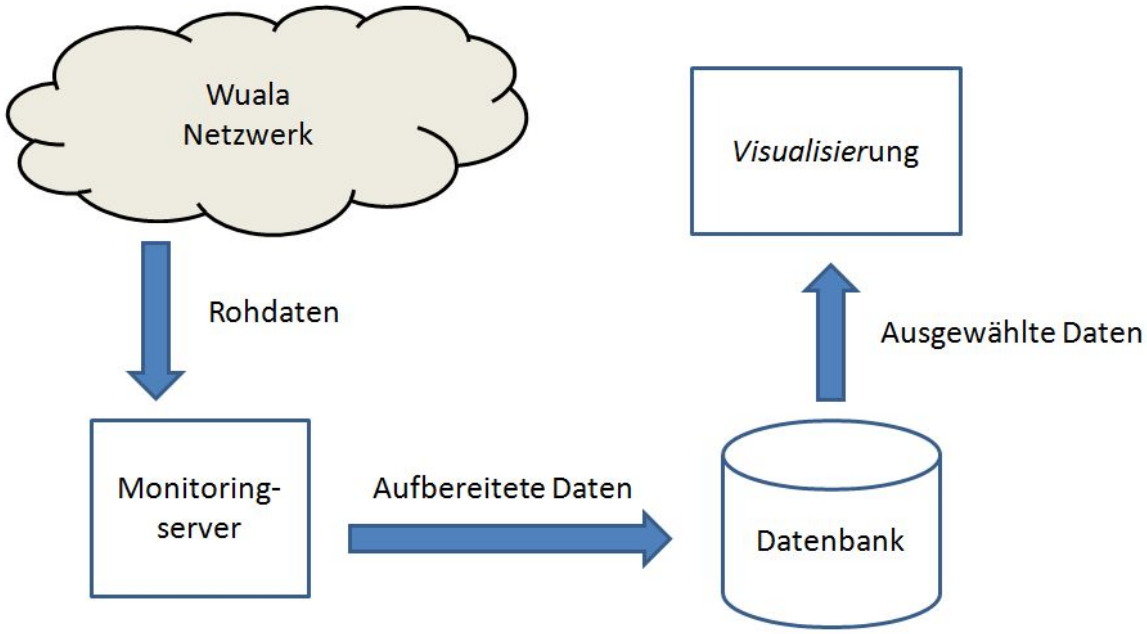

Abbildung 4.1: Datenfluss im Monitoringsystem

Die Reporterfassungskomponente erlaubt eine schnelle und vielseitige Sammlung vieler Daten und Ereignisse in allen Teilen des Gesamtsystems. Ein wichtiger Punkt bei der Umsetzung ist die Berücksichtigung von möglichen Kapazitätsengpässen. Abschnitt 4.5 zeigt die Überlegungen, die bezüglich der Skalierbarkeit vorgenommen wurden. Die ausgearbeitete Lösung setzt auf zwei verschiedene Ansätze. Die Standard-Reporterfassung kann zur Erfassung aller möglichen Ereignisse im gesamten System eingesetzt werden. Die Schnappschusssammlung hingegen hilft, um einen momentan gültigen Überblick über das Gesamtsystem zu erhalten.

Das Ziel der Visualisierungskomponente des Systems ist es verschiedene Applikationen bereitzustellen, mit welchen die von der Reporterfassungskomponente erhobenen Daten dargestellt und ausgewertet werden können.

Die Schnittstelle für diese beiden Subsysteme ist das Speichersystem in welchem die gesammelten Daten abgelegt werden. Die Auswahl des verwendeten Speichersystems wird in Abschnitt 4.6 erläutert.

In [20] wird ein Monitoring Modell vorgestellt, das in die folgenden vier Bereiche gegliedert ist: Erzeugung, Verarbeitung, Verteilung und Präsentation. Im Wuala Monitoringsystem kann die Datenerzeugung überall geschehen, die Verarbeitung wird von der Reporterfassungskomponente durchgeführt. Verteilung und Präsentation der Ergebnisse sind Aufgaben der Visualisierungskomponente.

4.4 Datenerfassung

Das Wuala Monitoringsystem unterstützt nebst der Standard-Reporterfassung einen zweiten Ansatz, um einen Schnappschuss über den Zustand des Systems zu einem fixen Zeitpunkt zu erfassen. Dieser Abschnitt beschreibt, welche Methoden dabei angewendet werden.

4.4.1 Aggregation im Netz

Die erste Idee für das Sammeln von Schnappschussdaten war die eines selbst aggregierenden Reports [17], der theoretisch von jedem Knoten im System ausgelöst werden kann. Der auslösende Knoten fungiert in diesem Ansatz auch als Zielknoten für den bereits fertig aggregierten Endreport. Das Vorgehen sieht wie folgt aus: Der gesamte Adressraum des Netzwerkes wird in n gleich grosse Bereiche unterteilt. Der Wurzelknoten sendet nun eine Anfrage an einen Knoten in jedem Adressbereich. Die Knoten die einen solchen Anfrage erhalten, leiten diese an alle bekannten Nachbarn im selben Adressbereich weiter. Diese Weiterleitung geschieht so lange, bis die Knoten keine neuen Nachbarn mehr haben; das heisst keine Nachbarn, die die Anfrage noch nicht bekommen haben. Ist dies der Fall, so wird der Wert der Anfrage berechnet und das Resultat an den Sender der Anfrage zurückgesendet. Sobald diese Zwischenknoten alle Antworten auf ihre Anfragen erhalten haben, werden die eingetroffenen Rückmeldungen zu einem einzigen Report aggregiert. Dieses aggregierte Resultat wird nun wiederum an den eigenen anfragenden Knoten zurückgesendet. Dies geschieht so lange, bis schlussendlich n Reports beim Wurzelknoten eintreffen, welche dieser ebenfalls zu einem einzigen Gesamtreport zusammenfasst (Abbildung: 4.2).

Der Grundgedanke hinter diesem Ansatz ist die gleichmässige Lastverteilung im gesamten Netzwerk. Im theoretischen Optimalfall sendet jeder Knoten so viele Anfragen wie er Nachbarn hat und er erhält eben so viele Antwortnachrichten. Diese Idee kann nur umgesetzt werden, wenn die Verlässlichkeit, dass die gesendeten Nachrichten ankommen, hoch ist und wenn der Onlinestatus der Knoten über den Zeitraum der Reporterfassung stabil bleibt.

Leider ist eine Voraussetzung im Wuala-Netzwerk nicht gegeben. Der grosse Schwachpunkt des Systems ist, dass die im Netz befindlichen Knoten jederzeit offline gehen können oder, dass sich der Status eines Knoten zwischen dem Senden der Anfrage und dem Empfangen der Antwort geändert hat (z.B.: Clientnode wird Storagenode oder Storagenode wird Supernode usw.). Damit wären einige Rückantwortadressen ungültig und ganze Teilbereiche des Netzes können verloren gehen.

4.4.2 Aggregation beim Wurzelknoten

Somit scheint sich ein leicht geänderter Ansatz besser für das Wuala-Netzwerk zu eignen. In diesem wird ebenfalls ein beliebiger Teilnehmer des Netzwerks als Wurzelknoten bestimmt. Dieser Knoten beschafft sich eine Liste aller Supernodes, die momentan online sind und sendet jedem die Anfrage zu. Die Supernodes unterhalten selbst eine Liste

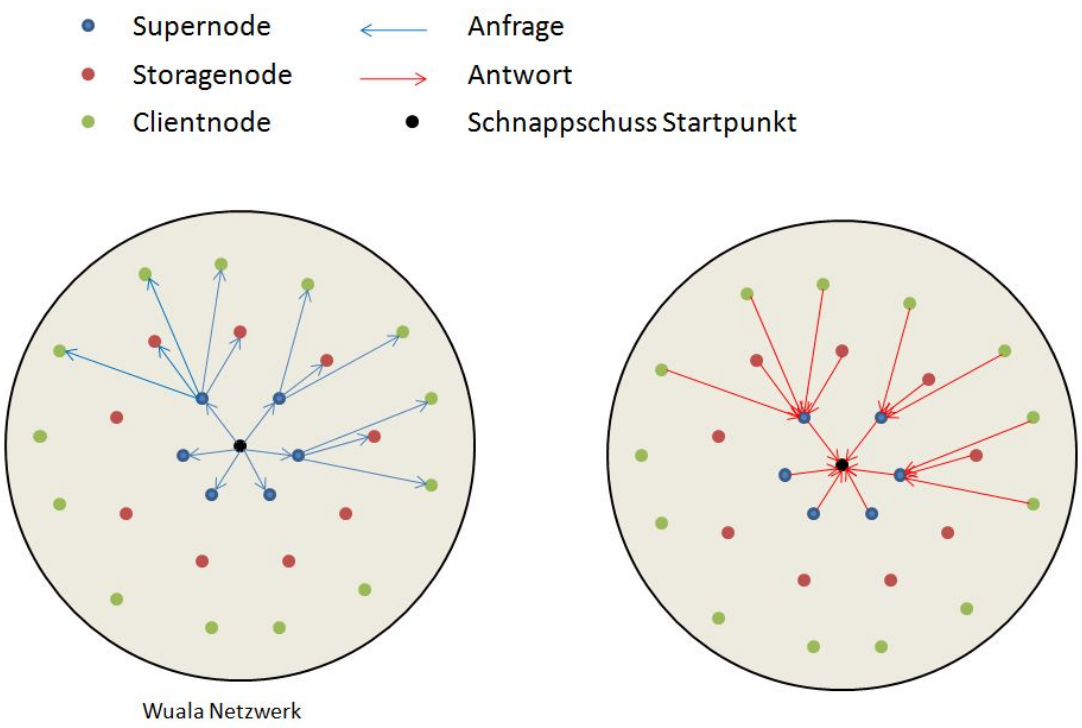

Abbildung 4.2: Antwortaggregation im Netz

mit den ihnen bekannten Client- und Storagenodes. Also wird die Nachricht an all diese weitergeleitet. Nun wird der Wert der Anfrage auf jedem Client berechnet und das Ergebnis an den Wurzelknoten direkt zurückgesendet (Abbildung: 4.3). Erst hier werden die Einzelwerte zu einem Endergebnis aggregiert.

Dieser Ansatz ist gegenüber 4.4.1 zu bevorzugen, da das Endergebnis robuster gegen Ausfälle oder Statusänderungen der Knoten reagiert. Wenn zum Beispiel ein Supernode offline geht, werden die bis an hin ihm zugeordneten Knoten trotzdem ein Resultat an den Wurzelknoten liefern. Der Nachteil dieses Ansatzes ist, dass die Last von eintreffenden Ergebnissen für den Wurzelknoten gross ist. Der Wurzelknoten muss also genügend Ressourcen erhalten, damit er in der Lage ist, viele Einzelergebnisse zu einem Hauptreport zu aggregieren.

4.4.3 Aggregation auf zusätzlichen Servern

Die Aggregation auf zusätzlichen Servern ist eine Mischung aus den beiden vorherigen Ansätzen. Die Nachrichtenverbreitung funktioniert ebenfalls über die Knotenstruktur der Wuala DHT. Die Aggregation der Antwortdaten erfolgt aber auf mehreren Servern, die ihre Resultate dem Wurzelknoten zur Schlussaggregation zusenden (Abbildung: 4.4). Dazu muss bei der Verbreitung der Anfrage die Adresse des Aggregationsservers mitgeschickt werden, der für den Adressbereich des Knoten zuständig ist. Der Vorteil dieses Ansatzes liegt einerseits darin, dass die Aggregation nicht im eigentlichen Wuala-Netzwerk erfolgt und somit das Missbrauchspotential bei der Aggregation auf die Fälschung von Daten

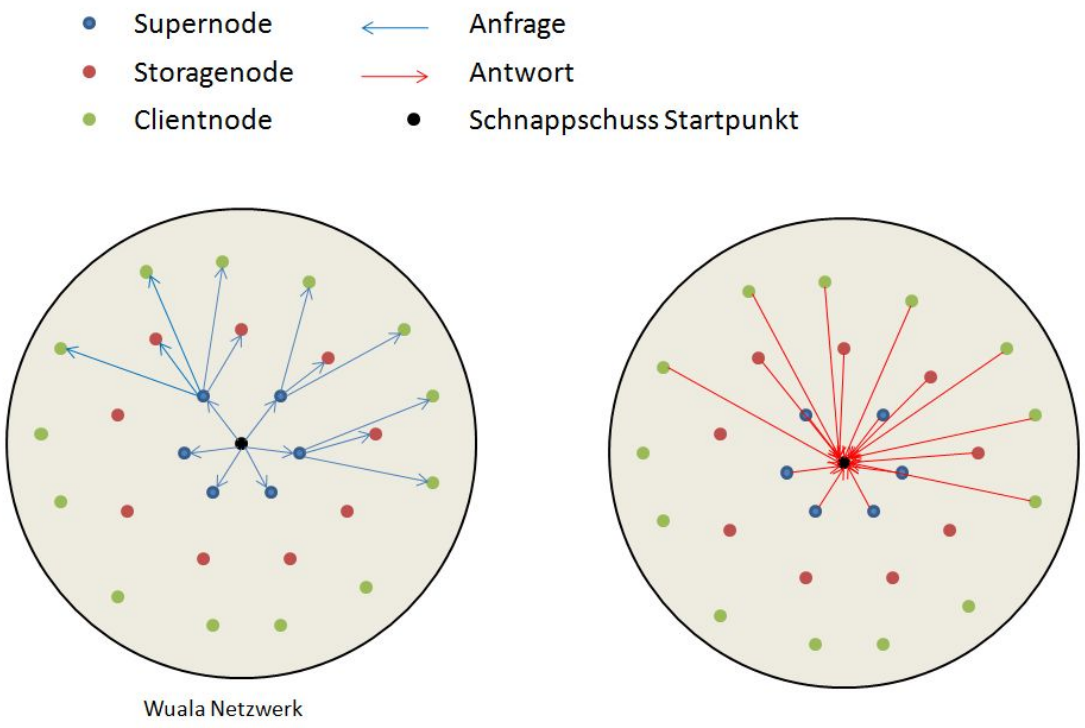

Abbildung 4.3: Antwortaggregation beim Zielknoten

einzelner Knoten reduziert wird. Andererseits verteilt dieser Scale-Out Ansatz die Aggregationslast auf mehrere Server und entlastet den Wurzelknoten. Es findet also eine Partitionierung des Gesamtsystems statt.

Dieser Ansatz wird im momentanen Wuala Monitoringsystem nicht eingesetzt. Er ist aber eine gute Alternative, falls das Wuala-Netzwerk in solche Dimensionen wachsen sollte, wo der Wurzelknoten mit der alleinigen Aggregation aller Antworten überlastet wäre.

4.5 Skalierbarkeit

Ein wichtiger Punkt, der beim Aufbau und Betrieb eines Monitoringsystems beachtet werden muss, ist die Skalierbarkeit des Systems. Wuala ist für ein Netzwerk mit mehreren 100'000, wenn nicht Millionen von Benutzern ausgelegt. Zwar sind solche Zahlen erst mittelfristig denkbar, doch muss bereits in einer frühen Phase des Monitoringsystems diese Problematik miteinbezogen werden. In diesem Abschnitt wird auf verschiedene Ansätze der Skalierbarkeit eingegangen, welche für das Wuala Monitoringsystem denkbar wären.

4.5.1 Ressourcenerweiterung

Der einfachste Ansatz, um immer grösser anfallende Datenmengen zu verarbeiten, ist die schrittweise Steigerung der zu Verfügung stehenden Ressourcen. Das heisst, jedes mal

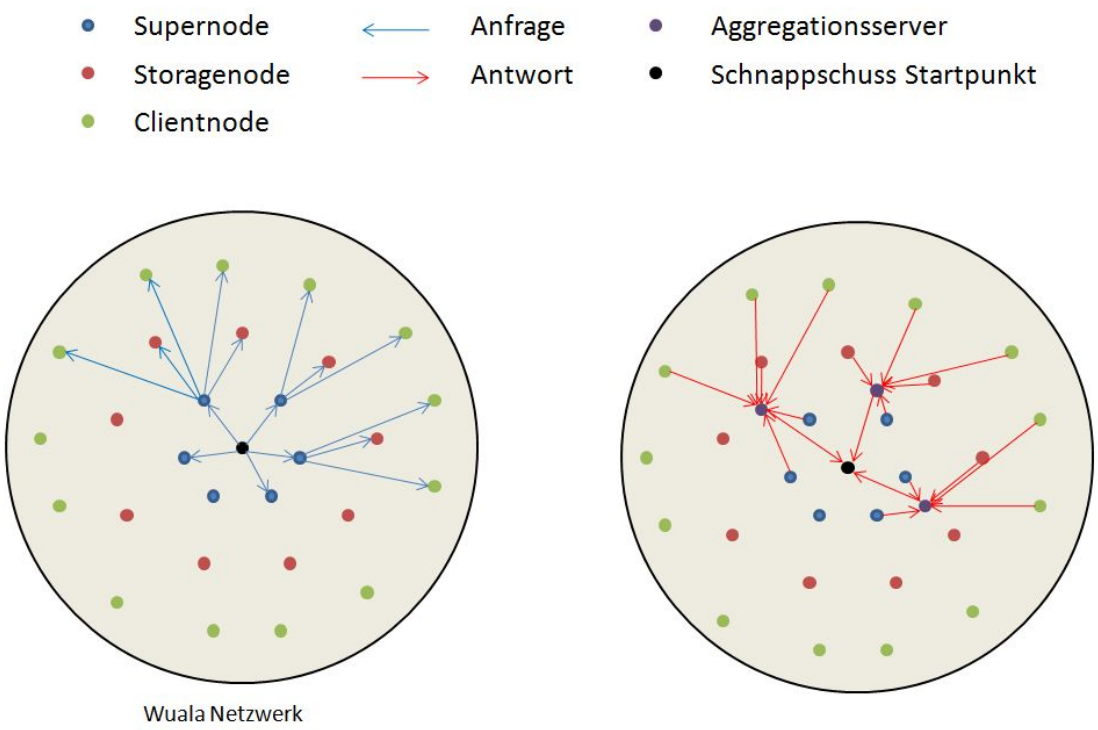

Abbildung 4.4: Antwortaggregation auf zusätzlichen Servern

wenn die Festplattenspeicher- oder die Bandbreitenkapazität des Monitoringservers beinahe ausgeschöpft sind oder ein vordefinierten Schwellenwert erreicht, werden die entsprechenden Ressourcen neu dazu gekauft und erweitert (Scale-Up). Dieser einfache Ansatz kann jedoch schon nach kurzer Betrachtung als unrealistisch ausgeschlossen werden. Die Kosten für die Anschaffung der Hardware sowie für die Bandbreitenerweiterung würden die zur Verfügung stehenden Mittel unnötig belasten und deshalb den Gesamtnutzen des Monitoringsystems mindern. Zusätzlich müssten die Datenbanken der verschiedenen Server aufeinander abgestimmt werden. Entweder funktionieren die einzelnen Server als Cluster einer einzelnen verteilten Datenbank (die Synchronisation der Daten wird dem Datenbanksystem überlassen) oder die Server verwalten jeweils nur einen disjunkt definierten Teilbereich des Netzwerkes. In diesem Fall muss die Zusammenfassung der Daten vom auswertenden Programm durchgeführt werden.

Ein reiner Ansatz der Ressourcenerweiterung ist aus Kostensicht ungeeignet und missachtet intelligente Möglichkeiten wie das Skalierungsproblem sonst noch zu bewältigen wäre.

4.5.2 Heuristische Auswahl der eintreffenden Daten

Ein weiterer Ansatz die steigenden Datenmengen in den Griff zu bekommen, ist die Aussortierung der von den Clients eintreffenden Daten beim Monitoringserver. Die einfachste Heuristik wäre, dass man nur einen gewissen Prozentsatz der eintreffenden Daten zur

internen Verarbeitung auf dem Monitoringserver weiterleitet. Die Filterwahrscheinlichkeit könnte als fixer Wert gesetzt oder von der Datenbank gesteuert werden: Wenn eine festgelegte Anzahl Daten pro Zeiteinheit überschritten wird, kann die Akzeptanzwahrscheinlichkeit gesenkt werden. So bleibt das Niveau der durchschnittlich gespeicherten Daten konstant.

Der Nachteil dieser Methode ist es, dass die Bandbreitenbeanspruchung nicht reduziert wird. Zwar kann die Datenbankverbindung und der Speicherplatz auf dem Server gesteuert werden, doch die Daten werden trotzdem von allen Clients zum Server versendet, wo sie dann erst behandelt oder verworfen werden können.

4.5.3 Heuristische Auswahl der zu sendenden Daten

Mit diesem Ansatz werden die auf den Knoten gesammelten Daten nur mit einer festgelegten Wahrscheinlichkeit an den Server gesendet. Somit wird, nebst der Speichereinsparung auf dem Server, auch noch die Netzlast reduziert. Dies ist der Vorteil gegenüber der Filterung der Daten auf dem Server. Ein aktiver Knoten, der mehr Reportdaten generiert, wird somit in den Enddaten trotzdem angemessen repräsentiert, da die Sendewahrscheinlichkeit pro Report und nicht pro Knoten definiert ist.

4.5.4 Aggregation der Daten auf dem Server

Eine weitere Möglichkeit die Datenmenge zu begrenzen, ist die Aggregation der Daten auf dem Server vor dem Abspeichern in die Datenbank. Dies kann geschehen, indem der Server mit dem Aggregations- und Speicherungsprozess eine gewisse Zeitspanne auf die spezifizierte Daten wartet oder aber, bis er eine festgeschriebene Anzahl der spezifizierten Daten erhalten hat. Ist der definierte Schwellenwert erreicht, so werden die eingegangenen Daten zu einem einzigen Gesamtereignis zusammengefasst, das anschliessend gespeichert wird.

Der Vorteil dieses Ansatzes ist die hohe Einsparung von Speicherplatz, wodurch das Monitoringsystem Daten über ein grösseres Netzwerk sammeln kann, bevor die Daten auf dem Monitoringserver weiter komprimiert werden, oder auf einen Archivspeicher ausgelagert werden müssen. Die Nachteile sind, dass die Enddaten in der Datenbank durch die Aggregation der Rohdaten vor dem Speichern an Aussagekraft verlieren. Sie bilden keine reale Repräsentation des Netzes, sondern nur noch ein in allen Eigenschaften möglichst detailgetreues Miniaturabbild des Gesamtsystems ab. Des Weiteren wird der Server zusätzlich beansprucht, da er nebst dem normalen Datenspeicherungsvorgang zusätzlich Rechenleistung für die Aggregation bereitstellen muss.

4.5.5 Wahl der Skalierung

Wie die Ausführungen über die verschiedenen Skalierungsansätze gezeigt haben, kann nicht ein einzelner Ansatz alle Kriterien und Wünsche eines effizienten und doch aussagekräftigen Monitoringsystems erfüllen. Vielmehr muss eine geschickte Kombination der

Möglichkeiten gefunden werden, so dass von der Wichtigkeit der Daten selbst auf die Art der Skalierung geschlossen werden kann. So ist es für Reports, die man auf den einzelnen Benutzer umrechnen will, völlig ausgeschlossen, dass diese mittels Heuristiken ausgedünnt werden.

Der Ansatz der reinen Ressourcenerweiterung ist plump und besitzt keinerlei technische Herausforderung, weshalb er nicht weiter verfolgt wurde.

Hingegen wurde eine Kombination der übrigen drei Ansätze ins Monitoringsystem aufgenommen. Es können die Sende- respektive die Speicherwahrscheinlichkeiten für die Daten festgelegt werden und es steht ein Mechanismus zur Datenaggregation auf dem Server zur Verfügung. Die genaue Umsetzung dieser Ansätze werden im Kapitel 5 dieser Arbeit genau beschrieben.

Ein wichtiger Aspekt, der beim Einsatz von Heuristiken bedacht werden muss, ist, dass die gesammelten Daten nicht mehr ein reales Abbild des Gesamtnetzes darstellen. Durch die Filterung wird nur noch eine Teilmenge der anfallenden Ereignisse gespeichert. Im Idealfall bildet diese Teilmenge das Gesamtnetz genau ab, im schlechtesten Fall weicht die Teilmenge stark vom Realzustand ab. Das Ziel beim Einsatz von Heuristiken muss es sein, einen möglichst optimalen Tradeoff zwischen Speicherverbrauch und Aussagekraft der Daten zu realisieren, so dass ein grosses Netzwerk mit kleinen Mengen von statistischen Daten analysiert werden kann.

4.6 Datenspeicherung

Die Wahl des Speichermediums ist eine zentrale Entscheidung bei der Konzeption des Monitoringsystems. Die Anforderungen sind, dass grosse Datenmengen bewältigt werden können, aber auch, dass die Analyse und Auswertung der Daten relativ einfach durchgeführt werden kann. Dieser Abschnitt stellt verschiedene Möglichkeiten mit ihren Vor- und Nachteilen vor und begründet die getroffene Wahl.

4.6.1 Berkeley Datenbank

Der potentielle Einsatz einer Berkeley Datenbank [21, 22] war ein nahe liegender Gedanke, da diese in vielen Bereichen der Wuala Applikation bereits benutzt wird. Die Berkeley Datenbank ist eine hoch performante, nicht relationale Datenbank. Sie verfügt über keine SQL Schnittstelle, sondern wird in die Benutzerapplikation eingebettet.

Die Benutzung der Berkeley Datenbank für das Monitoringsystem würde den Vorteil bringen, dass sie bereit in den Wuala Sourcecode integriert ist und eine einfache Verwendung garantiert werden kann. Das Fehlen einer SQL Schnittstelle ist der negative Aspekt, da dadurch die Auswertungs- und Anfragelogik neu implementiert werden muss.

4.6.2 Relationale Datenbanken

Relationale Datenbanken sind bekannt und geeignet für die Bewältigung grosser Datenmengen. Die meisten Hersteller bieten nebst den Standardlösungen noch Clustersysteme an. Mit deren Einsatz können die Datenmengen und Bandbreitenbelastung auf mehrere Teilsysteme aufgeteilt werden. Ebenso wird die Ausfallwahrscheinlichkeit des Gesamtsystems verringert. Die Einbindung der meisten relationalen Datenbanken in eine Javaapplikation ist dank den vorhandenen Treibern einfach umzusetzen.

Die Auswertung der Daten wird durch den Einsatz von SQL erheblich erleichtert. Für die meisten Produkte sind GUI-basierte Anfrageprogramme erhältlich, die die Ausführung von SQL Anfragen unterstützen und erleichtern.

4.6.2.1 Kommerzielle vs. Opensource Produkte

Im Bereich der relationalen Datenbanken sind nebst den bekannten kommerziellen Produkten wie Oracle [23], DB2 [24] oder MS SQL Server [25] auch Opensource Produkte erhältlich wie MySQL Community Server [26] oder PostgreSQL [27]. Mit den Opensource Produkten können die Lizenzkosten eingespart werden, jedoch bieten lizenzierte Produkte spezifischere Lösungen für spezialisierte Problemstellungen.

4.6.3 Dateisystem

Eine Möglichkeit grosse Mengen von Daten zu speichern, bietet ein Dateisystem. Ein für grosse Datenmengen skalierendes System ist das Google Filesystem [28] mit dem Einsatz von MapReduce [29]. MapReduce ist ein Framework, dass die parallele Verarbeitung grosser Datenmengen unterstützt, die im Google Filesystem gespeichert werden. Dies ist ein verteiltes Dateisystem, ausgerichtet auf die Handhabung grosser Daten und auf den Einsatz auch mit unzuverlässigen Teilsystemen. Hadoop [30] ist eine Opensource Implementierung, die auf den Spezifikationen von MapReduce und dem Google Filesystem basiert.

Der Vorteil des Einsatz eines verteilten Dateisystems, wäre das grundsätzlich endlos erweiterbare Speicherangebot und die redundanten Speichermechanismen zur Vermeidung von Datenverlust. Der Nachteil wäre, dass für ein Teilsystem ein grosser Implementierungsaufwand betrieben wird, für welchen einfache und stabile Lösungen vorhanden sind und, dass das Google Filesystem nur für grosse Dateien effizient ist.

4.6.4 Wahl des Speichersystems

Aufgrund der beschriebenen Vor- und Nachteilen der möglichen Speichervarianten, fiel die Wahl auf ein Opensource Produkt einer Relationalen Datenbank, konkret auf den MySQL Community Server. Die Implementierung eines eigenen Dateisystems wäre zu aufwändig gewesen und hätte einen produktiven Einsatz des Monitoringsystems zu weit

hinausgezögert. Die Berkeley Datenbank wurde nicht eingesetzt, da sich das Fehlen der SQL Schnittstelle negativ auf das Erstellen von Auswertungen auswirkt. Die geeignetste Lösung ist daher eine relationale Datenbank, da die Einbindung in eine Javaapplikation einfach zu machen ist und die Verwendung von SQL die Auswertung der gespeicherten Daten vereinfacht. Die Wahl eines Opensource Produkts wurde aus Kostenüberlegungen gefällt, da einerseits die Mittel von Caleido als Startup Unternehmen geschont werden und andererseits die Opensource Variante bezüglich Stabilität den Ansprüchen des Monitoringsystems genügt. Mit der Wahl von MySQL konnte eine zielgerichtete und flexible Umsetzung des Wuala Monitoringsystems realisiert werden.

Wenn in Zukunft das gewählte Speichermedium an seine Grenzen stossen sollte und auch der Einsatz von Clustern nicht genügend Freiraum schaffen kann, dann kann der Einsatz von Hadoop als Speicherkomponente in Betracht gezogen werden.

Kapitel 5

Design und Implementierung

Die Implementierung des Monitoringsystems für Wuala ist in zwei Hauptteile unterteilt. Der erste Teil ist das eigentliche Monitoringsystem, welches die Sammlung der interessanten Daten im Netz und deren zentrale Speicherung vornimmt. Der zweite Hauptteil der Implementierung beschreibt die Visualisierungskomponente des Monitoringsystems. Diese beiden Teile der Implementation sind total voneinander unabhängig. Das heisst, sowohl die Datenbeschaffung wie auch deren Darstellung kann grundsätzlich durch eine alternative Implementation ersetzt werden. Diese Möglichkeit scheint für die Datenbeschaffung wenig Sinn zu machen, jedoch ist die Auswechslung oder die Erweiterung der Darstellungs- und Auswertungskomponente für das gesamte Monitoringsystem gut denkbar.

Dieses Kapitel beschreibt auf den folgenden Seiten den Aufbau, die Hierarchien sowie die eingesetzten Implementierungstechniken für die beiden Teile des Monitoringsystems. Wichtige Teile des Systems werden zusätzlich mit UML Klassendiagrammen dargestellt. Diese sind nicht zwingend vollständig, sondern stellen nur jene Attribute, Methoden und Klassen dar, welche für das Verständnis der Funktionsweise von Nutzen sind. Ebenso werden Beispiele und eine konkrete Anleitung zur Benutzung des Systems gegeben, damit das Monitoringsystem mit Hilfe dieser Arbeit weiter sinnvoll eingesetzt werden kann.

Das Monitoringsystem ist vollständig in Java implementiert, da die Datenerfassung im Wuala Sourcecode geschieht und es somit vollständig in die Wuala Applikation integriert werden konnte.

5.1 Implementierung der Reporterfassung

Dieser Abschnitt wird zeigen, wie Reports im Wuala Monitoringsystem definiert und wie sie im Netzwerk versendet und schliesslich als Eintrag in eine Datenbanktabelle geschrieben werden. Zuerst wird der Aufbau eines Reports gezeigt, dann wie die Übermittlung im Wuala-Netzwerk funktioniert. Im Abschnitt 5.1.3 wird die Implementierung des Monitoringservers gezeigt, ins besonders mit welchen SQL Anweisungen die Javaobjekte in der Datenbank abgelegt werden.

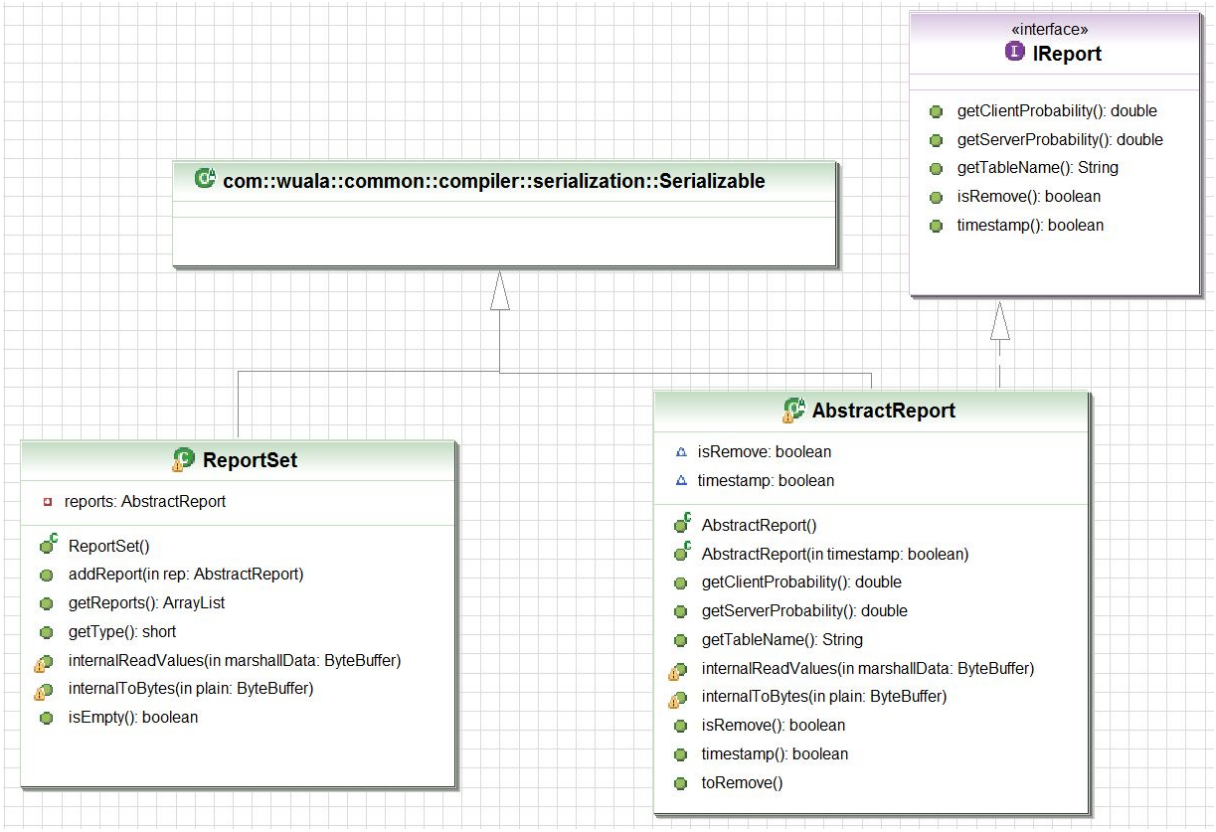

Abbildung 5.1: UML Diagramm eines Reports

5.1.1 Report

Die Hauptkomponente des Datensammlungsteils des Monitoringsystems ist der Report (Abbildung: 5.1). Für jedes Ereignis in Wuala, das überwacht werden soll, muss eine neue Javaklasse geschrieben werden, die von der Klasse `AbstractReport` erbt. Durch diesen Vorgang werden die Daten einerseits serialisierbar, da sie von der Klasse `Serializable` erben und können somit das Wuala Remote-Method-Invocation (RMI) Framework (Abschnitt: 5.1.2) benutzen und andererseits implementieren sie das Interface `IReport`, wodurch sie für den Monitoringserver gekennzeichnet werden.

Eine wichtige Rolle für die Kommunikation zwischen den einzelnen Reports und dem Monitoringserver spielt die Annotations-Klasse `@AMonitoringField` (Abbildung: 5.2). Dank den darin enthaltenen Methoden kann das Verhalten des Servers bezüglich der einzelnen Felder des Reports gesteuert werden.

Der Aufbau eines konkreten Reports ist somit simpel umzusetzen. Der konkrete Report erbt von der Klasse `AbstractReport`. Dann werden die Felder des neuen Reports definiert, wobei jedes Feld einer Spalte in der Datenbank entspricht. Dann werden die Klasse mit der Annotation `@ASerializable` und alle Felder mit `@ASerialize` ausgezeichnet, damit sie vom Wuala RMI Compiler erkannt werden.

Beispiel (zwei Felder werden definiert):

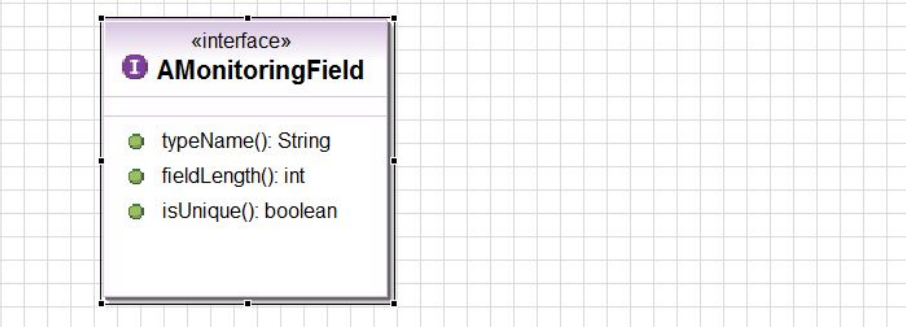

Abbildung 5.2: UML Diagramm der Annotationsklasse

```
@ASerializable
public class ConcreteReport extends AbstractReport {

    @ASerialize
    private String name;

    @ASerialize
    private int amount;
```

Nun muss für jedes Feld des Reports eine Getter-Method definiert werden, die denselben Namen trägt wie das Feld selbst, da der Datenbankspaltenname von dieser abgeleitet wird (Beispiel: Feldname: ”time”, Methodenname: ”getTime()”). Diese Getter-Methoden werden mit der `@AMonitoringField` Annotation ausgezeichnet. Das folgende Beispiel setzt den Typ ”String” fest mit der maximalen Länge 40 in der Datenbank und das Feld wird als Teil des eindeutigen Datenbankschlüssels definiert. Alle drei Attribute dieser Annotation sind fakultativ. Werden sie nicht explizit gesetzt, werden die vordefinierten Standardwerte benutzt (String, 50, false).

Beispiel:

```
@AMonitoringField (typeName = "String", fieldLength = 40, isUnique = true)
public String getName() {
    return name;
}
```

Die dritte obligatorische Komponente eines konkreten Reports ist der Konstruktor der Klasse. Dieser initialisiert standardmässig alle Felder des Reports. Wichtig ist jedoch der Aufruf des Superklassen-Konstruktors. Diesem wird ein boolescher Wert übergeben, der festlegt, ob in der Datenbank für diesen Report eine Spalte mit dem Einfügedatum generiert werden soll. Diese Konvention für den Konstruktoraufbau erleichtert später die Anwendung des Reports, vor allem wenn er an vielen Stellen im Code generiert werden soll.

Beispiel (Datumsspalte wird erstellt):

```
public ConcreteReport(String name, int amount) {
     super(true);
     this.name = name;
     this.amount = amount;
}
```

Mit diesen einfachen Schritten kann ein neuer Report definiert werden, der schlussendlich als Datenbanktabelle abgespeichert wird. Sollten in Zukunft Änderungen betreffend der Reportverarbeitung auf der Datenbankseite nötig sein, so kann die Annotation `@AMonitoringField` erweitert werden.

5.1.2 Wuala Remote Method Invocation (RMI)

Der Abschnitt 5.1.1 hat gezeigt, wie ein neuer Report aufgebaut ist. Diese Reports müssen nun ins Wuala Remote Methode Invocation Framework eingebaut werden, so dass sie vom Ort ihres Auftretens zum Monitoringserver übertragen werden können.

Jedes Objekt, das über das Wuala RMI Framework versendet wird, muss serialisierbar sein. Deshalb werden alle Reports und deren Felder mit den entsprechenden Annotations ausgezeichnet. Wird der Serialisierungscompiler ausgeführt, so generiert dieser für die gekennzeichneten Klassen die Methoden, welche für das Umwandeln der Feldwerte in einen Bytestrom und dessen Rückübersetzung zuständig sind. Die eine Methode wird für die Serialisierung vor dem Versenden an den Server benutzt, die andere zur Deserialisierung, wenn der Bytestrom vollständig beim Server angekommen ist. Die nun serialisierbaren Objekte können also vom RMI Framework benutzt werden.

Damit die Objekte korrekt deserialisiert werden können, muss für jedes Objekt eine eindeutige Typennummer festgelegt werden. Dies ist der Hauptunterschied zur Java Standard Serialisierung, wo die Auflösung über den Klassennamen gemacht wird. Der Nachteil bei dieser Methode ist jedoch, dass serialisierte Objekte nach Änderungen des Klassennamens nicht wieder deserialisiert werden können. Dies kann im Wuala RMI Framework nicht geschehen, da die Auflösung über die Typennummer geschieht.

Das Interface `IMonitoringServer` (Abbildung: 5.3) definiert die Methoden `sendReport()` und `sendAll()` zum Übertragen von einzelnen Reports und von ReportSets. Das Wuala RMI Framework benötigt nun für beide Methoden ein Dispatcherinterface, welches die Methode für die Antwortnachricht definieren.

Im Fall des Monitoringservers wird das Interface zweimal implementiert. Einerseits von der Klasse `MonitoringServer` im Clientcode, welche zusätzlich die Klasse `PingPongClient` erweitert. In dieser Klasse werden die IP-Adresse und die Portnummer des Servers festgehalten, an welche das zu sendende Objekt geroutet werden soll. Hier wird nicht der auszuführende Code implementiert, sondern festgehalten, wo der richtige Code gefunden werden kann (Stub). Andererseits wird das Interface auch im Code des Monitoringservers implementiert, ebenfalls in einer Klasse namens `MonitoringServer`. Diese enthält den realen Code, der ausgeführt wird, wenn die Methoden `sendReport()` oder `sendAll()` aufgerufen werden.

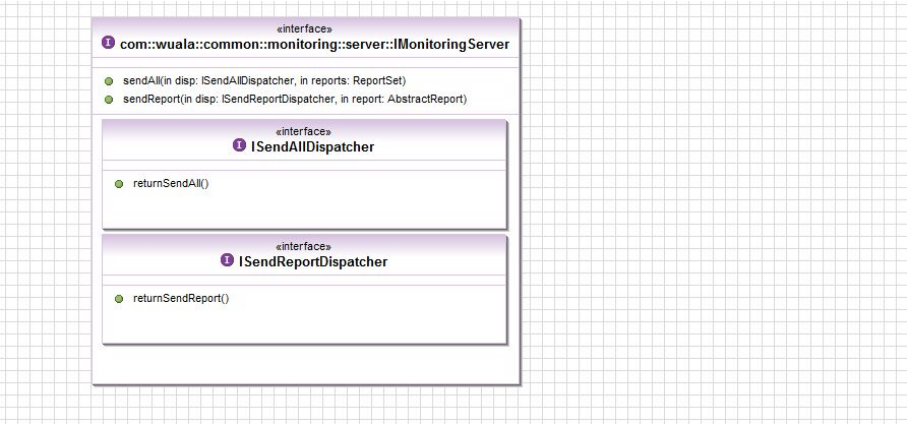

Abbildung 5.3: UML des IMonitoringServer Interfaces

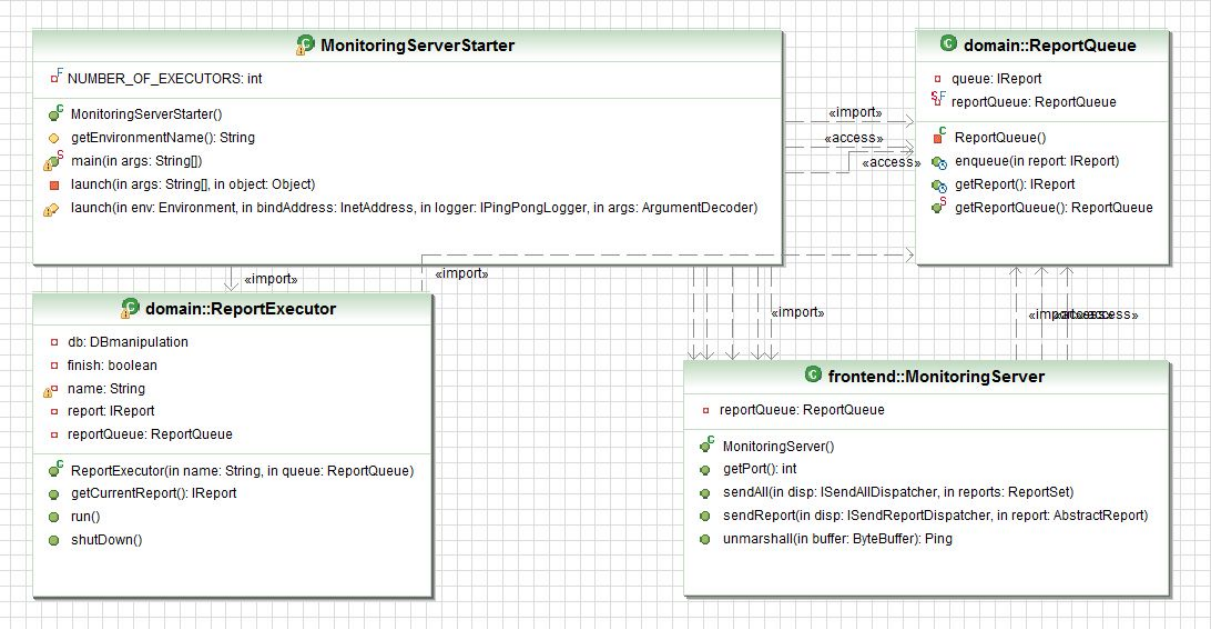

Abbildung 5.4: UML Diagramm des Monitoringservers

5.1.3 Monitoringserver

Bis jetzt wurde gezeigt, wie Reports erstellt werden können und Abschnitt 5.1.2 beschrieb, wie die Daten aus einem beliebigen Teil des Wuala-Netzwerks an den Monitoringserver gesendet werden können. Dieser Abschnitt beschreibt nun wie die eintreffenden Reports auf dem Server gehandhabt und schlussendlich in die Datenbank gespeichert werden.

Abbildung 5.4 zeigt die Starterklassen des Servers. Beim Start werden ein `ReportExecutor` und eine `ReportQueue` initialisiert. Die beim Monitoringserver eintreffenden Reports werden in die ReportQueue geschrieben. Jedes mal wenn dies geschieht, benachrichtigt die Queue den ReportExecutor. Dieser holt sich den zuerst eingefügten Report von der Queue und leitet ihn zur Ablegung in der Datenbank weiter. Momentan arbeitet das System nur mit einem Executor, der die Queue abarbeitet. Es ist aber so aufgebaut, dass das Framework relativ einfach erweitert werden kann, so dass mehrere Executor-Threads die eintref-

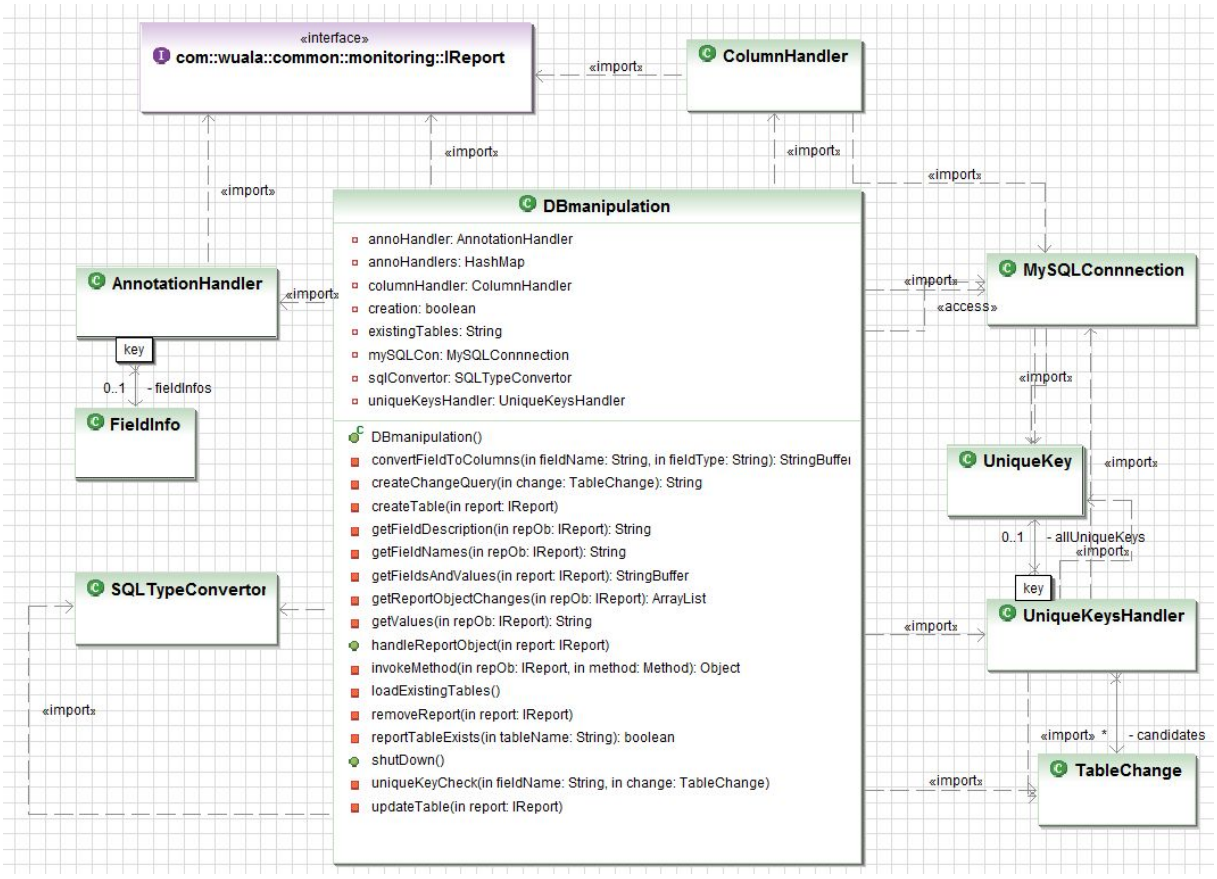

Abbildung 5.5: UML Diagramm der Reportspeicherung

fenden Reports parallel abarbeiten könnten. Für jeden einzelnen Executor-Thread könnte eine eigene Datenbankverarbeitung mit dazugehöriger Datenbankverbindung erstellt werden, so dass die Reportspeicherung parallelisiert werden kann. Die Locking-Mechanismen der einzelnen Tabellen würden dabei dem Datenbankmanagementsystem überlassen. Der Nachteil bei mehreren Executor-Thread ist der zusätzliche Synchronisationsaufwand damit die einzelnen Threads mit denselben Datenbankmetadaten arbeiten.

Der Executor übergibt die Reports mit dem Aufruf der Methode `handleReportObject()` an die Klasse `DBmanipulation`. Die Reports haben soeben das Zentrum der Monitoringsystems erreicht. Auf dieser Ebene werden die Reports analysiert und in die entsprechenden Datenbankeinträge umgewandelt. Abbildung 5.5 zeigt die `DBmanipulation`-Klasse und alle mit ihr interagierenden Klassen, die in den folgenden Zeilen betrachtet werden.

Am Anfang des gesamten Verarbeitungsprozesses steht die Verbindung zur Datenbank. Die auf dem Server verwendete Datenbank ist MySQL Version 5.0.32. Zur Herstellung der Verbindung wird das Apache Opensource Projekt DBCP (Database Connection Pooling) eingesetzt [31]. Mittels Connection Pooling wird verhindert, dass die Datenbankverbindungen bei jeder Anfrage neu geöffnet und danach wieder geschlossen werden müssen. Momentan wird jedoch beim Start des Monitoringservers nur eine einzelne Verbindung geöffnet und für die gesamte Betriebszeit offen gehalten. Das Connection Pooling bietet jedoch die Möglichkeit, dass beim Einsatz von mehreren ReportExecutors jeder einzelne eine eigene Datenbankverbindung erhalten kann. Für den Aufbau der Verbindung ist die Klas-

Tabelle 5.1: Entwicklung der Dauer einer Einfügeoperation

Bedingungen	# Reports	Zeit pro Report
SQL Verbindung wird jedes Mal neu aufgebaut	4000	17 ms
Apache Connection Pooling, Verbindung bleibt offen	15000	3,8 ms
Apache CP, offene Verbindung, DB-Metadaten in Cache	15000	0,2 ms

se `MySQLConnection` zuständig. Sie enthält die benötigten Datenbankinformationen, wie Datenbankname, Benutzername und Passwort. Mit dem Einsatz der offenen Verbindung und durch Caching von Metainformationen der Datenbank konnte der durchschnittliche Dauer für eine Einfügeoperation von anfänglich 17 Millisekunden auf 0.2 Millisekunden gesenkt werden (Tabelle: 5.1).

Der erste Schritt nach der Übergabe eines konkreten Reports an die Klasse `DBmanipulation` ist das Laden oder, wenn er noch nicht existiert, das Erstellen eines `AnnotationHandlers` für den entsprechenden Report Typ. Der AnnotationHandler iteriert über die Methoden des Reportobjekts und speichert all jene, die mit der `@AMonitoringField` Annotation gekennzeichnet sind. Nebst den annotierten Methoden werden ebenfalls die Werte der Annotationsattribute festgehalten. Der AnnotationHandler enthält also die gewünschte Datenbankbeschreibung für den neu eingetroffenen Report.

5.1.3.1 Reportspeicherung

Für die Abarbeitung des Reports können nun folgende Fälle unterschieden werden:

1. Die Datenbanktabelle für den Reporttypen existiert nicht.
2. Die Datenbanktabelle für den Reporttypen existiert.
   (a) Die Datenbanktabelle stimmt mit der Reportbeschreibung überein.
   (b) Die Datenbanktabelle unterscheidet sich von der Reportbeschreibung.
3. Der Report soll aus der Datenbank gelöscht werden.

Für den Fall 1 muss eine neue Tabelle in der Datenbank erstellt werden. Also wird eine MySQL CREATE Anweisung erstellt und ausgeführt. Der Name der neuen Datenbanktabelle entspricht demjenigen des Reports. Die Bezeichnung der einzelnen Spalten werden von den annotierten Methoden (verfügbar dank dem AnnotationHandler) abgeleitet. Ebenso sind die Werte für die Feldlänge und den Feldtypen vom AnnotationHandler abrufbar. Da die Datentypen von Java und MySQL nicht immer übereinstimmen, müssen sie entsprechend konvertiert werden. Die gültigen Java Datentypen und die dazugehörigen MySQL Entsprechungen werden in der Klasse `SQLTypeConvertor` festgehalten, wo sie bei Bedarf noch erweitert werden können. Für unseren Beispielreport würde die automatisch generierte CREATE-Anweisung wie folgt aussehen:

```
CREATE TABLE ConcreteReport
(timestamp TIMESTAMP, name VARCHAR(40),
amount INT, UNIQUE INDEX uniqueIndex_name (name))
```

Anschliessend an das erfolgreiche Erstellen der Tabelle werden die Werte des Reports in der Datenbank gespeichert (Fall 2a).

Für den Fall 2a existiert die Tabelle bereits. Jetzt muss überprüft werden, ob die Datenbanktabellenbeschreibung des eingetroffenen Reports (verfügbar durch den AnnotationHandler) mit der sich in der Datenbank befindlichen Tabelle übereinstimmt. Hierzu wird die Klasse `ColumnHandler` für den Report initialisiert. Diese liest die Metadaten der angeforderten Tabelle aus der Datenbank. Stimmen die ermittelten Werte (die konkreten Datenbankbeschreibung) mit jenen des AnnotationHandler (die Beschreibung der Datenbank vom Report) überein, so kann die generierte SQL Anweisung abgesetzt werden.

```
REPLACE INTO ConcreteReport (name, amount) VALUES ("DummyUser", 123)
```

Wie man sieht, wird keine INSERT-Anweisung abgesetzt, wie es eigentlich zu erwarten gewesen wäre, sondern das MySQL spezifische REPLACE. Wieso dies so gewählt wurde wird, im Abschnitt 5.1.3.2 beschrieben.

Nun kommen wir zum Fall 2b, der besagt, dass die Tabellenbeschreibung des neuen Reports nicht mit der bereits bestehenden Datenbanktabelle übereinstimmt. Es gibt drei Arten von Änderungen, die vom Monitoringserversystem automatisch erkannt und ausgeführt werden können. Erstens werden Änderungen der Datentypen behandelt (z.B.: ”integer” wird in ”long” geändert), zweitens werden neue Felder im Report erkannt und drittens werden nicht mehr benutzte Datenbankspalten bemerkt. Für jede Änderung wird ein `TableChange` Objekt generiert. Nachdem alle Änderungen gefunden wurden, wird pro TableChange Objekt eine SQL Anweisung abgesetzt. Wenn Änderungen an einem sich bereits im Einsatz befindlichen Report durchgeführt werden, müssen einige Regeln beachtet werden. Werden die Datentypen geändert, so muss der neue Typ so gewählt sein, dass die bereits in der Datenbank gespeicherten Werte in den neuen Typen überführt werden können (z.B. ”integer” zu ”long”). Ist dies nicht der Fall, wird die Änderung von der MySQL-Datenbank zurückgewiesen und der neue Report kann nicht gespeichert werden. Des Weiteren ist es nicht möglich einen Feldnamen nachträglich zu ändern und die bereits gespeicherten Daten beizubehalten. Diese Änderung wird vom Monitoringserversystem als Einfügen eines Feldes und als Löschen eines anderen Feldes interpretiert, wobei die bisherigen Daten mit dem nun überflüssigen Feld zusammen gelöscht werden. Das folgende Beispiel zeigt eine zulässige Änderung des Java-Datentyps von ”integer” zu ”long” (MySQL intern als BIGINT-Datentyp gespeichert):

```
ALTER TABLE ConcreteReport MODIFY COLUMN amount BIGINT
```

Der Fall 3 wird benötigt, wenn man in einer Datenbanktabelle ein aktuelles Abbild des Netzwerkes haben will, das heisst es müssen nicht mehr gültige Einträge aus der Datenbank entfernt werden. Dies geschieht, indem im Report ein alternativer Konstruktor implementiert wird, bei welchem nur der eindeutige Schlüssel des Reports übergeben wird.

Im Konstruktor wird der Report zusätzlich mit dem Aufruf der Methode `toRemove()` gekennzeichnet. Jeder eintreffende Report wird zuerst auf dieses Flag überprüft und, falls es gesetzt ist, wird der Eintrag mit dem entsprechenden eindeutigen Schlüssel gelöscht.

```
DELETE FROM ConcreteReport WHERE name = "DummyUser"
```

5.1.3.2 Eindeutige Schlüssel

Auf den letzten Zeilen wurden die eindeutigen Schlüssel oft erwähnt. Auf jeder Datenbanktabelle kann ein solcher eindeutiger Schlüssel definiert werden, der aus einem, mehreren oder allen Feldern des Reports besteht. Die Funktion dieser Schlüssel ist es, dass die Werte der gewählten Felderkombination nur einmal in der Datenbank gespeichert werden. Die Organisation der eindeutigen Schlüssel übernimmt die Klasse `UniqueKeyHandler`. Wenn im Fall 2b Änderungen im Aufbau des Reports festgestellt werden, wird überprüft, ob ein Feld involviert ist, das mit dem `@AMonitoringField`- Attribute `isUnique()` ausgezeichnet ist. Wenn dem so ist, wird das aktive TableChange Objekt dem UniqueKeyHandler übergeben. Dieser überprüft nun, ob diese Änderung tatsächlich den eindeutigen Schlüssel des Reports betrifft. Dazu werden, analog zum ColumnHandler, über die Metadaten der Datenbank der bestehende eindeutige Schlüssel der betreffenden Tabelle ausgelesen und mit den neuen Informationen abgeglichen. Wird eine Änderung entdeckt, wird die entsprechende SQL Anweisung ausgeführt. Das Beispiel zeigt den Fall, dass der eindeutige Schlüssel um das Feld "`amount`" erweitert wird.

```
ALTER TABLE ConcreteReport DROP INDEX uniqueIndex_name,
ADD UNIQUE INDEX uniqueIndex_name (name,amount)
```

An dieser Stelle kann nun auch erklärt werden, wieso bei einer einfachen Einfügeoperation keine INSERT Anweisung, sondern eine REPLACE Anweisung ausgeführt wird. Der Vorteil hierbei ist es, dass die REPLACE Anweisung die eindeutigen Schlüssel berücksichtigt. Ist kein solcher vorhanden, werden die neuen Werte als einfache INSERT Anweisung ausgeführt, wenn doch, dann wird der alte Eintrag gelöscht, bevor der neue eingefügt wird [32]. Deshalb wird die REPLACE Anweisung für alle Einfügeoperationen benutzt.

5.1.3.3 Implementierung von Heuristiken

Das Wuala Monitoringsystem unterstützt zwei verschiedene Arten von Heuristiken. Das heisst, es können prozentuale Werte gesetzt werden, wie oft das Versenden eines Reports oder die Speicherung in die Datenbank ausgeführt wird. Der Sinn dieser Heuristiken ist es, dass der Entwickler die Anzahl der eintreffenden und versendeten Reports steuern kann, damit das Monitoringsystem auch beim vielen Wuala-Benutzern noch effektiv eingesetzt werden kann. Wenn die Wahrscheinlichkeiten bekannt sind, kann auch aus der erfassten Teilmenge auf die Gesamtheit des Systems geschlossen werden. Zum Beispiel könnten Aufrufe des Wuala-Weltbereichs und des Wuala-Gruppenbereichs jeweils mit zehnprozentiger Wahrscheinlichkeit erfasst werden. Wenn nun der Weltbereich 100 Einträge und der Gruppenbereich 20 Einträge in der Datenbank haben, dann kann daraus geschlossen werden,

dass der Weltbereich 1000 mal und der Gruppenbereich 200 mal besucht worden, was den Weltbereich um Faktor fünf attraktiver für potentielle Einblendungen macht.

In der Klasse *SendProbabilityChecker*, die sich im allgemeinen Wuala Clientcode befindet, wird überprüft, ob die Wahrscheinlichkeit, dass der Report versendet werden soll kleiner als einhundert Prozent ist. Dieser Standardwert kann bei jedem Auftreten eines Reports separat gesetzt werden oder aber auch gleich für alle Instanzen eines Reports gemeinsam definiert werden. Der Versand über das Wuala RMI Framework wird nur mit der angegebenen Wahrscheinlichkeit durchgeführt.

Eine weitere Wahrscheinlichkeitsprüfung wird nach der Übermittlung an den Monitoringserver durchgeführt. Die Klasse `EnqueueProbabilityChecker` prüft, ob der eingetroffene Report zur Weiterverarbeitung in die Queue gestellt werden soll. Der Wahrscheinlichkeitswert wird ebenfalls aus dem Report selber ausgelesen. Im Gegensatz zur Versendungwahrscheinlichkeit kann die Verarbeitungswahrscheinlichkeit nur global für alle Instanzen eines Reports gesetzt werden. Auch ist der Verarbeitungsansatz für die Reports auf dem Monitoringserver anders gewählt: Ist der Wert auf 20 Prozent gesetzt, dann wird genau jeder fünfte Report dieses Typen weiterverarbeiten. Dies im Unterschied zur Versendungwahrscheinlichkeit, wo jeder Report, unabhängig von der Reihenfolge, versendet werden kann.

Die beiden Wahrscheinlichkeitswerte können auch kombiniert werden, so dass die Reports eine doppelte Filterung, einmal auf dem Client und einmal auf dem Monitoringserver, durchlaufen.

5.2 Die Arten der Reporterfassung

Das Wuala Monitoringsystem bietet zwei verschiedene Ansätze, wie Ereignisse festgehalten werden. Erstens können Reports überall im Wuala-Netz ausgelöst werden. Dies kann auf in der Clientapplikation oder auf einem der Wuala Server geschehen. Die zweite Möglichkeit an interessante Daten zu gelangen, ist das Erfassen eines Schnappschusses (Snapshot) über alle Knoten, die im Moment der Durchführung online sind.

5.2.1 Standard-Reporterfassung

Ist ein Report, wie in Abschnitt 5.1.1 beschrieben, erstellt worden, dann kann der Konstruktor diese Reports überall im Sourcecode des Wuala-Clients oder eines Servers aufgerufen werden, um eine neue Instanz zu erzeugen. Sind die relevanten Werte des Reports initialisiert, so ist der Report zum Versand an den Monitoringserver bereit. Abbildung 5.6 zeigt die Klasse `Monitoring`, welche das Versenden der Reports durch den Aufruf von statischen Methoden ermöglicht.

```
ConcreteReport conRep = new ConcreteReport("DummyUser", 123);
Monitoring.report(conRep);
```

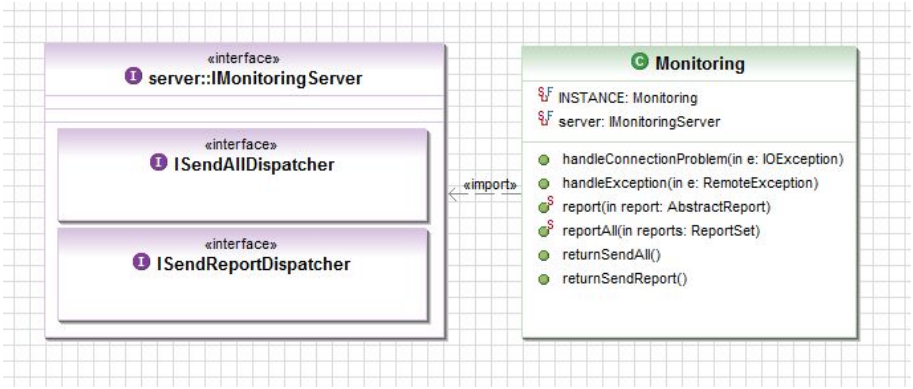

Abbildung 5.6: UML Diagramm der Monitoringklasse

Das Beispiel zeigt, wie einfach ein Report im Code generiert und versendet werden kann. Die Klasse Monitoring bietet somit eine einfache Standardlösung, die für den Versand der meisten Reports genügt. Werden aber spezielle Anforderungen an das Versenden gestellt, so können zusätzliche Optionen implementiert werden, indem der Dispatcher für die benötigte Versandmethode überschrieben wird. So kann das Timing des Verschickens variiert werden oder aber der Report wird bei Verbindungsproblemen mit dem Monitoringserver nochmals versendet, bis die Übermittlung funktioniert hat.

5.2.2 Sammeln von Schnappschussdaten

Die hier vorgestellte Implementierung ist die Umsetzung des in Abschnitt 4.4.2 vorgestellten Ansatzes. Abbildung 5.7 gibt einen Überblick über die wichtigsten involvierten Klassen und zeigt, wie sie interagieren. Der Wurzelknoten, der den Start und Zielpunkt der Sammlung des Schnappschusses ist, ist eine Instanz der Klasse `NodeReportCollector`. Diese holt sich vom Wuala `Machineserver` eine Liste mit allen Supernodes, die momentan online sind. Nun werden die Schnappschussanfragen (ererbte Instanzen der Klasse `NodeReport`) an alle Supernodes versendet, welche dieselbe an all ihr bekannten Client- und Storagenodes weiterleitet. Es ist zusätzlich möglich, die Tiefe des Versendens für jeden Schnappschuss einzeln zu definieren, so dass zum Beispiel nur alle Supernodes oder alle Knoten im System angesprochen werden können. Die Übermittlung der NodeReports geschieht, wie bei einem normalen Report, über das Wuala RMI Framework (Abschnitt 5.1.2).

Die verschiedenen Arten von NodeReports werden in der Klasse `ReportTimer` festgelegt. Jedem NodeReport wird ein separates Zeitintervall zugeordnet, welches festlegt in welchem Abstand der Schnappschuss durchgeführt werden soll. Somit kann die Veränderung des Netzwerks über die Zeit festgehalten werden. Eine weitere Funktion des ReportTimers ist die Koordination der Ausführungszeiten der verschiedenen NodeReport Typen. So wird im Beispiel der konkrete NodeReport alle 30 Minuten ins Netzwerk verschickt.

```
addReportTime(INodeReportTypes.CONCRETE_NODE_REPORT, HALF_AN_HOUR);
```

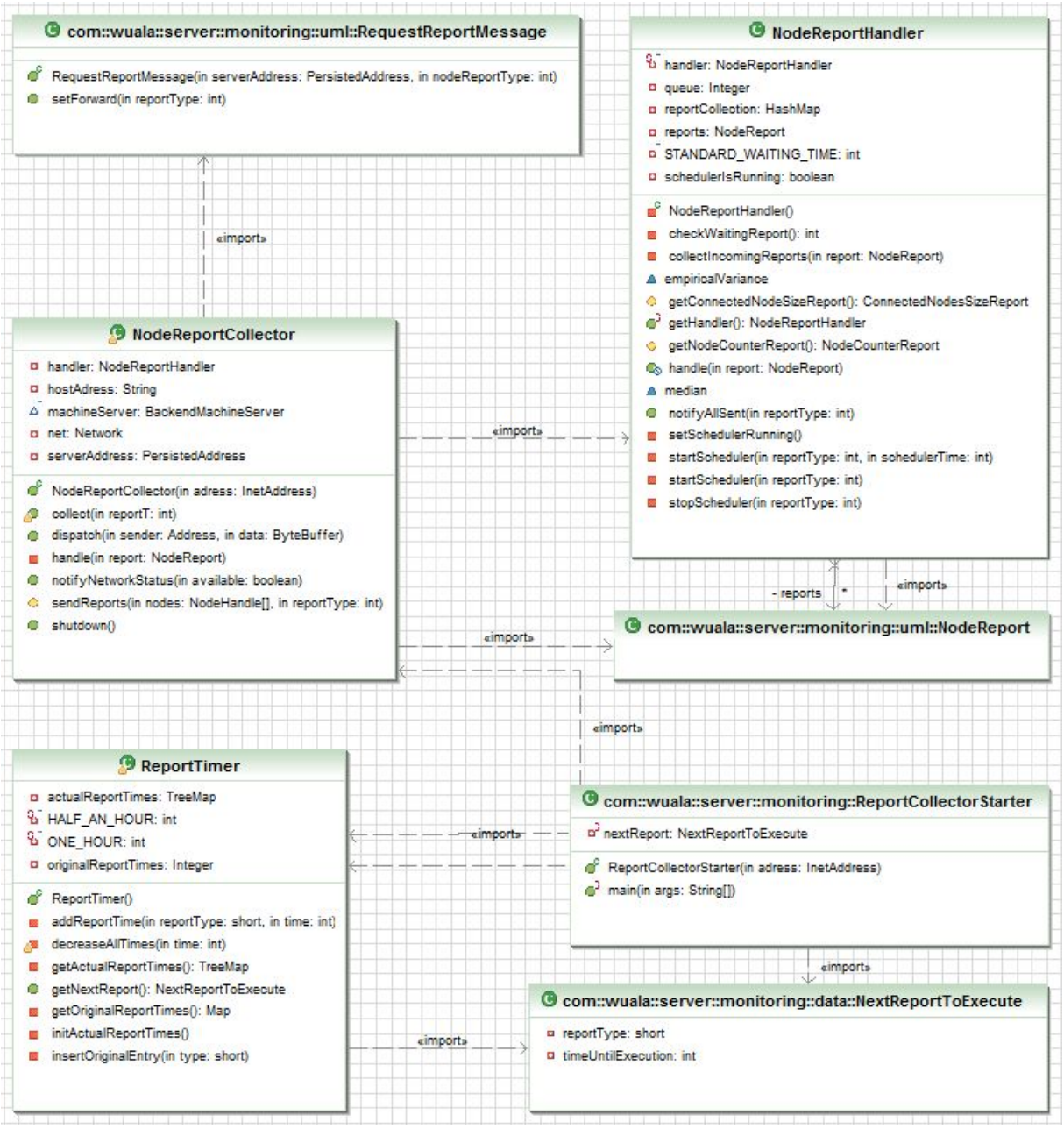

Abbildung 5.7: UML Diagramm des NodeReportCollectors

Damit das Netz durch die Schnappschüsse nicht übermässig belastet wird, übergibt der ReportTimer jeweils nur einen einzelnen NodeReport an den NodeReportCollector. Erst wenn dieser die Rückmeldung bekommt, dass die zurückgekommenen Werte abgearbeitet worden sind, darf der nächste Report in der Schlange ausgeführt werden.

Da die Anzahl zurückkehrender Antworten nicht mit der Anzahl Anfragen übereinstimmen muss, da einzelne Knoten in der Zwischenzeit offline gegangen sein könnten, ist ein einfacher Mechanismus implementiert, der garantiert, dass alle regulären Antworten auch in das Endresultat einfliessen. Sobald der ReportCollector mit dem Versenden eines NodeReport-Typs beginnt, können die eintreffenden Antworten vom `NodeReportHandler` gesammelt werden. Hat der ReportCollector alle NodeReports an alle bekannten Supernodes erfolgreich versendet, benachrichtigt er nun den ReportHandler. Dieser wartet zusätzlich eine festgelegte Zeit lang auf weitere eintreffende NodeReports dieses Typs, dann beginnt er mit der Aggregation der Ergebnisse. Alle verspätet eintreffenden Werte werden nicht mehr berücksichtigt und weggeworfen. Die Sammlung für diesen Typen beginnt erst wieder neu, wenn der ReportCollector einen neuen Versandzyklus beginnt.

Die Aggregationslogik muss für jeden NodeReport-Typen im NodeReportHandler separat implementiert werden. Damit das Ergebnis eines Schnappschusses gleich wie ein unaggregiertes Ereignis in die Datenbank geschrieben wird, muss für jeden NodeReport ein dazugehöriger Report (5.1.1) erstellt werden. Jedes Feld des neuen Report enthält einen aggregierten Wert der Schnappschusserfassung. Der NodeReportHandler benutzt die Standard-Reporterfassung (5.2.1) zum Versenden des Reports.

Grundsätzlich kann der Schnappschuss von jedem Knoten im Netzwerk ausgelöst werden. Aus praktischen Gründen wird der `ReportCollectorStarter` gleich auf dem Monitoringserver selbst ausgeführt, da der Server dazu geeignet ist höhere Lasten zu verarbeiten als ein beliebiger Knoten im Netzwerk.

5.3 Implementierung der Monitoringsystemvisualisierung

Teile der Visualisierungskomponente des Monitoringsystems basieren auf der Opensource Programmbibliothek Prefuse [33, 34]. Prefuse ist ein Toolkit, das verschiedene Werkzeuge zur schnellen Implementierung von anspruchsvollen und interaktiven Visualisierungen enthält. Im Rahmen dieser Diplomarbeit wurden zwei verschiedene Darstellungsmodule umgesetzt, deren Inputdaten auf einfache Weise ausgetauscht und variiert werden können. Die dritte Komponente ist ein Statistikmonitor, mit welchem mehrere Auswertungen der Datenbankdaten gleichzeitig und aktuell dargestellt werden.

5.3.1 Prefuse Visualisierungen

5.3.1.1 Funktionsweise von Prefuse

Das Prefuse Visualisierung Toolkit funktioniert nach dem Prinzip, dass die darzustellenden Daten in eine interne Prefuse Tabellenstruktur geladen werden. Für jeden darzustellenden Wert wird ein `VisualItem` generiert, welche von verschiedenen Renderern gezeichnet werden. Damit entschieden werden kann, welcher Renderer für welche Visual Items zuständig sind, werden Prädikate definiert, die die Zugehörigkeit von Renderer und einer Gruppe von Visual Items festlegen. Ein Kriterium, das besonders für den Einsatz von Prefuse sprach, ist die Möglichkeit, dass die darzustellenden Daten mit einer SQL Anfrage direkt in eine Prefuse Tabelle geladen werden können. Als anderes Eingabeformat sind XML Dokumente möglich, welche die Datenbeschreibung enthalten.

Die nachfolgend vorgestellten Visualisierungen basieren auf Beispielanwendungen, die das Standardrelease des Prefuse Visualisierung Toolkits beinhaltet.

5.3.1.2 Graphen-Visualisierung

Abbildung 5.8 zeigt das UML-Klassenmodell der Graphen-Visualisierung. Die Klasse `RadialGraphView` kann Graphen darstellen, die im GraphML Format vorliegen. Das GraphML Dateiformat [35] ist eine XML basierte Strukturbeschreibung für Graphen. Es ist in die Beschreibung von Knoten und Kanten des Graphen gegliedert. Die Knoten bestehen aus dem darzustellenden Namen und einer eindeutigen Id, jede Kante wird durch zwei Ids definiert.

Um die in der SQL Datenbank gespeicherten Daten als Graph darstellen zu können, müssen sie in das erwähnte GraphML Format konvertiert werden. Deshalb erbt jeder konkrete Graph von der Klasse `AbstractGraphMLGenerator`, welche die Methoden zum generieren der XML Datei enthält. Im konkreten Graph werden die SQL Anfragen für die Knoten und die Kanten definiert. Die Klasse `DBConnection` stellt die Verbindung zur Datenbank her und liefert die gewünschten Daten zurück. Alle hier beschriebenen Visualisierungen nutzen diese Klasse zum Datenbankzugriff. Die SQL Antworten werden in eine XML Datei geschrieben, welche nur temporär existiert, bis die Visualisierung wieder geschlossen wird.

Für eine neue Graphen Visualisierung muss jeweils der gesamte XML Datei Generator neu definiert werden, da die Verarbeitung der SQL Anfragen für Knoten und Kanten für jeden Anwendungsfall stark variieren kann. Ausserdem können bei jeder Darstellung andere Sonderfälle auftreten, welche in die XML Datei hineingeschrieben werden müssen. Deshalb ist die Erstellung eines Graphen weniger generisch wie die Visualisierung 5.3.1.3. Insbesonders können Sonderfälle auftreten, wenn die benötigten Knoten oder Kanten nicht mit einer einzelnen SQL Anfrage zusammengefasst werden können, sondern aus unterschiedlichen Datenbankabfragen zusammengesetzt werden müssen.

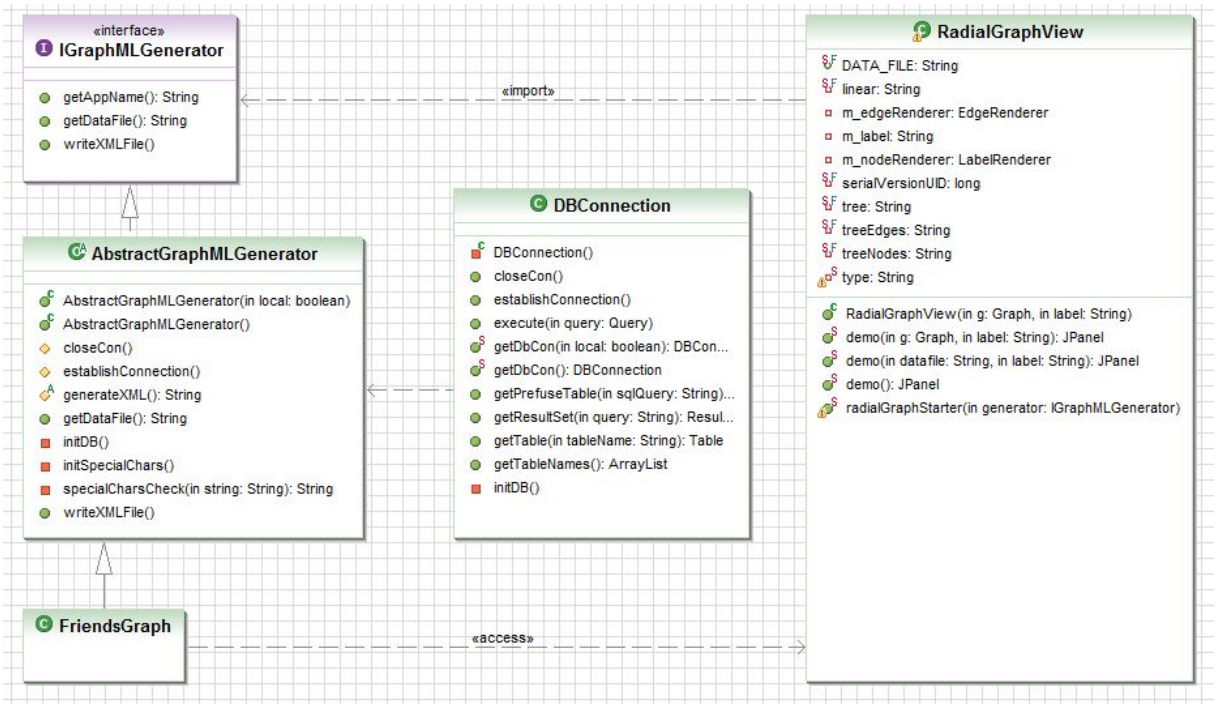

Abbildung 5.8: UML Diagramm der Graphen Visualisierung

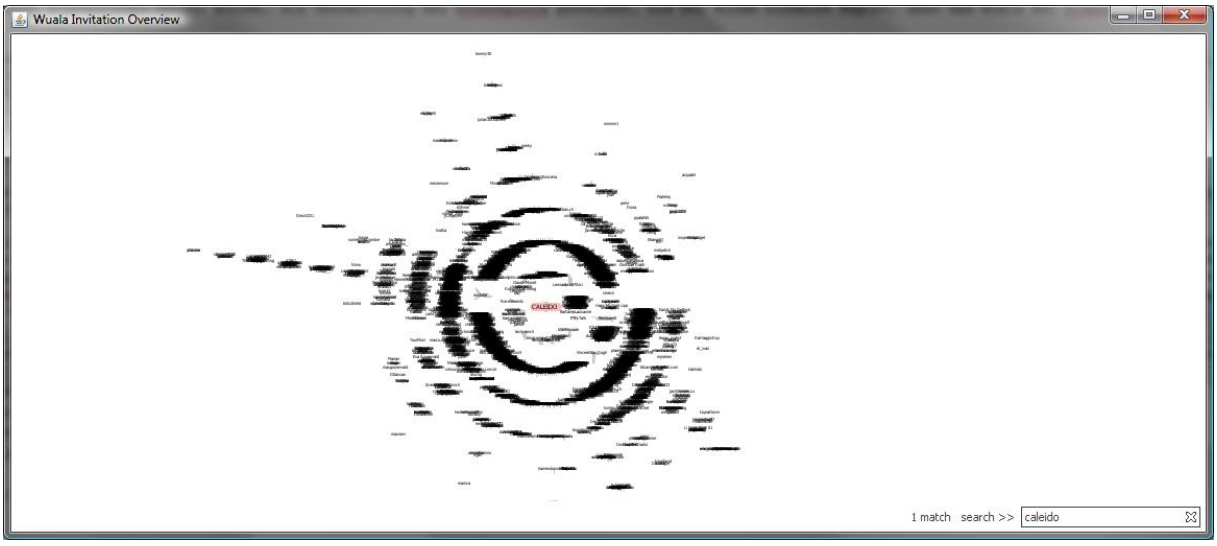

Abbildung 5.9: Beispiel einer Graphen Visualisierung

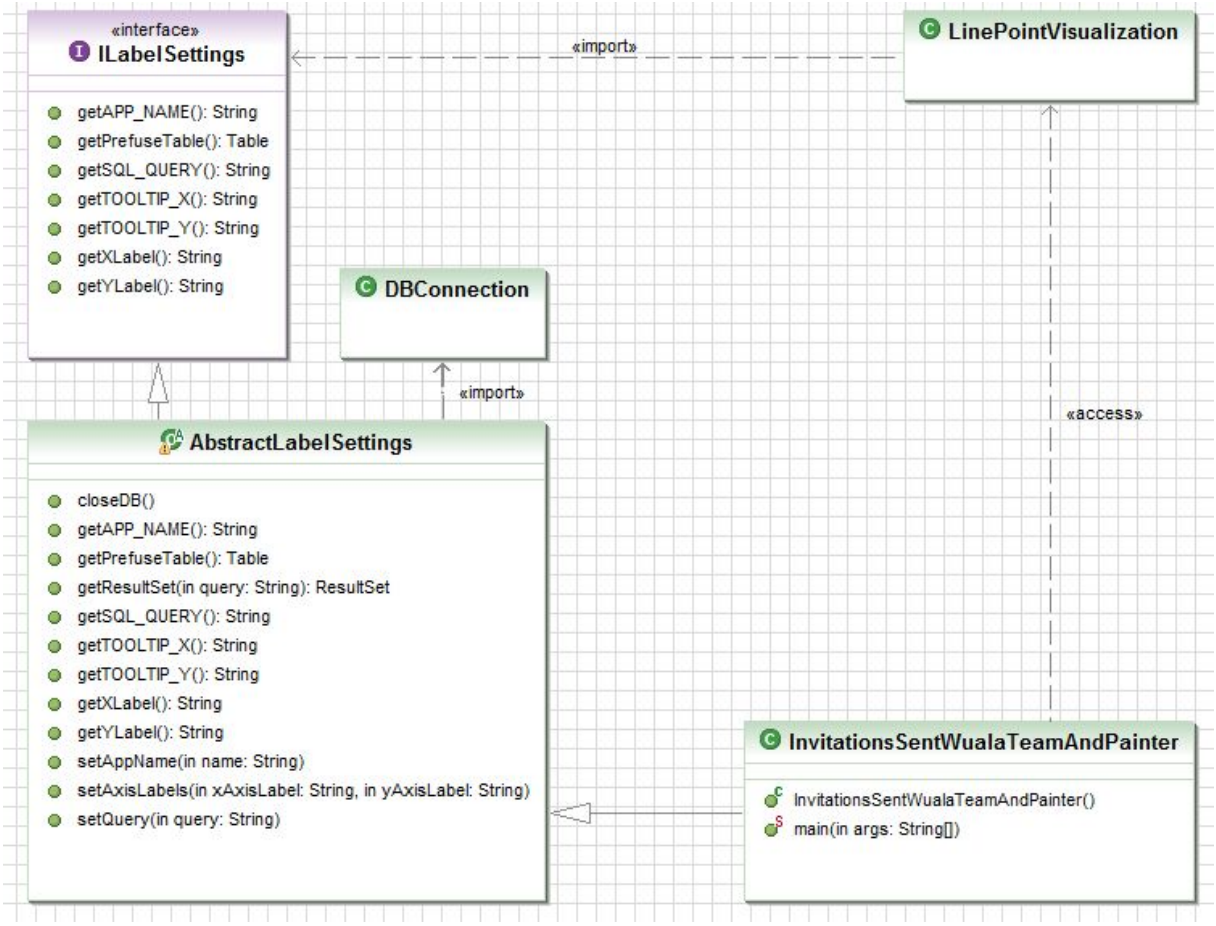

Abbildung 5.10: UML Diagramm der Linien und Punkte Visualisierung

5.3.1.3 Linien und Punkte-Visualisierung

Mit der Linien und Punkte-Visualisierung können Daten in einem zweidimensionalen Koordinatensystem wahlweise als Punktwolke oder als Liniendarstellung visualisiert werden. Für diese Visualisierung wird das direkte Importieren der Daten aus einer SQL Datenbank eingesetzt. Dazu muss die Datenbankabfrage so aufgebaut sein, dass sie ein Resultat mit zwei Spalten zurück liefert. Die erste Spalte definiert den Wert für die Y-Achse und die zweite den Wert für die X-Achse.

Für jede konkrete Instanz dieser Visualisierung wird eine Klasse erstellt, die von der Klasse `AbstractLabelSettings` (Abbildung: 5.10) erbt. Im Konstruktor der konkreten Klasse müssen nur noch die Datenbankanfrage, sowie die Beschriftungen für die Achsen und den Tooltip gesetzt werden. Diese Klasse enthält sozusagen die Metadaten der Visualisierung und kann zur eigentlichen Darstellung an die Prefuseklasse `LinePointVisualisation` übergeben werden. In ihr werden vier verschiedene Renderer eingesetzt; ein Renderer zeichnet die Linien, der zweite die Punkte und die anderen beiden je eine Achse mit den dazugehörigen Beschriftungen. Wenn die Punktwolkendarstellung aktiviert ist, kann für jeden Punkt eine Tooltipausgabe betrachtet werden, welche das Wertepaar des Punktes anzeigt (Abbildung: 5.11).

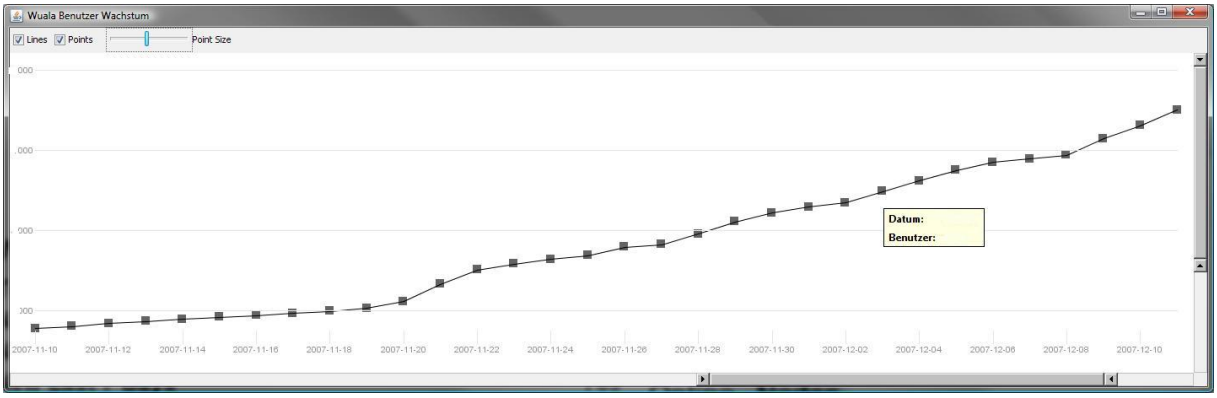

Abbildung 5.11: Beispiel einer Linien und Punkte Visualisierung (Daten zensiert)

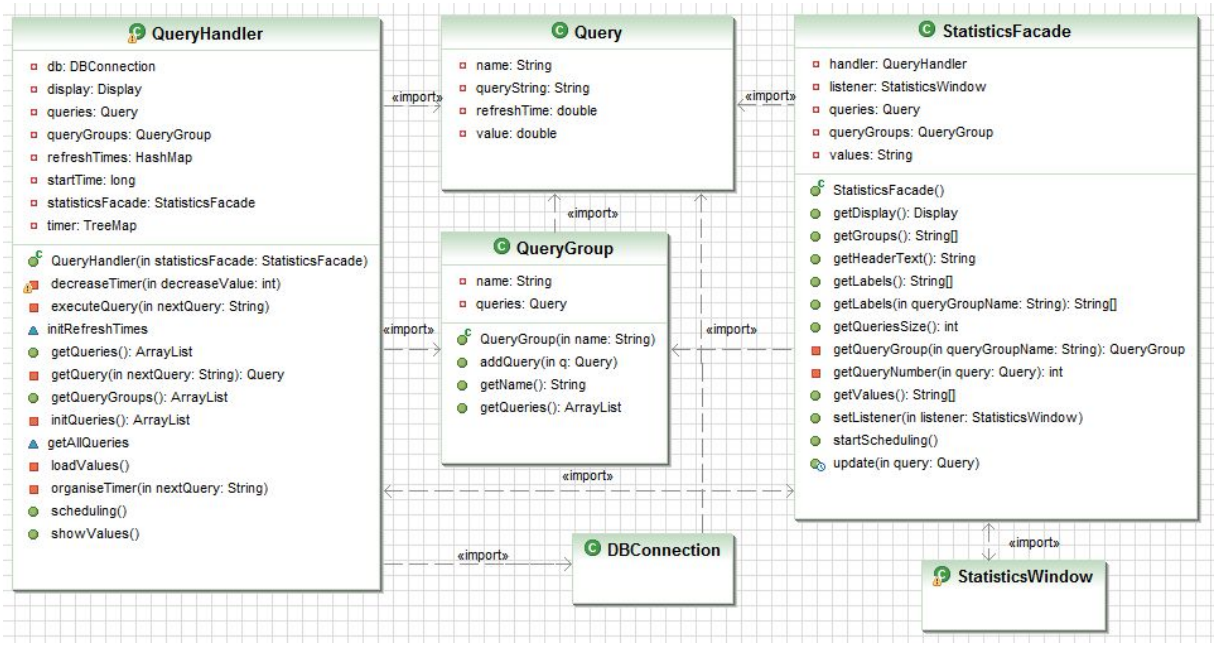

Abbildung 5.12: UML Diagramm des Statistics Monitors

5.3.2 Wuala Statistics Monitor

Die dritte Darstellungsvariante für die mit dem Wuala Monitoringsystem erfassten Daten ist der Wuala Statistics Monitor. Er ist eine Art Live-Darstellung vieler verschiedener Werte über das Wuala-Netzwerk. Da das Einfügen von Daten in die Datenbank unabhängig von der Darstellung derselben sein soll, ist es nicht möglich, dass der Monitor beim Eintreffen von neuen Daten automatisch aktualisiert wird. Deshalb wird die Aktualisierung des Monitors nach einem fixen Zeitintervall durchgeführt. Der Statistics Monitor basiert nicht wie die anderen Visualisierungen auf Prefuse, sondern wird mittels Standard Widget Toolkit (SWT) dargestellt.

Für jeden Wert, der im Wuala Statistics Monitor dargestellt werden soll, muss ein `Query` Objekt erstellt werden (Abbildung: 5.12). Die einzelnen Queries eines Themenbereichs werden in einer `QueryGroup` zusammengefasst. Alle Queries werden aus einer CSV Datei

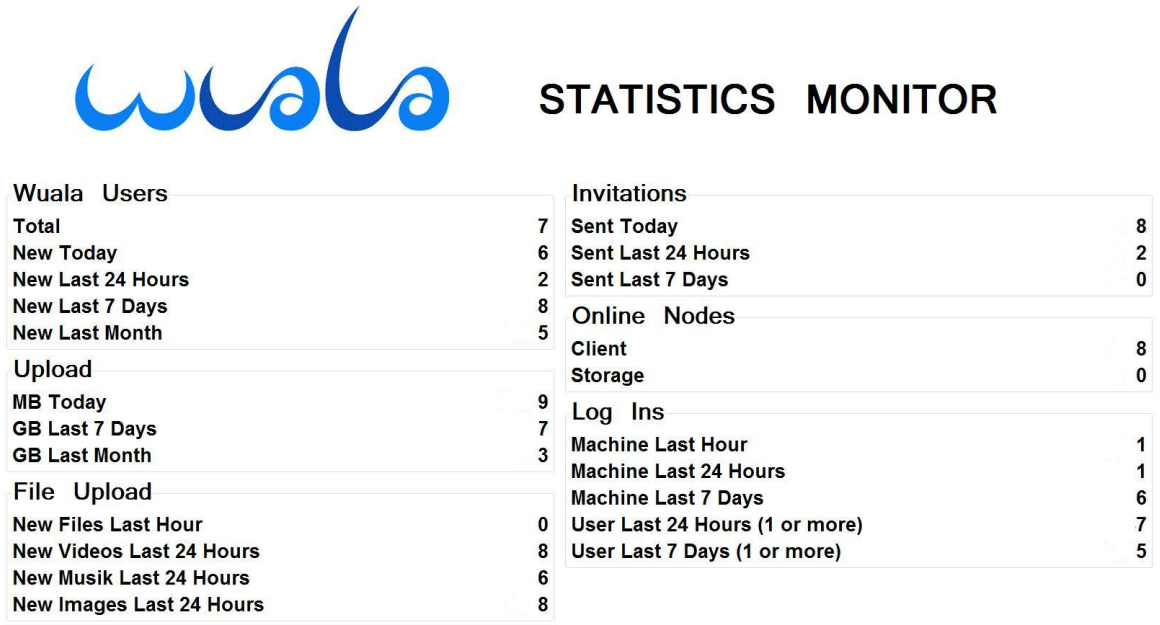

Abbildung 5.13: Wuala Statistics Monitors

gelesen, wo sie in folgendem Format abgelegt werden:

```
Group;Wuala Users;
Total;SELECT count(*) FROM UserIdentificationReport;2
```

Die erste Zeile zeigt die Definition einer neuen Gruppe mit dem Namen ”Wuala Users”. Alle nachfolgenden Queries werden in diese Gruppe geschrieben, bis eine neue Gruppe eröffnet wird. Die zweite Zeile enthält den Darstellungsnamen, die SQL Anfrage und die Aktualisierungszeit für diesen Wert in Minuten. Beim Start des Wuala Statistics Monitor initialisiert der `QueryHandler` alle Query Objekte, die in der CSV Datei enthalten sind. Danach werden die Werte anhand ihrer festgelegten Aktualisierungszeiten neu geladen. Wenn sich der Wert einer Query ändert, wird die `StatisticsFacade` benachrichtigt und die Darstellung aktualisiert. Momentan ist der Monitor nur zur Darstellung von Zahlen geeignet, doch können auf dieselbe Weise auch einfach Texte aus der Datenbank geholt und dargestellt werden. Die Abbildung 5.13 zeigt den Statistics Monitor mit Beispieldaten.

Kapitel 6

Einsatzmöglichkeiten

In diesem Kapitel werden verschiedene Einsatzmöglichkeiten des Wuala Monitoringsystems beschrieben. Das Ziel ist es, dass ein Überblick über die vielfältigen Verwendungsmöglichkeiten aufgezeigt werden kann. Wuala legt hohen Wert auf die Privatsphäre des einzelnen Benutzers. Das Monitoringsystem wird nicht zur Überwachung der Benutzerdaten, sondern zur Auswertung von Wuala selbst eingesetzt. Mit Hilfe dieser Daten soll eine stabilere und besser bedienbare Version entwickelt werden. Den nachfolgenden Ausführungen ist also anzumerken, dass die Benutzer von Wuala gefragt werden, ob die Verhaltensweisen wie Aufenthaltsdauer und Besuchshäufigkeit erfasst werden dürfen. Auch die Angaben von Wohnland und Alter sind freiwillig. Die einzigen vorausgesetzten Informationen für den Einsatz von Wuala ist ein einmaliger Benutzername und ein privates Passwort.

Die bis an hin durchgeführten und erfassten Reports sind hauptsächlich in den Bereichen Wachstum und Benutzerverhalten angesiedelt. Der Einsatz des Wuala Monitoringsystems ist aber genauso für die Überprüfung des Netzzustandes oder für die Serverüberwachung machbar. In Entwicklungsphase der Kangoo DHT und später in der Entwicklung von Wuala wurden Simulationen hinsichtlich des Netzaufbaus und des Verhaltens des Netzwerkes durchgeführt [36]. Mit dem Wuala Monitoringsystem ist es jetzt möglich zu überprüfen, ob das reale Netzverhalten mit den damals ermittelten Ergebnissen übereinstimmt.

6.1 Wachstum

Ein Einsatzgebiet des Wuala Monitoringsystems ist die Überwachung und Analyse des Wuala-Wachstums. Ein Aspekt des Wachstums ist die Anzahl der Benutzer, die ein Wuala-Konto eröffnen. Ein wichtiger Wert für die Verbreitung wird die Wachstumsrate sein. Sie zeigt auf, wie stark Wuala durch virale Verbreitung wächst und wann eine Stagnation eintritt, so dass verstärkt auf Marketingstrategien gesetzt werden muss.

Ein anderer Aspekt ist die momentane Grösse des Netzwerks, das heisst wie viele Knoten sich im Augenblick im Netzwerk eingeloggt haben. Ein dritter Punkt ist das Wachstum der Datenmenge im Netzwerk. So kann die durchschnittliche Speichermenge pro Benutzer eruiert werden und somit auf die Anzahl benötigter Fragmentserver geschlossen werden.

6.2 Benutzerverhalten

Im Bereich des Benutzerverhaltens gibt es viele Einsatzmöglichkeiten für das Wuala Monitoringsystem. So kann mit den freiwilligen Alters- und Wohnlandangaben der Benutzer herausgefunden werden, wie die demographische und geographische Verbreitung von Wuala aussieht. Eine Statistik darüber, ob der Benutzer die Werbeeinblendungen deaktiviert, liefert wichtige Hinweise, da damit die konkrete Grösse des Zielpublikums für Werbemassnahmen ersichtlich ist.

Die Benutzung der Wuala-Autostartoption lässt Rückschlüsse darauf zu, wie viele Benutzer grundsätzlich bereit sind, Wuala als dauernd laufende Hintergrundapplikation zu akzeptieren und welche Benutzer Wuala nur sporadisch einsetzen. Damit in Zusammenhang steht die Anzahl der Leute, die bereit sind, von ihrem lokalen Festplattenspeicher ans Wuala Netzwerk abzugeben und wie viele die Mindestlaufzeit von durchschnittlich vier Stunden pro Tag erreichen. Aus diesen Kennzahlen geht hervor, wie sich die Grösse und Stabilität des P2P Speichersystems entwickelt und in welchem Umfang das System durch Server unterstützt werden muss oder ob das P2P Speichersystem selbsttragend wird.

Mit dem Monitoring von Logindauer und Häufigkeit können sporadische von aktiven Benutzern unterschieden werden und auch der Prozentsatz von nicht mehr aktiven Benutzern wird sichtbar. Diese können vielleicht zu einem späteren Zeitpunkt mit einem Erinnerungs-E-Mail von einer stabileren und erweiterten Wuala-Version überzeugt und zu aktiven Benutzern gemacht werden.

Eine wichtige Analyse zeigt die Aufteilung der Dateien in die bekannten Kategorien Bilder, Filme, Musik, Dokumente und Andere. Wird Wuala vor allem für das Backup von Dokumenten eingesetzt, teilen die Benutzer hauptsächlich Fotos oder wird Wuala für das Streamen von hochauflösenden Filmen benutzt. Diese Analysen können auch Aufschluss über den weiteren Entwicklungsaufwand bringen. Je nachdem kann mehr Zeit in die Bilddarstellung oder in die Dokumentenverwaltung investiert werden.

Die Auswertung der Aufteilung der Dateien in den Bereichen privat, geteilt und öffentlich und die Analyse von Gruppen gibt ebenfalls Aufschluss darüber, für welche Zwecke Wuala eingesetzt wird. Wenn sich ein grosser Weltbereich mit öffentlichen Videoclips und Bildern entwickelt, dann muss vielleicht die Entwicklung und Bereitstellung einer vereinfachten Wuala-Version nur zur Datenbetrachtung angestrebt werden. Wenn festgestellt wird, dass private Gruppen mehrere tausend Mitglieder haben und die Dateien in der Gruppe hauptsächlich Musikdateien sind, so wird der Einsatzzweck von Gruppen höchstwahrscheinlich missbraucht und man ist gezwungen die Mitgliedergrösse von privaten Gruppen einzuschränken.

Weitere nützliche Informationen zur Verbesserung von Wuala liefern Daten über die Anzahl Besuche und die Aufenthaltsdauer der Benutzer in den verschiedenen Wuala-Bereichen. Werden alle Sucheigenschaften (Neuste Bilder, Top Filme etc.) benutzt oder kann auf eine Option verzichtet werden? Werden die Dateien mit Tags beschrieben und suchen die Benutzer auch nach solchen? In Kombination dieser Seitenbesuche mit der Logindauer und Häufigkeit können die Benutzer auch in Aktivitätsgruppen eingeteilt werden. Diese wiederum können für optimierten Kundensupport oder treffendere Werbeeinblendungen verwendet werden.

6.3 Netzzustand

Das Monitoringsystem kann auch für Untersuchungen des Netzzustandes eingesetzt werden. So kann mit der Schnappschussreporterfassung erkannt werden, wenn die Verteilung von Clientnodes und Storagenodes pro Supernode unausgeglichen ist. Auch Überprüfungen der Nachbarschaft der einzelnen Knoten helfen, damit falsche Gewichtungen und fragmentierte Subnetze erkannt und behoben werden können.

Das Auffinden von Hot Spots, also von Knoten des Netzes, die überdurchschnittlich belastet werden, kann durch die Abweichung zur durchschnittlichen Belastung eruiert werden.

Ebenso können mögliche ortsspezifische Fragmentierungen der Storagenodes überprüft werden. Das könnte der Fall sein, wenn bestimmte Fragmente nur auf europäischen Knoten abgelegt sind. Dies führt dazu, dass diese Fragmente zu europäischer Nachtzeit nur noch beschränkt und allenfalls nur noch langsam von den unterstützenden Fragmentservern bezogen werden können.

Die Grösse der Routingtabelle der Supernodes zeigt, wie die Kommunikation zwischen den Knoten funktioniert, während die durchschnittliche Anzahl Hops, die eine Nachricht bis zu ihrem Bestimmungsort benötigt, aufzeigen kann, wie die Performance des wachsenden Netzes skaliert.

Die Überprüfung des CPU- und Bandbreitenauslastung der Supernodes im passiven Modus (das heisst der Benutzer tätigt momentan keine Up- oder Downloads) zeigt wie diese Knoten von Wartungs- und Weiterleitungsaufgaben belastet werden.

Wenn bei der Messung des RAM Verbrauchs der Wuala-Applikation festgestellt wird, dass Wuala im passiven Betrieb zu viel Arbeitsspeicher benötigt, dann beeinträchtigt dies die allgemeine Computernutzung (vor allem bei älteren Systemen), was zur Folge haben kann, dass weniger Benutzer bereit sind Wuala ständig zu betreiben. Somit können sie auch keinen eigenen Speicherplatz beitragen, was dazu führt, dass das Peer-to-Peer Speichersystem langsamer als beabsichtigt wachsen kann.

6.4 Serverüberwachung

Die Performanceüberwachung der Wuala Server kann grundsätzlich ebenfalls über das Monitoringsystem geschehen. Dazu müssen auf den Servern Methoden zur Erfassung von Werten, wie verfügbarer Speicherplatz, durchschnittliche Bandbreitenauslastung oder CPU-Auslastung, bereitgestellt werden. Der Wuala Statistics Monitor könnte dann beim Erreichen eines fixen Schwellenwerts eine frühzeitige Warnung ausgeben. Wenn eine Echtzeitalarmierung gewünscht ist, so muss die Reporterfassung dahingehend erweitert werden.

Auch für die Überwachung von Servertransaktionen, wie zum Beispiel der Datenübertragungsgeschwindigkeit bei der Erstellung von Serverreplikationen, kann das Monitoringsystem eingesetzt werden.

Kapitel 7

Evaluation

In diesem Kapitel wird der Nutzen und die zielgerichtete Einsatzfähigkeit des Wuala Monitoringsystems hinsichtlich den sechs Eigenschaften Skalierbarkeit, Robustheit, Erweiterbarkeit, Verwaltung, Portabilität und Overhead beurteilt. Diese Beurteilungsfaktoren sind aus [12] entnommen.

7.1 Skalierbarkeit

Das Wuala Monitoringsystem ist in der Lage über 500 Millionen Reports pro Tag in der Datenbank abzulegen und die Festplatte des Servers kann approximativ 3,2 Milliarden Reports speichern (Annahme: durchschnittlicher Report mit fünf Werten und einem Index). Auf den ersten Blick mag das als sehr viel erscheinen, jedoch kann diese Zahl mit einigen Millionen Wuala-Benutzern bald erreicht werden. In diesem Fall würde das Monitoringsystem nicht mehr skalieren. In Abschnitt 4.5 wurden verschiedene Ansätze zur Verbesserung der Skalierbarkeit aufgezeigt, welche dann eingesetzt werden können.

Eine Verbesserung der Skalierbarkeit kann für die Schnappschusserfassung erzielt werden, wenn ein verteilter Algorithmus eingesetzt wird, mit dem Nachteil, dass das System anfälliger für Missbrauch wird. Dem Problem der Bandbreitenbelastung des Monitoringservers kann entgegengewirkt werden, wenn mehrere lokal getrennte Monitoringserver für die Reportverarbeitung eingesetzt werden. Auch die lokale Trennung der Reportaggregation von der Reportspeicherung entlastet sowohl Bandbreite wie auch Rechenleistung.

7.2 Robustheit

Das System verhält sich durch den Einsatz eines zentralen Servers robust gegen den Ausfall einzelner Knoten und den Verlust von einzelnen Reports. Wenn bestimmte Einzelreports zwingend ankommen müssen, so wird der Sendevorgang so lange wiederholt, bis die Transaktion erfolgreich durchgeführt werden konnte. Auch der verursachte Schaden von mutwillig versendeten falschen Daten kann durch den Verzicht der Datenaggregation im

Netz minimal gehalten werden, da jeder Knoten nur seine eigenen, nicht aber die Daten anderer Knoten beeinflussen kann.

Ein möglicher Schwachpunkt des Systems ist der einzelne zentrale Monitoringserver, da bei dessen Ausfall das gesamte Monitoringsystem zum Erliegen kommt. Diese Schwachstelle kann jedoch beseitigt werden, indem der Monitoringserver und die MySQL Datenbank lokal voneinander getrennt werden. Es könnten dann mehrere Monitoringserver gestartet werden und beim Ausfall des Haupt-Monitoringserver bekommen die Knoten die Adresse eines anderen Monitoringservers vom Wuala Managementserver (dieser verwaltet alle Server im Wuala-Netzwerk) zugewiesen, welcher dann die Reportverarbeitung übernimmt.

7.3 Erweiterbarkeit

Die gute Erweiterbarkeit ist ein starker Eigenschaft des Wuala Monitoringsystems. Dank dem entwickelten Framework können neue Reporttypen innert weniger Minuten erstellt und im gesamten Wuala Sourcecode eingesetzt werden. Auch die Erweiterung der Datenbankverarbeitung ist durch das Hinzufügen von neuen Annotationsfeldern schnell zu machen. Es gilt aber zu beachten, dass die neu erstellten Reports nur von Knoten ausgeführt werden, welche das Update mit dem entsprechenden Code bereits erhalten haben. Ebenso muss der Code des Monitoringservers neu deployt werden, damit das Wuala RMI Framework den neuen Reporttypen erkennen und verarbeiten kann.

Die Erweiterung von Visualisierungen ist einfach umzusetzen. Es können entweder die bestehenden Prefuse Visualisierungen erweitert oder neue Visualisierungen mit Prefuse implementiert werden. Neue darzustellende Werte im Wuala Statistics Monitor sind mit dem Hinzufügen einer einzelnen Zeile durchführbar. Grundsätzlich können alle Arten von Visualisierungen eingesetzt werden, welche die Daten über eine Schnittstelle aus der MySQL Datenbank auslesen können.

7.4 Verwaltung

Die Handhabung und der Aufwand für das Wuala Monitoringsystem benötigt wenig Ressourcen. Beliebig viele Entwickler können voneinander unabhängig sowohl neue Reports, als auch neue Visualisierungen definieren, ohne dass sie sich gegenseitig behindern oder beeinflussen. Der einzige mögliche Konfliktpunkt ist das Interface mit den Reporttypen-Konstanten. Dieser Punkt kann aber mit den modernen Java-Entwicklungsumgebungen einfach behoben werden. Zusätzlicher Aufwand entsteht jedoch erstmals, wenn die bisherige Systemkonfiguration bezüglich Speicher an ihre Grenzen stösst, da die Daten entweder ausgelagert oder die Kapazitäten erweitert werden müssen.

7.5 Portabilität

Das Wuala Monitoringsystem ist zwar sehr eng mit der Wuala Applikation verknüpft, aber trotzdem kann es grundsätzlich in allen in Java entwickelten Programmen eingesetzt werden. Die Reporterfassung und die Abspeicherung in der Datenbank sind vollständig unabhängig vom überwachten System. Einzig die Datenübertragung muss für ein anderes System neu implementiert werden, da die in dieser Diplomarbeit beschriebene Lösung das Wuala eigene RMI Framework einsetzt.

Die Portabilität auf ein anderes Datenbanksystem variiert mit der Nähe zu MySQL. Beim Einsatz anderer relationalen Datenbanken muss im besten Fall nur die Datenbankverbindung und allenfalls einige SQL Anfragen geändert werden. Beim Einsatz eines Dateisystems als Speichermedium müssten grundlegende Änderungen bei der Reporterfassung vorgenommen werden.

7.6 Overhead

Die zusätzlichen Belastungen von Speicherplatz und Bandbreite der einzelnen Knoten durch das Wuala Monitoringsystem ist gering, da das Versenden der Reports in den meisten Fällen erst nach der erfolgreichen Durchführung einer Benutzerinteraktion und den damit verbundenen Transaktionen durchgeführt wird. Ebenso wird die Aggregation von Schnappschussdaten auf einem zentralen Server bewältigt.

Ein Faktor der zu einer merkbaren Mehrbelastung des Netzwerkes führen könnte, wäre der exzessive Einsatz der Schnappschussreporterfassung. Wenn die einzelnen Schnappschüsse in zu geringen Zeitabständen durchgeführt werden, dann wird die Bandbreite des Monitoringservers stark belastet und der Aggregationsmechanismus benötigt viel Rechenzeit auf dem Server.

Kapitel 8

Zusammenfassung

8.1 Fazit

In der Zeit dieser Diplomarbeit konnte mit dem Wuala Monitoringsystem ein effizientes System zum Sammeln und Auswerten von statistischen Daten über relevante Ereignisse im Wuala-Netzwerk erstellt werden. Die Vorteile dieses Systems sind die einfache Erstellung neuer Reports und die automatische Datenerfassungskomponente. Der Entwickler erhält ein Framework, mit welchem er neue Reports definieren und diese entweder mit der Standard- oder mit der Schnappschussreporterfassung im Netz ermitteln kann. Die Daten werden in einer standardisierten Datenbank abgelegt, wo sie mittels spezialisierten Visualisierungen oder SQL Abfragen analysiert werden können. Ebenso ist die Visualisierungskomponente durch die Unabhängigkeit vom Restsystem beliebig austausch- und erweiterbar.

Das Wuala Monitoringsystem ermöglicht die Abdeckung eines breiten Spektrums von statistischen Informationen, seien diese über das Benutzerverhalten, den Netzwerkzustand oder über die Serverleistungen. Durch diese Vielfältigkeit ist klar, dass es nicht für jeden Teilbereich die optimale Lösung bietet, nichtsdestotrotz bildet es eine gute Grundlage für allfällige spezifische Erweiterungen (siehe 8.2). Informationen und Ereignisse die bis an hin überall verteilt waren, können nun erfasst, zentral gespeichert und analysiert werden, was die Beurteilung und weitere Entwicklung von Wuala stark vereinfacht.

8.2 Mögliche Erweiterungen

Obwohl das Wuala Monitoringsystem voll funktions- und einsatzfähig ist, gibt es mehrere Ansatzpunkte für zukünftige Verbesserungen und Erweiterungen. Es können neue Reports eingeführt werden und mit ihnen darauf spezialisierte Visualisierungen, welche entweder auf das Prefuse Toolkit oder auf andere Hilfen zurückgreifen.

Eine Möglichkeit zukünftigen Skalierungsproblemen zu entgegnen, wäre die Verbesserung der Schnappschusserfassung, so dass die Werte schon auf den Storagenodes oder auf den in

Abschnitt 4.4.3 vorgestellten Aggregationsservern voraggregiert werden. Hinsichtlich des Speicherplatzes sind die Verwendung von Datenbank Clustern oder eines Dateisystems denkbar.

Um die Ausfallsicherheit zu erhöhen, wäre eine Umsetzung mit mehreren Monitoringserver-Threads, welche auf eine gemeinsame, aber lokal getrennte Datenbank zugreifen, ein guter Ansatzpunkt. Dieses Konzept ermöglicht auch eine parallele Abarbeitung der eintreffenden Reports.

Zu guter Letzt könnte ein spezifisches Server-Performance Warnsystem auf den Grundlagen des bestehenden Wuala Monitoringsystems entwickelt werden.

Literaturverzeichnis

[1] Wuala Homepage, `http://wua.la/` (Dezember 2007).

[2] Omnidrive Homepage, `http://www.omnidrive.com/` (Dezember 2007).

[3] All My Data Homepage, `http://www.amdwebhost.com/` (Dezember 2007).

[4] Box.net Homepage, `http://www.box.net/` (Dezember 2007).

[5] Omemo Homepage, `http://www.omemo.com/` (Dezember 2007).

[6] D. Grolimund: Kangoo - A Distributed Storage System for the Internet, ETH Zurich, Technical Report, 2006.

[7] L. Meisser: Metadata and Security in the Kangoo Distributed File System, ETH Zurich, Technical Report, 2006.

[8] D. Grolimund, L. Meisser, S. Schmid, R. Wattenhofer: Cryptree: A Folder Tree Structure for Cryptographic File Systems, 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06), 2006.

[9] D. Grolimund, L. Meisser, S. Schmid, R. Wattenhofer: Havelaar: A Robust and Efficient Mechanism for Fair Bandwidth Trading in Peer-to-Peer Systems, 1st Workshop on the Economics of Networked Systems (NetEcom), 2006.

[10] R.Subramanyan, J. Miguel-Alonso, J. A. B. Fortes: A scalable SNMP-based distributed monitoring system for heterogeneous network computing, In Proceedings of Supercomputing2000, 2000.

[11] Jimeng Sun, E. Hoke, J. D. Strunk, G. R. Ganger, C. Faloutsos: Intelligent System Monitoring on Large Clusters, Proceedings of the 3rd workshop on Data management for sensor networks: in conjunction with VLDB 2006 DMSN '06, 2006.

[12] M. L. Massie, B. N. Chun, D. E. Culler: The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, University of California, Berkeley Technical Report, `http://ganglia.sourceforge.net/`, 2003.

[13] P.Yalagandula, P. Sharma, S. Banerjee, S. Basu, Sung-Ju Lee: S3: A Scalable Sensing Service for Monitoring Large Networked Systems, Proceedings of the 2006 SIGCOMM workshop on Internet network management INM '06, 2006.

[14] P. G. Bridges, A. B. MacCabe: IMPuLSE: Integrated Monitoring and Profiling for LargeScale Environments, Proceedings of the 7th workshop on Workshop on languages, compilers, and run-time support for scalable systems LCR '04, 2004.

[15] A.-M. Kermarrec, M. van Steen: Gossiping in Distributed Systems, ACM SIGOPS Operating Systems Review, 2007.

[16] R. Garg, V. K. Garg, Y. Sabharwal: Scalable Algorithms for Global Snapshots in Distributed Systems, Proceedings of the 20th annual international conference on Supercomputing ICS '06, 2006.

[17] R. Van Renesse, K. P. Birman, W.Vogels: Astrolabe: A Robust and Scalable Technology For Distributed System Monitoring, Management, and Data Mining, ACM Transactions on Computer Systems (TOCS), 2001.

[18] S. Abiteboul, B. Marinoiu: Distributed Monitoring of Peer to Peer Systems, Proceedings of the 9th annual ACM international workshop on Web information and data management WIDM '07, 2007.

[19] B. Clifford Neuman: Scale in Distributed Systems. In Readings in Distributed Computing Systems. IEEE Computer Society Press, 1994.

[20] M. Mansouri-Samani, M. Sloman: Monitoring Distributed Systems (A Survey) Imperial College Research Report No. DOC92/23, 1992

[21] M. A. Olson, K. Bostic, M. Seltzer: Berkeley DB, Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference, 1999.

[22] An Oracle Technical White Paper: A Comparison of Oracle Berkeley DB and Relational Database Management Systems`http://www.oracle.com/database/docs/Berkeley-DB-v-Relational.pdf`, 2006.

[23] Oracle Homepage, `http://www.oracle.com/lang/de/database/index.html`, (Dezember 2007).

[24] IBM DB2 Homepage, `http://www-306.ibm.com/software/de/db2/db2ims/db2ud.html`, (Dezember 2007).

[25] Microsoft SQL Server Homepage, `http://www.microsoft.com/sql/default.mspx`, (Dezember 2007).

[26] MySQL Community Server Homepage, `http://dev.mysql.com/downloads/mysql/5.0.html`, (Dezember 2007).

[27] PostgreSQL homepage `http://www.postgresql.org/`, (Dezember 2007).

[28] S. Ghemawat, H. Gobioff, Shun-Tak Leung: The Google File System, ACM SIGOPS Operating Systems Review, 2003.

[29] J. Dean, S. Ghemawat: MapReduce: Simplified Data Processing on Large Clusters, 6th Symposium on Operating System Design and Implementation (OSDI 2004), 2004.

[30] Hadoop Homepage, `http://lucene.apache.org/hadoop/`, (Dezember 2007).

[31] Apache Commons DBCP Homepage, `http://commons.apache.org/dbcp/`, (Dezember 2007).

[32] MySQL Handbuch, Replace Statement `http://dev.mysql.com/doc/refman/5.1/de/replace.html`, (Dezember 2007).

[33] J. Heer: prefuse: a software framework for interactive information visualization, University of California, Technical Report, 2004.

[34] J. Heer, S. K. Card, J. A. Landay: prefuse: a toolkit for interactive information visualization, CHI 2005, Human Factors in Computing Systems (2005), 2005.

[35] GraphML File Format Homepage, `http://graphml.graphdrawing.org/`, (Dezember 2007).

[36] L. Meisser, D. Grolimund: Refining Kangoo Real-World Implementation, ETH Zurich, Technical Report, 2005.

Abkürzungen

CSV	Character Separated Values
DBCP	Database Connection Pooling
DHT	Distributed Hash Table
GUI	Graphical User Interface
P2P	Peer to Peer
P2PML	Peer to Peer Monitor Language
RMI	Remote Method Invocation
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SWT	Standard Widget Toolkit
UML	Unified Modeling Language
XML	Extensible Markup Language

Glossar

Knoten Teilnehmende Einheit im Wuala-Netzwerk. Die drei Typen Supernode, Storagenode und Clientnode werden unterschieden.

Monitoring Laufende Überwachung und das Erfassen von relevanten Daten eines Hauptsystems.

Monitoring Server Server, auf welchem die Reports in die automatisch verarbeiten und in die Datenbank geschrieben werden.

NodeReport Einheit der Schnappschussreporterfassung. Nach der Aggregation der Antwortnachrichten auf den NodeReport wird der entsprechende Report generiert und vom Monitoringsystem abgespeichert.

Report Einheit des Wuala Monitoringsystems. Jeder Report erfasst Daten über ein interessantes Ereignis. Pro Report wird eine eigene Datenbanktabelle angelegt.

Schnappschuss Abbild des gesamten Netzzustandes zu einem bestimmten Zeitpunkt.

Visualisierung Eine Darstellungsvariante von Auswertungen, die aufgrund der erfassten Daten auf dem Monitoringserver durchgeführt werden.

Wuala Ein grosse verteilter Onlinespeicher, die eine Peer-to-Peer Strategie zum Erreichen eines persistenten Speichers einsetzt. Die Applikation, die vom Wuala Monitoringsystem überwacht und ausgewertet wird.

Wuala Monitoringsystem Das gesamte System zur Überwachung von Wuala. Besteht aus den Teilen Reporterfassung, Reportspeicherung und den Visualisierungen der Auswertungen.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Liste der Reports

- AddFileReport

  Erfasst das Einfügen neuer Dateien.

- AdvertEnabledReport

  Erfasst ob die Benutzer die Werbeeinblendung deaktivieren.

- AutoStartReport

  Erfasst wie viele Benutzer Wuala im Autostartmenü haben.

- ConnectedNodeSizeReport

  Erfasst die Aufteilung von Client- und Storagenodes auf die Supernodes.

- ExistingGroupsReport

  Erfasst die Erstellung von neuen Gruppen in Wuala.

- FileStatsReport

  Erfasst zusätzliche Informationen über die Dateien in Wuala, wie Anzahl globale Aufrufe.

- FriendsReport

  Statistik über die Vernetzung der Benutzer.

- InvitationEventReport

  Liste der Einladungscodes für spezielle Anlässe.

- InvitationSentReport

  Erfasst die versendeten Einladungscodes.

- InvitationLeftReport

  Erfasst wie viele Einladungen die Benutzer noch übrig haben.

- InvitationUsedReport

  Erfasst welche Einladungen von den neuen Benutzern eingelöst wurden.

- LogInReport
  Erfasst Häufigkeit und Dauer der Logins ins Wuala.
- MachineIdReport
  Liste der konkreten Computer, die am Wuala-Netzwerk teilnehmen.
- MachineLoginReport
  Erfasst die Logins der konkreten Computer ins Wuala-Netzwerk.
- MetaRWloadReport
  Erfasst die Lese und Schreibzugriffe auf dem Wuala Metaserver.
- NodeCounterReport
  Erfasst einen Schnappschuss über die momentane Wuala Netzgrösse.
- StatisticsReport
  Erfasst die Anzahl Aufrufe eines Benutzers zur Missbrauchsverhinderung.
- UserIdentificationReport
  Liste mit allen Wuala Benutzern.
- UserInfoReport
  Erfasst die freiwilligen Zusatzinformationen Wohnland und Geburtsdatum.
- UserQuotaReport
  Erfasst die Veränderungen des Speicherplatzes der Benutzer.

Anhang B

Inhalt der CD

Einzelne Dateien

inhalt.txt : CD Inhaltsverzeichnis

Diplomarbeit_Gregor_Berther.pdf : der Bericht im pdf- Format

Diplomarbeit_Gregor_Berther.ps : der Bericht im ps- Format

Zusfsg.pdf : Die deutsche Zusammenfassung

Abstract.pdf : Die englische Zusammenfassung

Schlusspräsentation.ppsx : Die Folien der Schlusspräsentation

Ordner Bericht

Enthält die LaTex- Dateien dieses Berichts, sowie die im Bericht verwendeten Bilder und den Bericht in den geforderten Formaten.

Ordner Wuala Monitoringsystem

Enthält den Java Sourcecode des Wuala Monitoringsystems, geordnet in die Bereiche Monitoringserver und Visualisierungen.

Ordner Zusätzliche Software

Enthält die vom Wuala Monitoringsystem benutzte Opensource Software (Prefuse Toolkit und Apache Commons DBCP).

Ordner Papers

Enthält alle für diese Diplomarbeit verwendeten Arbeiten und Berichte. Die Zahl im Dateinamen stimmt mit der Referenzzahl in diesem Bericht überein.

Ordner Bilder

Enthält alle für diese Diplomarbeit verwendeten Bilder im gif-Format.