Endbericht - Netidee Projekt 33980
„Spotlight“ - Einblicke in die Underground Economy

13. Juli 2009

1 Allgemeines

Die primäre Zielsetzung von „Spotlight“ besteht darin, automatisiert Informationen aus der Domäne der Underground Economy im Internet zu sammeln. Diese können in weiteren Schritten für die Verarbeitung und Analyse des Verhaltens von Online-Kriminellen verwendet werden.

Prinzipiell werden in der Underground Economy unterschiedliche Technologien zum Informationsaustausch und zur Geschäftsabwicklung verwendet. Am verbreitetsten ist der Internet Relay Chat (kurz IRC), aber auch andere Kommunikationsmittel wie z.B. Newsgroups oder Webforen werden eingesetzt.

Im Rahmen von Spotlight wurde ein flexibles Framework implementiert, das es ermöglicht, Sensoren für unterschiedliche Kommunikationstechnologien mit geringen Aufwand zu entwickeln und zentral zu verwalten.

Die Arbeitsschritte können in zwei Schwerpunkte aufgegliedert werden:

1. Framework
2. Sensorik

Die Entwicklung wurde ab Projektbeginn parallel an beiden Schwerpunkten betrieben, wobei im Rahmen der Sensorik ein IRC-Sensor implementiert wurde, der bereits für Langzeitanalysen verwendet wird.

In den folgenden Absätzen wird zuerst auf den Aufbau und die Funktionsweise des Frameworks eingegangen. Anschließend werden die Gestaltung des IRC Sensors und der damit verbundenen technischen Herausforderungen erläutert.

2 Framework

Das Framework bildet das zentrale Modul dessen primäre Aufgabe die Koordinierung der einzelnen Sensoren ist.

2.1 Architektur

Die Schnittstellen mit den Sensoren sind derart konzipiert, dass diese dynamisch geladen und konfiguriert werden können und nicht schon zum Übersetzungszeitpunkt gebunden werden müssen. Dadurch wird eine flexible Entwicklung der Sensoren getrennt vom Framework ermöglicht.

Zusätzlich stellt das Framework die Grundbausteine für die Sensoren zur Verfügung, damit diese ohne Probleme die gemeinsamen Ressourcen effizient nutzen können und durch Multithreading und Message Queue Abstraktionen ein hochparalleler Betrieb der Sensoren gewährleistet wird.

In Abbildung 1 wird die Basisarchitektur schematisch dargestellt.

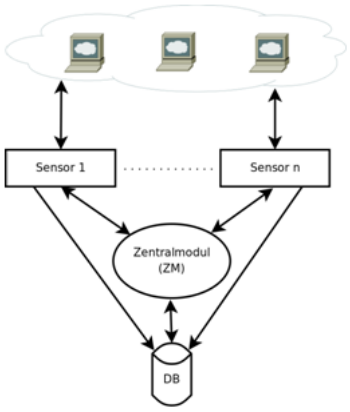

Abbildung 1: Framework - Architektur

Das Framework ist in Java [1] implementiert und unterstützt zusätzlich das aspektorientierte Programmierparadigma welches durch das Einbinden von AspectJ [2] ermöglicht wird.

Als Zielplattform wurde ein Server mit Ubuntu Linux inklusive mehrerer Netwerkanbindungen, MySQL Server und Sun Java konfiguriert und in einer Testumgebung verwendet.

2.2 Kommunikationsschicht

Vorbereitend für die Sensorik wurde eine Kommunikationsschicht entwickelt. Sie abstrahiert die Netzwerkanbindung von Java und erlaubt die Verwendung von asynchronen Kommunikationsmethoden.

2.3 Kommandozeile

Nach der Festlegung der Grundarchitektur wurde eine Kommandozeileneingabe zur komfortableren Bedienung in das Framework integriert.

Die Definition der Kommandobefehle und deren Verhalten wurde mittels Command Design Pattern [6] umgesetzt. Das Command Design Pattern erlaubt eine einfach Definition des Befehlssyntax und der damit verbunden Ausführungseigenschaften ohne dass das zu definierende Kommando dem Eingabeinterpreter zum Übersetzungszeitpunkt bekannt sein muss. Das bietet den Sensorikmodulen die Möglichkeit eigene Eingabebefehle zu registrieren und beim Entladen der Module wieder aus dem Befehlssatz zu entfernen.

2.4 Zentrale Konfiguration

Zusätzlich zu der Kommandozeileneingabe ist aufgrund vielseitiger Konfigurationsparameter ein einheitlicher Konfigurationsservice für das Framework notwendig. Der Service wurde so implementiert, dass für die Sensorikmodule ein einfacher Zugriff sichergestellt ist und die Sensorentwickler eigene Parameter definieren können.

Der Konfigurationservice verwaltet die Einstellungen in einer XML-Datei und basiert auf dem Configuration Paket von Apache Commons [8]. Er beinhaltet zusätzlich vordefinierte allgemeine Konfigurationsoptionen, z.B. für die zur Verfügung stehenden Netzwerkschnittstellen, Anzahl der Verbindungsversuche, usw. die von den Sensorikmodulen verwendet werden.

Es bleibt jedoch dem Sensorikentwickler überlassen, auf diese Parameter zuzugreifen. Diese Flexibilität soll die Unabhängigkeit der Module unterstützen und dadurch auch einen getrennten Einsatz ermöglichen.

2.5 Namensgenerator

Des Weiteren stellt das Framework den Sensorikmodulen einen Namensgenerator zur Verfügung. Er wird für die Erzeugung von Identitäten (z.B. in einem Chat) verwendet und unterstützt Vor- und Nachnamen, Namenskürzel sowie die Generierung von Domainnamen.

Der Generator arbeitet nicht mit einer vordefinierten Liste von Namen, sondern verwendet eine Sammlung von häufig vorkommenden Vokal- bzw. Konsonatenkombinationen um neue Namen zu generieren.

2.6 Datenspeicherung

Für die Datenspeicherung wird das Datenbankmanagementsystem (DBMS) MySQL [3] verwendet. Durch die Abstrahierung im Framework durch Hibernate [4] kann ein beliebiges DBMS verwendet werden, man ist also nicht auf einen bestimmten Typ bzw. SQL-Dialekt beschränkt, solange ein JDBC Connector [5] beim Laden zur Verfügung steht.

3 Sensor für Internet Relay Chat

Die folgenden Absätze beschreiben die einzelnen Entwicklungsschritte die bei der Umsetzung des IRC-Sensors durchgeführt wurden.

3.1 IRC-Nachrichtensystem

Das IRC Protokoll [7] ist ein Klartext-Nachrichtensystem, das auf TCP/IP basiert. Prinzipiell werden dabei mehrere Server zu einem “Netzwerk” gruppiert, wobei in einem Netzwerk Gesprächskanäle (genannt Channels) von Teilnehmern eröffnet werden können.

3.1.1 Protokoll

Nach Abschluss der Analysephase wurde offensichtlich, dass es von Vorteil ist, einen eigenen IRC-Protokollstack für den Sensor zu entwickeln, weil keine existierende und zur Verfügung stehende Lösunge die definierten Anforderungen erfüllt. In der Implementierung des Protokollstacks wird mittels Sprachdefinition (durch reguläre Ausdrücke) die einzelnen IRC-Nachrichten in ein internes Datenformat umgewandelt.

Der entwickelte Protokollstack wurde einer Reihe von extensiven Tests unterzogen, damit sichergestellt werden kann, dass das Verhalten den Protokoll-"Regeln" entspricht, und möglichst alle Fehlerquellen beseitigt sind. Dies ist besonders wichtig, da es sich um den Datenübertragungskanal und damit um die primäre Informationsquelle des Sensors handelt.

Basierend auf den Ergebnissen dieser Tests wurde das Protokoll über die im Standard [7] definierten Nachrichtenformate hinaus erweitert, um mit den gängigen IRC-Server Implementierungen wie z.B. Undernet [9] konform zu sein. Dies ist nötig, da diese die Nachrichten zum Teil unterschiedlich handhaben bzw. eigene Erweiterungen verwenden.

3.1.2 Nachrichten Bots

Nach der Implementierung des Protokolls wurde eine Abstraktionsschicht entwickelt, um die Protokollnachrichten über eine Schnittstelle zur Verfügung stellen zu können. Diese Abstraktionsschicht übernimmt das Gundverhalten das von gängigen IRC-Clients zur Verfügung gestellt wird, und verwendet hierzu den vom Framework abstrahierten Java Socket Layer um einen zuverlässige Kommunikation zu garantieren.

Zusätzlich wurden Verhaltensmaßnahmen nach dem „Monte Carlo" Prinzip für die Übertragung hinzugefügt, die durch explizit eingeführte zeitliche Unschärfen einen menschlichen Gesprächspartner simulieren.

3.2 Architektur

Das Systemdesign wurde so konzipiert, dass mit möglichst wenig Netzwerkressourcen eine möglichst große Anzahl von Nachrichtenkanälen (unabhängig von dem Netzwerk in dem sie sich befinden) parallel beobachtet werden können.

3.2.1 Komponenten

Für die Umsetzung der Architektur wurden nachfolgende Komponenten implementiert:

IRC Sensorikmodul In einem ersten Schritt wurde ein Sensorikmodul implementiert, dass die zentrale Verwaltung für den Sensor realisiert.

Für das Module wurden die Schnittstellen implementiert, die dem Framework das Laden des Sensors erlauben und die Aufgabenzuteilung ermöglichen. Ein Teil der Aufgaben - jene, die

sich auf die Steuerung von einzelnen Nachrichtenkanälen innerhalb eines Netzwerkes beziehen - werden an den Netzwerkmanager weitergereicht.

Zusätzlich sind alle zentralen Funktionen für den Aufbau der Datenbankverbindung und zum Anfordern der Netzwerkressourcen implementiert.

Netzwerkmanager Aufbauend auf dem Sensorikmodul wurde eine Managementeinheit erstellt, die die Aufträge innerhalb des Netzwerkes administriert. Die Managementeinheit übernimmt auch die Verwaltung der Bots, was mittels eines Object Pool Patterns umgesetzt wurde. Dies erleichtert die Handhabung der Strategien, da diese so immer von einem wohldefinierten Zustand ausgehen können.

Grundbausteine für Strategien Strategien definieren ein Verhalten für die Ausführung von Aufgaben. Um eine effiziente Entwicklung weiterer Strategien (die als Plugins geladen werden können) zu unterstützen, wurden einige allgemeine Funktionalitäten wie z.B. der Zugriff auf die Bots, Statusrückmeldungen usw. gekapselt, und als abstrakte Implementierung zur Verfügung gestellt.

3.2.2 Nachrichtenfluß

Für die parallele Verarbeitung von vielen Nachrichten müssen diese asynchron weitergeleitet werden. Dazu stützt sich die Implementierung auf das Prinzip von Message Queues, die in jedem Nachrichtenglied eingebunden sind. Diese werden sowohl für eingehende-, als auch ausgehende Nachrichten verwendet.

In Abbildung 2 wird der Nachrichtenfluß für eingehende Nachrichten schematisch dargestellt. Die Nachricht wird über den Socket empfangen, asynchron an den zuständigen Bot weitergeleitet, dort in das interne Datenformat umgewandelt, und schließlich an den überliegenden Netzwerkmanager weitergegeben. Dieser entscheidet anhand des Nachrichtentypes wie diese verarbeitet werden sollen, und reicht die Nachricht an die Strategien weiter die darauf das Verhalten des Bots bestimmen.

Analog zu den eingehenden Nachrichten wurde das selbe Prinzip für ausgehende Nachrichten umgesetzt, wie in Abbildung 3 zu sehen ist. Die Strategie definiert ein Verhalten, das vom Bot umgesetzt wird, indem dieser ein Nachrichtenobjekt generiert und in seinen Sendepuffer legt. Dieser wird asynchron abgearbeitet und via Socket an den Server gesendet.

3.3 Datenspeicherung

Nach der Entwicklung der Kernkomponenten für die Datenspeicherung im Framework wurde mit der Umsetzung der Persistenzschicht im IRC Sensor begonnen.

Die zu speichernden Informationen sind durch das IRC-Protokoll, das nicht nur die Nachricht selbst, sondern auch Daten über die Benutzer beinhaltet, gegeben. Zusätzlich ist auch das Client-To-Client Protokoll (kurz CTCP) auf dem IRC-Protokoll aufgesetzt.

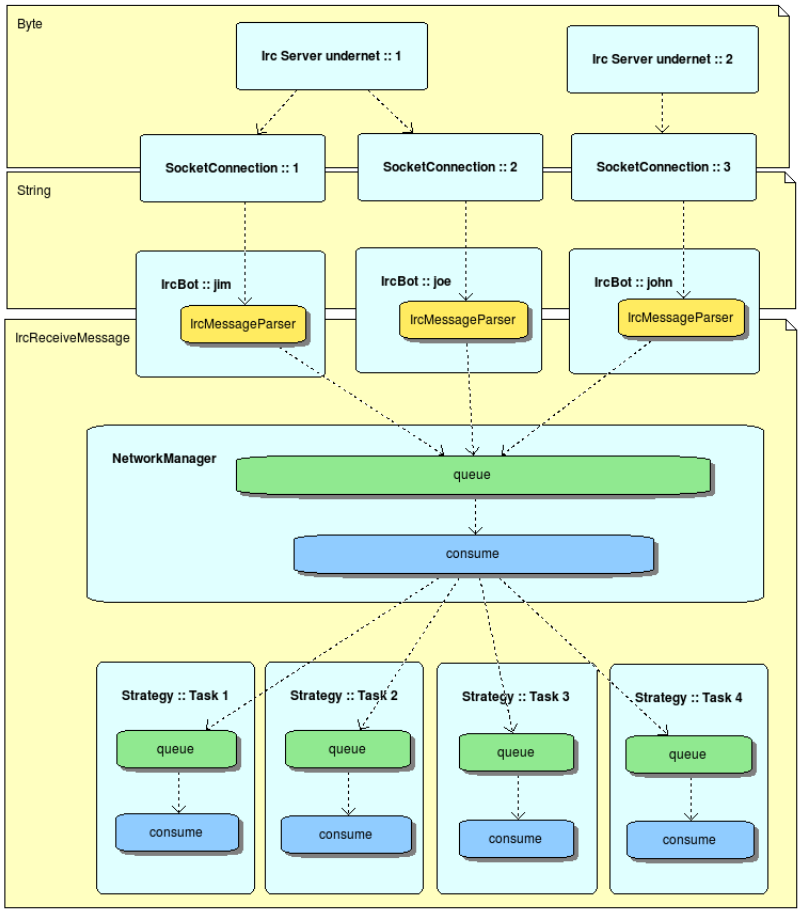

Abbildung 2: Nachrichtenfluß für eingehende Nachrichten

Die Persistenzschicht selbst ist durch die Ausnutzung des aspektorientierten Paradigmas sehr flexibel und kann an jedem Punkt im Nachrichtfluß eingehängt werden. Dies bietet die Möglichkeit Daten zwischen allen Transformationsschritten zu speichern.

In Abbildung 4 wird eine schematische Übersicht über die Struktur der gespeicherten Daten und deren Zusammenhänge dargestellt.

3.4 Weiterentwicklung der Strategien

Die Zielsetzung der umgesetzten Strategien ist es, so viele Informationen wie möglich in Channels zu sammeln. Jedoch muss gleichzeitig gewährleistet werden, dass man die Funktionalität des IRC Channels nicht beeinträchtigt und aus dem Nachrichtenkanal ausgeschlossen wird.

Die folgenden vier Strategien wurden als Basis für die Informationsgewinnung implementiert.

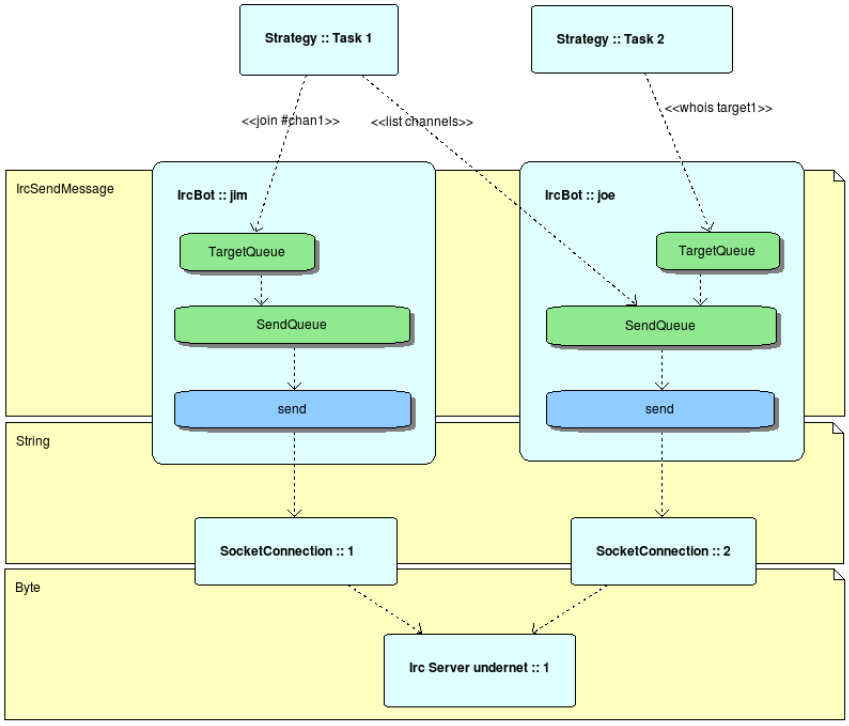

Abbildung 3: Outgoing Message Flow

3.4.1 Basic Strategy

Die Basic Strategy ist der Ausgangspunkt für alle weiteren Strategien. Sie bietet einen gewissen Grad an Robustheit in dem z.B. sofort ein anderer Bot in einem Channel eingesetzt wird, falls der ursprüngliche aus diesem Channel geworfen wurde oder Verbindungsprobleme hat.

Zusätzlich bietet die Strategie die Möglichkeit das Netzwerk nach Nachrichtenkanälen die ein bestimmtes Muster aufweisen zu durchsuchen und zu beobachten. Dabei werden HOIS Abfragen durchgeführt, und alle öffentlichen Nachrichten erfasst.

3.4.2 Chain Strategy

Das Hauptaugenmerk der Chain Strategy ist das Auffinden “interessanter” Nachrichtenkanäle. Basierend auf den Funktionalitäten der Basic Strategy werden Bots in definierbare Channels plaziert, von denen aus sie Informationen über die Benutzer (insbesondere in welchen zusätzlichen Channels sich diese befinden) sammeln. Hierbei wird versucht, die neu aufgenommenen Channels weiter zu verfolgen, falls dies die Ressourcen zulassen.

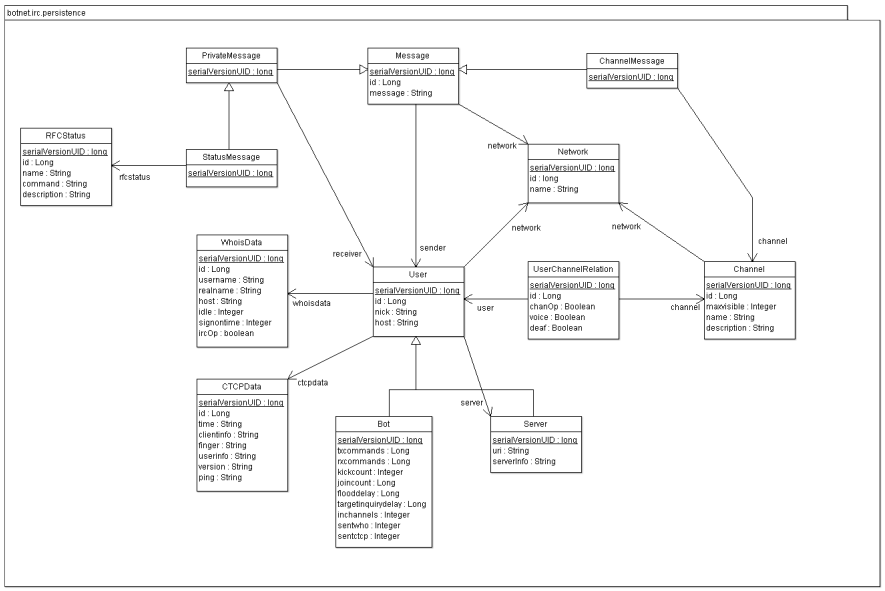

Abbildung 4: Sensorik IRC - EER-Diagramm

3.4.3 Active Strategy

Die Active Strategy ist tritt direkt mit anderen Benutzern in Kontakt: Sobald ein Benutzer eine Nachricht schreibt, die ein bestimmtes Muster erfüllt, wählt die Strategie einen Bot aus und sendet dem Benutzer je nach der Auswahl der Nachrichten eine Message, und beobachtet das weitere Verhalten des Benutzers.

3.4.4 Oper Strategy

Die Oper Strategy wurde implementiert, um die Hierarchie der Underground Economy auszunützen - “vertrauenswürdige” User, die in vielen Channels mit Voice oder Operator Status konnotiert werden, werden besonders genau beobachtet.

Literatur

[1] Sun Java6, http://java.sun.com/

[2] Eclipse AspectJ, http://www.eclipse.org/aspectj/

[3] Sun MySQL, http://www.mysql.com/

[4] Hibernate, http://www.hibernate.org/

[5] Java Database Connectivity (JDBC), http://java.sun.com/javase/technologies/database/

[6] Command Pattern, http://en.wikipedia.org/wiki/Command_pattern

[7] Internet Relay Chat Protocol (RCF 1459), http://www.irchelp.org/irchelp/text/rfc1459.txt

[8] Apache Commons Configuration, http://commons.apache.org/configuration/

[9] Undernet Server, http://coder-com.undernet.org/