

Image Multicasting

1 Problemdefinition

Datum: 1. November 2005

Projektbezeichnung: Image Multicasting

Autor:

Name: Geith Alois

Matrikelnummer: 9450190

Kurzbeschreibung:

Aufgabe dieses Projektes ist es eine Client/Server Applikation zu entwickeln die Bilder per IP Multicast (UDP) an Clients versendet. Die Applikation wird nach objektorientierten Gesichtspunkten in der Programmiersprache JAVA entwickelt.

Projektziele:

Folgende Anforderungen soll das Client Server Paar erfüllen.

Server:

- Kommando Zeilen Applikation (kein GUI).
- Bilder werden aus einem konfigurierbaren Verzeichnis gelesen.
- Die Dauer bis das nächsten Bild gesendet wird soll konfigurierbar sein.
- Nach jedem Durchlauf wird das Bilder-Verzeichnis erneut geprüft und die enthaltenen Dateien werden gesendet.
- Konfigurationsdaten werden in einer Datei gespeichert die zum Start des Servers einmal gelesen wird.
- Die Übertragung erfolgt per IP Multicast an eine Gruppe von Clients.

Client:

- Java Swing GUI.
- Empfangene Bilder werden in einem frei skalierbaren Fenster dargestellt.
- Konfigurationsdaten werden in einer Datei gespeichert.
- Das aktuelle Bild wird erst nach Empfang eines vollständigen neuen gewechselt.

Anmerkungen:

1. Bild bezieht sich immer auf eine Datei im JPG, GIF oder PNG Format.

Projektumfang:

Das Projekt muss innerhalb von 12 Wochen im Zuge der Lehrveranstaltung Praktikum aus Programmierung am Institut für Informationswirtschaft an der Wirtschaftsuniversität Wien abgeschlossen werden.

Lösungsvorschlag:

Neben den durch die Projektziele genannten detaillierten Anforderungen an die Funktionalität der Software ist das Hauptaugenmerk vor allem auf die Übertragung der Bilder zu legen. Da UDP im Gegensatz zu TCP weder ein zuverlässiges noch ein verbindungsorientiertes Protokoll darstellt, ist darauf zu achten, dass Bilder nicht nur teilweise oder falls auf mehrere Pakete aufgeteilt in der falschen Reihenfolge vom Client empfangen werden. Um diese Probleme zu handhaben kann auf ein bereits in JAVA implementiertes „reliable multicast“ Protokoll zurückgegriffen (beispielsweise TRAM - [Java Reliable Multicast Service\[tm\]](#)) werden.

Verglichen mit Unicast-Protokollen wie z.B. HTTP, ermöglicht es IP Multicast einem Server seine Daten an eine Gruppe von Empfängern in einem IP Netzwerk zu senden. Empfänger „registrieren“ sich für den Empfang in dem sie sich einer Gruppe, definiert durch eine spezielle Klasse von IP Adresse (Klasse D), anschließen. Durch diese Art der Übertragung müssen Daten (hier Bilder) nicht sequentiell zu jedem Client übertragen werden. Bandbreite wird dadurch effektiver als bei einer Serie von Unicasts genutzt.

Da durch dieses Protokoll nur die Reihenfolge und Vollständigkeit der Daten gewährleistet ist aber noch keinerlei Information über Ende bzw. Start einer Datei (die Übertragung erfolgt in mehreren Paketen) geliefert werden, muss zusätzlich zu TRAM ein weiteres Protokoll angewendet werden.

Als Logsubsystem kann Log4j des Apache Jakarta Projekts ([Log4j](#)) zum Einsatz kommen.

2 Analyse

Nach Auswertung der Anforderungen konnten zwei Haupt-Usecases identifiziert werden. Bilder versenden und Bilder empfangen auf die im Folgenden näher eingegangen wird. Anforderungen wie z.B. Konfiguration werden nicht im Detail beschrieben da sie keine wesentlichen Benutzeranforderungen darstellen.

Nach der Beschreibung der Usecases wird auf die Art der Übertragung bzw. auf das Protokoll, das den wesentlichen technologischen Aspekt dieses Projekts darstellt, näher eingegangen.

2.1 Usecase Diagramm

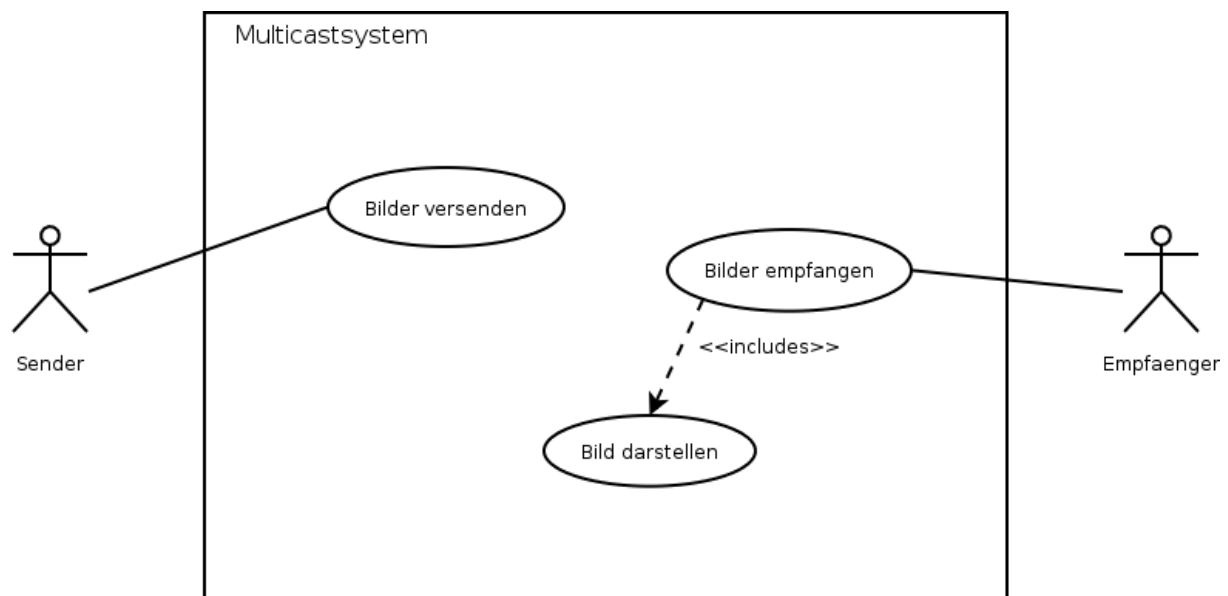


Abbildung 1: Usecase Diagramm

2.1.1 Bilder versenden

Usecase Name: Bilder versenden

Akteure: Sender

Vorbedingungen:

- Bilddateien befinden sich in einem Verzeichnis.
- Konfigurationsdatei der Serverapplikation ist erstellt.
- Empfänger vorhanden.

Nachbedingungen:**Auslöser:**

- Sender möchte Bilder versenden.

Ablaufbeschreibung:

1. Bildverzeichnis auf Bilddateien durchsuchen.
2. Ein Bild zum Senden einlesen.
3. Headerpaket senden.
4. Datenpakete senden.
5. nächstes Bild (weiter mit Schritt 2.)

Fehlsituationen:

- kein Netzwerk.
- keine Empfänger.
- Verzeichnis ist leer.
- keine Bilddateien im Verzeichnis.

Instanzen:

- Ein Sender stellt die Bilder seiner letzten Griechenlandreise in ein Verzeichnis auf seinem PC und möchte diese seinen Nachbarn per Multicast vorführen. Dazu konfiguriert er die Serverapplikation und startet diese. Die Applikation sendet Datei für Datei an eine Menge von Empfängern.

Ergebnisse:

- Bilder werden gesendet.

Autor: Geith Alois

Ursprung: Praktikum aus Programmierung Projektarbeit

2.1.2 Bilder empfangen

Usecase Name: Bilder empfangen

Akteure: Empfänger

Vorbedingungen:

- Sender versendet Bilder.

Nachbedingungen:**Auslöser:**

- Empfänger möchte empfangene Bilder eines Multicasts am Display darstellen.

Ablaufbeschreibung:

1. Empfänger startet die Client Applikation.
2. Empfänger wählt den Vollbild- oder Fenstermodus.
3. Empfänger registriert sich zum Multicastempfang.
4. Client Applikation synchronisiert sich auf den Datenstrom.
5. Client Applikation empfängt Daten.
6. Die Daten werden dargestellt.

Fehlsituationen:

- kein Netz.
- kein Sender.
- Daten sind keine Bilder.

Variationen:**Instanzen:**

- Der Empfänger möchte die Urlaubsbilder seines Nachbarn der sie per Multicast im lokalen in-house Netz zur Verfügung stellt empfangen. Dazu startet er seine Client-Applikation und wartet auf das Eintreffen der Daten.

Ergebnisse:

- Bilder werden empfangen.
- Bilder werden dargestellt.

Autor: Geith Alois**Ursprung:** Praktikum aus Programmierung Projektarbeit

2.1.3 Bild darstellen

Usecase Name: Bild darstellen**Akteure:****Vorbedingungen:** Bild wurde empfangen.**Nachbedingungen:****Auslöser:** Bild wurde empfangen.**Ablaufbeschreibung:**

1. Bild wird empfangen.
2. Bild wird umgewandelt (in passendes Format).
3. Bild wird in Fenster oder Fullscreen dargestellt.

Fehlsituationen:

- Daten sind kein Bild.

Variationen:**Instanzen:**

- Ein Nachbar empfängt per Multicast Urlaubsbilder seines Kollegen und stellt diese in einem Fenster auf seinem PC dar.

Ergebnisse:

- Bild wird dargestellt.

Autor: Geith Alois**Ursprung:** Praktikum aus Programmierung Projektarbeit

2.2 Daten Übertragung (Protokolle)

2.2.1 Basisprotokoll

IP Multicast ([RFC 1112](#)) [Deer89] ist ein Internetprotokoll das im Gegensatz zu TCP/IP (unicast) die Übermittlung von Datenpaketen an eine Gruppe von Empfängern ermöglicht. Hierbei werden Datenpakete (Datagrams) an eine gemeinsame Klasse D IP Adresse gesendet auf die registrierte Gruppenmitglieder (Empfänger) horchen.

Klasse D IP Adressen sind Adressen aus einem speziell für Multicast reservierten Adressraum. Im Moment stehen zwar einige Methoden zur Allokation einer Adresse aus diesem Bereich zur Verfügung aber diese werden nicht immer eingesetzt und haben noch einige Nachteile [RoKH98]. Die Folge ist, dass es im Moment durch einen nicht abgestimmten Einsatz von ein und der selben Klasse D IP Adresse durch 2 verschiedene Multicast-Gruppen zu Kollisionen kommen kann die den Applikationseinsatz beeinträchtigen oder sogar verhindern können.

Die Aufteilung des Wertebereichs D kann folgender Tabelle entnommen werden.

Start Adresse	End Adresse	Beschreibung
224.0.0.0	224.0.0.255	Reserviert für „well known“ Multicast-Dienste.
224.0.1.0	238.255.255.255	Globale Multicast-Adressen.
239.0.0.0	239.255.255.255	Lokale Multicast-Adressen.

Tabelle 1: Klasse D IP Adreßraum

Damit Multicast funktioniert müssen IP Router Registrierungs-Anfragen von Clients für eine bestimmte Gruppe verarbeiten können. Dies ist notwendig um dazu in der Lage zu sein die vom Server versandten Datenpakete zu duplizieren und an alle registrierten Empfänger zu verteilen. Diese Kommunikation zwischen Client und Router erfolgt über das Internet Group Management Protocol ([RFC 3376](#)) [CDKF02] welches eine Erweiterung des Internet Protokolls darstellt.

IP Multicast setzt zur Kommunikation das User Datagram Protocol (UDP) ein welches weder zuverlässig noch verbindungsorientiert ist. Da Bilddateien (in mehreren Datenpaketen) übertragen werden sollen sind beide Eigenschaften in diesem Projekt aber von wesentlicher Bedeutung. Um diese Eigenschaften zu erlangen wird also ein zusätzlicher Mechanismus das sowohl die Vollständigkeit als auch die Reihenfolge der Datenpakete sicherstellt benötigt.

TRAM (A Tree Based Reliable Multicast Protocol) ein auf einer Baumstruktur basierendes zuverlässiges Multicast Protokoll [CHKW98] erfüllt genau diese Anforderungen. Entwickelt wurde TRAM durch Sun Microsystems. Eine Implementierung dieses Protokolls und anderer Services die zur Erstellung von verlässlichen Multicast Applikationen notwendig sind stehen durch das Projekt [Java Reliable Multicast Service\[t\]](#) zur Verfügung.

Wie schon erwähnt erzielt TRAM seine Zuverlässigkeit durch den Einsatz einer Baumstruktur die auf eine skalierbare Art und Weise sicherstellt, dass alle Empfänger die Daten erhalten.

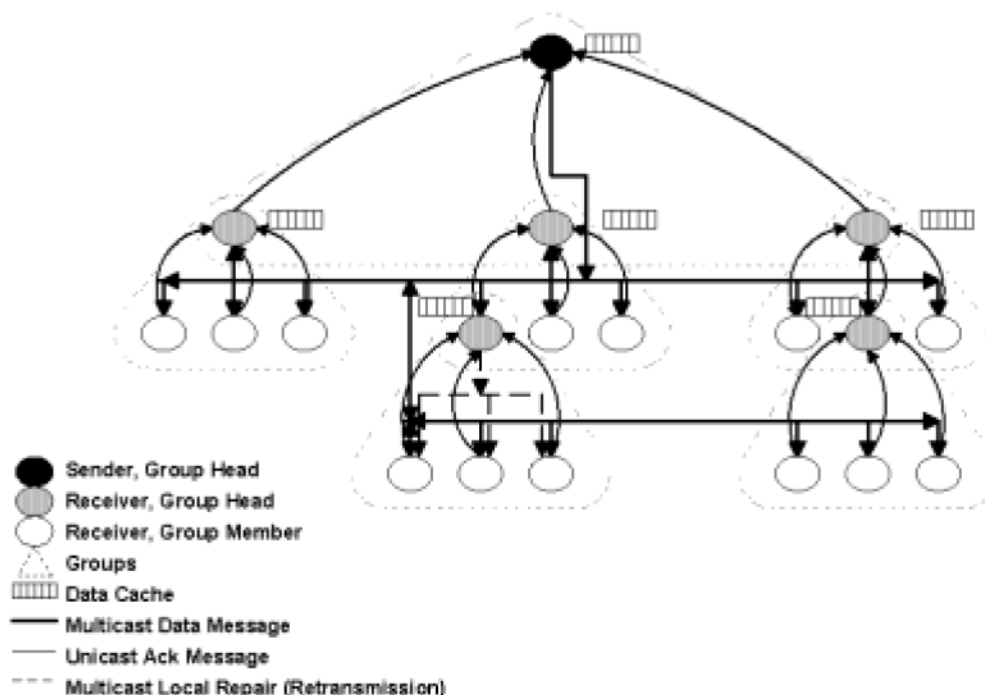


Abbildung 2: Hierarchischer "Repair-Tree" in TRAM [CHKW98]

Abbildung 2 zeigt einen typischen hierarchischen „Repair-Tree“ wie er bei TRAM eingesetzt wird. Der Sender verschickt seine Datenpakete per Multicast (UDP) an die Gruppe. Dabei agieren einige der Gruppenmitglieder als Repair-Heads für eine Untergruppe der Empfänger die sich üblicherweise in lokaler Nähe, bezogen auf das Netz, befinden. Jeder Empfänger benachrichtigt durch Acknowledgment Nachrichten (eine pro Fenster an Paketen) per Unicast seinen Repair-Head über den erfolgreichen Empfang der Pakete. Tritt ein Fehler auf können Pakete lokal (im jeweiligen Ast des Baums) nochmals versendet werden.

Ein weiteres nützliches Services welches durch Bibliotheken dieses Projekts zur Verfügung steht ist das (automatische) Aushandeln eines Kanals (einer benannten Verbindung bzw. eines Services) über einen so genannten Advertiser und Channelmanager. Dieser Mechanismus folgt dabei dem Session Announcement Protocol ([RFC 2974](#)) [HaPW00] und dem Session Description Protocol ([RFC 2327](#)) [HaJa98]. Informationen über das Service (= der Kanal) werden dabei in einem so genannten Advertisement gekapselt und an eine well-known Multicast Adresse und Port gesendet. Clients die auf diese horchen bzw. dafür registriert sind können damit Informationen über ein Service erlangen. Für detaillierte Informationen über diese Protokolle sei auf die beiden genannten RFCs und Referenzen in diesem Dokument verwiesen.

2.2.2 Aufsetzendes Protokoll

Wie schon im Lösungsvorschlag der Problemdefinition erwähnt muss zusätzlich der Anfang bzw. das Ende der Bilddaten erkannt werden. Um dies zu bewerkstelligen wurde zusätzlich folgendes einfaches Protokoll implementiert.

Alle Pakete enthalten einen Header und eine Payload und haben die gleiche Länge. Unterschieden wird dabei zwischen einem Dataheader-Paket, einem Paket das gesendet wird bevor eine neue Datei versendet wird, und einem Data-Paket. Die im Headerteil enthaltenen Informationen unterscheiden sich dabei.

Ein **Dataheader-Paket** enthält abgesehen vom Datennamen keine weiteren Daten und entspricht folgendem Format.

Header (18 Byte):

length of the packet	- 2 bytes
sequence number	- 4 bytes (0 für einen Dataheader)
header type	- 1 byte (0 für einen Dataheader)
data size	- 8 bytes (Anzahl von Bytes des Dateinamens)
unused	- 2 bytes
datanamelength	- 1 byte

Payload:

dataname	- remaining bytes
----------	-------------------

Ein **Data-Paket** entspricht folgendem Format.

Header (6 Byte):

length of the packet	- 2 bytes
sequence number	- 4 bytes

Payload:

data	- remaining bytes
------	-------------------

Header Type 255 ist für ein End-Paket reserviert.

Sendeablauf:

1. Dataheader-Paket senden
2. Datenpaket senden.
3. Datenpaket senden.
4. ...
5. [Daten wurden versendet] => gehe zur nächsten Datei und beginne wieder mit 1.

3 Design

3.1 Klassendiagramme

3.1.1 Server

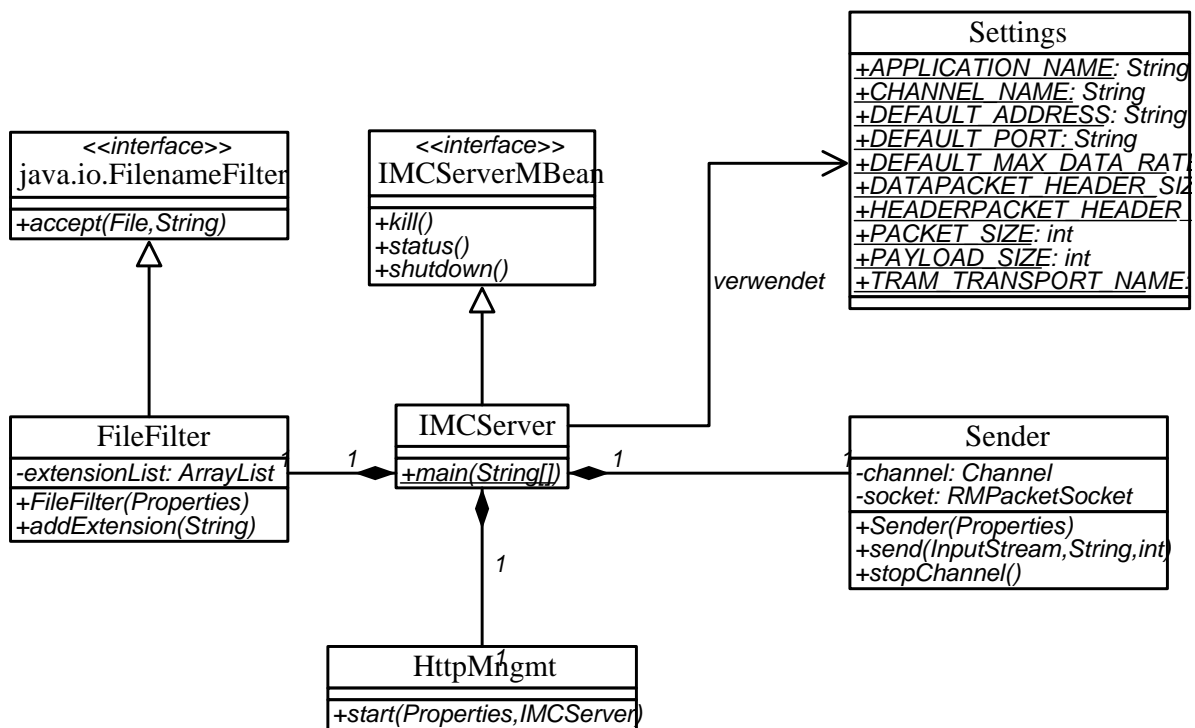


Abbildung 3: Server Klassendiagramm

Klassenbeschreibung (Übersicht):

IMCServerMBean	Interface für die Java Management Extension (JMX) Anbindung.
FileFilter	Konfigurierbarer FileNameFilter der in Verbindung mit einem Directorylisting genutzt wird.
IMCServer	Server der Dateien aus einem Verzeichnis liest und diese mittels eines Sender-Objekts versendet.
Sender	Versendet Daten per IP Multicast und nutzt das TRAM Protokoll als Basisprotokoll
Settings	Konstanten die von Server und Client benutzt werden.

Tabelle 2: Server Klassenbeschreibung

Siehe Java-Doc für nähere Beschreibungen.

3.1.2 Client

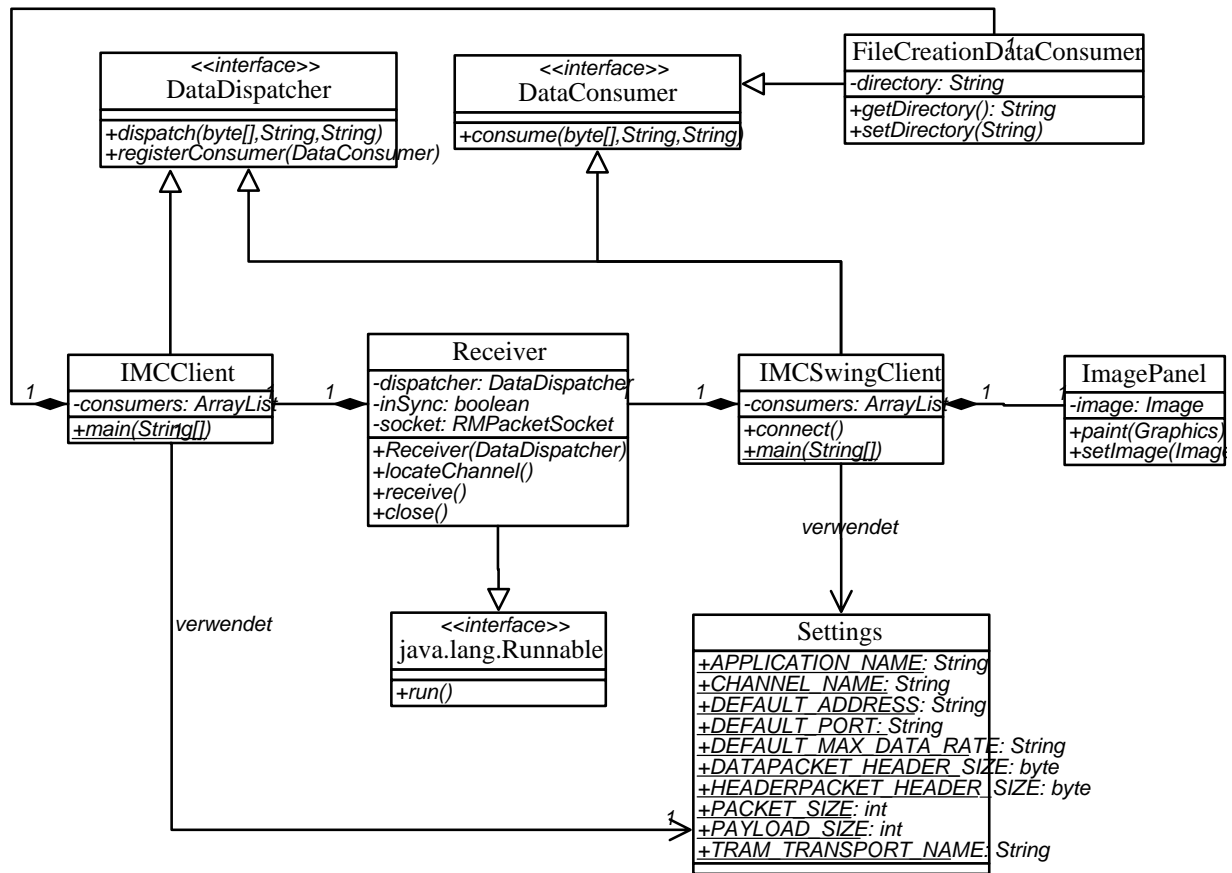


Abbildung 4: Client Klassendiagramm

Klassenbeschreibung (Übersicht):

DataConsumer	Klassen die dieses Interface implementieren können als Datenkonsument eingesetzt werden.
DataDispatcher	Klassen die dieses Interface implementieren können vom Receiver zur Verteilung der Daten eingesetzt werden.
FileCreationDataConsumer	Datenkonsument der Daten als Datei im Filesystem abspeichert.
ImagePanel	Panel zur skalierten Bilddarstellung.
IMCClient	NUR FÜR TESTZWECKE Client ohne GUI der empfangene Daten in Dateien speichert.
IMCSwingClient	SwingClient der einen Receiver startet und empfangene Pakete in einem frei skalierbaren Panel (ImagePanel) darstellt.
Receiver	Empfängt Daten per IP Multicast / TRAM Protokoll.
Settings	Konstanten die von Server und Client benutzt werden.

Tabelle 3: Client Klassenbeschreibung

Siehe Java-Doc für nähere Beschreibungen.

3.2 Sequenzdiagramme

3.2.1 Daten senden

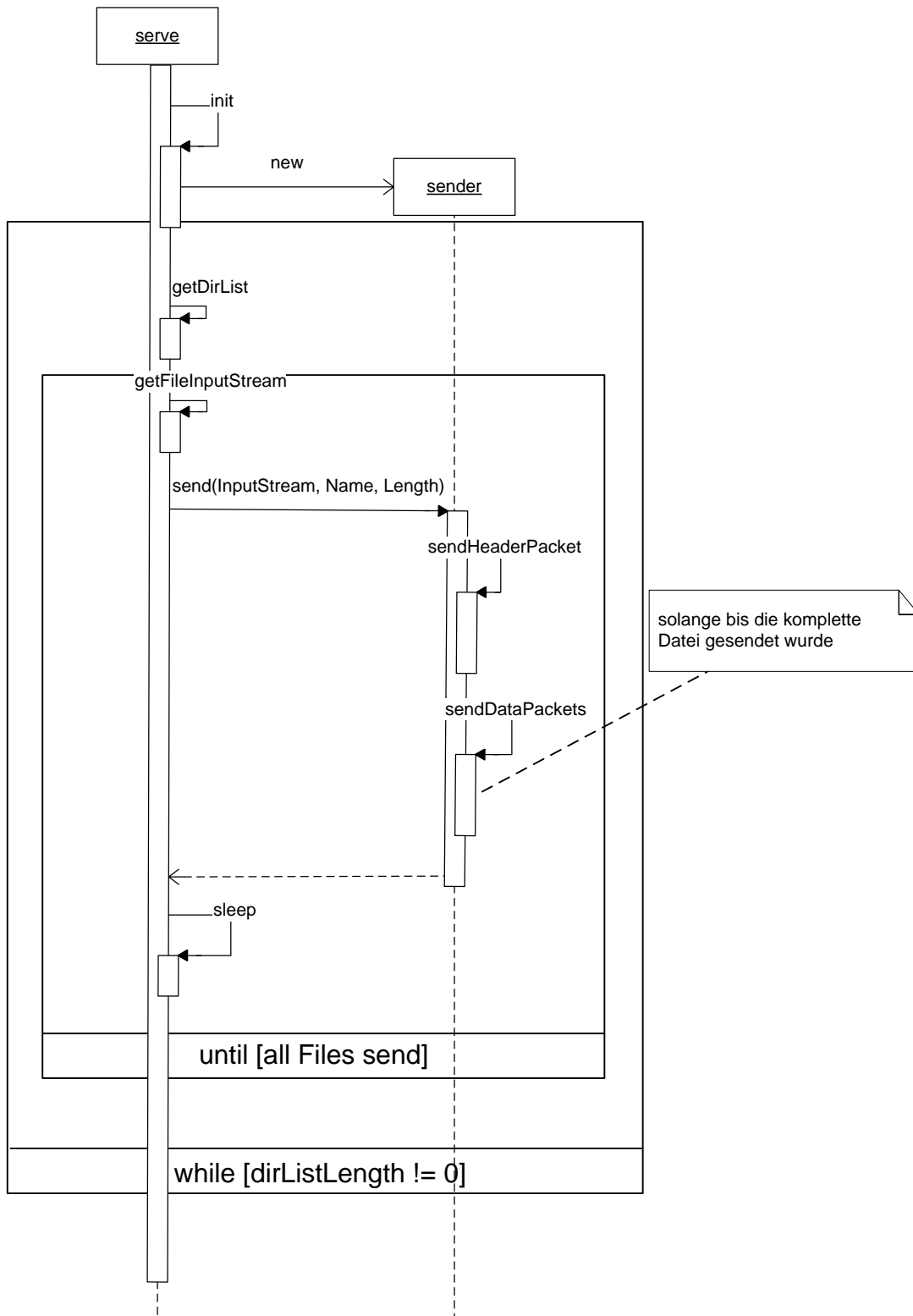


Abbildung 5: Sequenzdiagramm - Daten senden

3.2.2 Daten empfangen

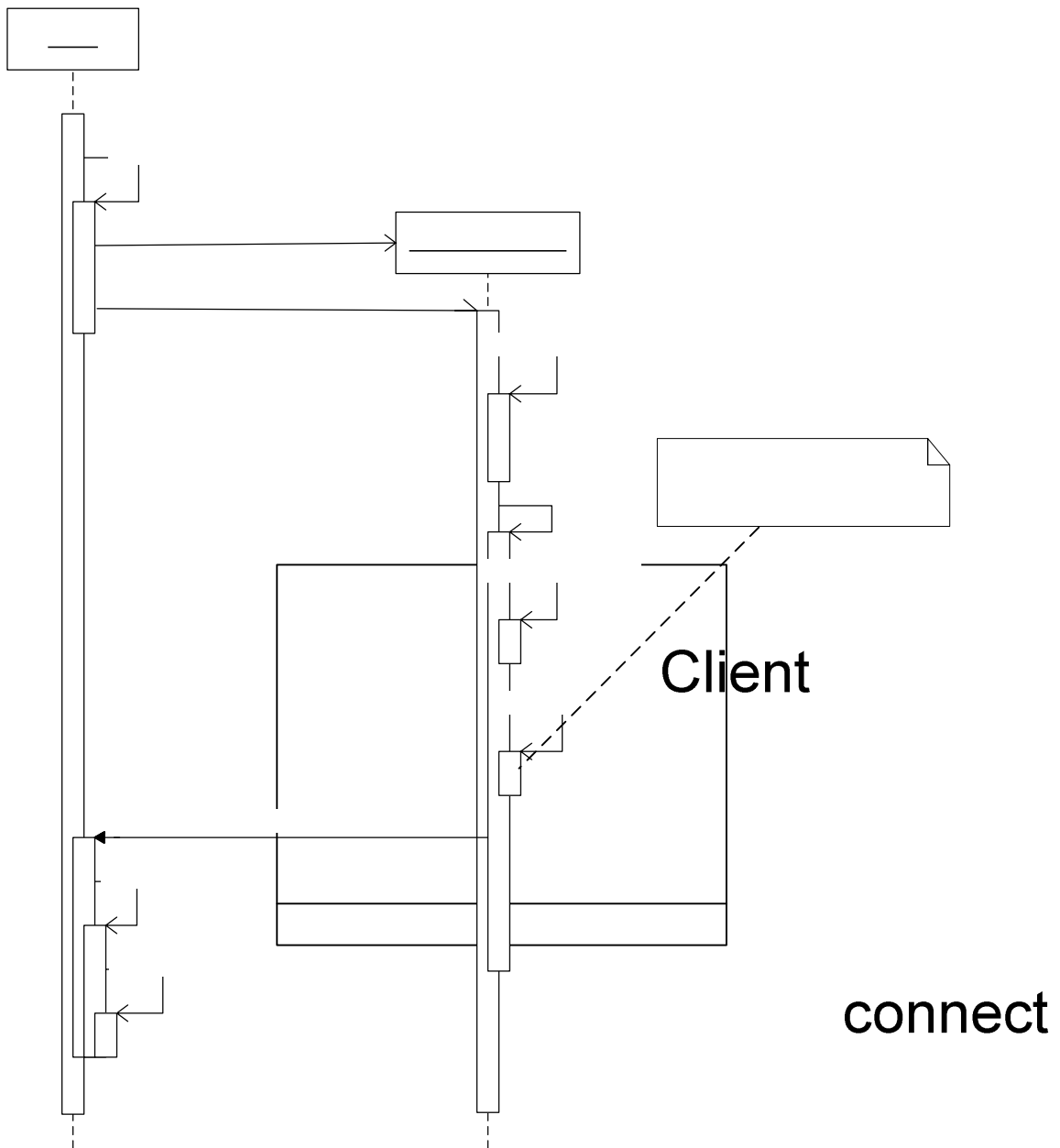


Abbildung 6: Sequenzdiagramm - Daten empfangen

new(D

4 Installation

Folgende Aufgaben müssen bei einer Installation der ImageMulticast-Software auf einem Windows Betriebssystem durchgeführt werden.

Pre-Installationsaufgaben

1. Systemvoraussetzungen prüfen.
 - a. Hardware: Minimalanforderungen Pentium 3; 300 MHz; 256MB RAM; 4MB an verfügbaren Harddisk-Speicher.

b. Software: Java Virtual Machine JDK 5.0 Update 5

2. Stellen sie sicher das der Pfad zu der ausführbaren Datei ‚*java.exe*‘ in ihrer *PATH* Umgebungsvariable eingetragen ist.

Installation

3. Entpacken sie die Datei *imc.zip* in einem Verzeichnis ihrer Wahl (dieses Verzeichnis wird im Folgenden als *HOME* Verzeichnis bezeichnet)

Post-Installationsaufgaben (nur für den Serverbetrieb notwendig)

4. Nehmen sie die Konfiguration des Systems durch bearbeiten der Datei *imc.properties* im Verzeichnis *HOME\conf* vor.

Beschreibung der Konfigurationsparameter:

Parameter	Beispiel	Wert
<code>server.imageDirectory</code>	<code>C:\\imc\\images</code>	Verzeichnis welches die zu übertragenden Bilder enthält. Hinweis: Bitte beachten sie die doppelten ‚\‘ im Pfad. (für Pfadangaben auf Windows-Systemen notwendig)
<code>server.imageFormats</code>	<code>jpg;gif</code>	Durch ‚;‘ getrennte bekannte Bildformate. Dateien die nicht eine dieser Endungen aufweisen werden nicht übertragen.
<code>server.intervall</code>	<code>10</code>	Intervall in Sekunden in dem Dateien gesendet werden.
<code>server.port</code>	<code>6824</code>	Serverport.
<code>server.address</code>	<code>239.100.100.224</code>	Klasse D IP Adresse. (239.0.0.0- 239.255.255.255) Wählen sie frei aus dem angegebenen Adressraum.
<code>server.maxDataRate</code>	<code>400000</code>	Maximale Übertragungsrage in Byte/Sekunde.
Steuerung des Servers via HTTP		
<code>server.httpManagement.hostname</code>	<code>localhost</code>	Hostname
<code>server.httpManagement.port</code>	<code>8080</code>	Portnummer
<code>server.httpManagement.userName</code>	<code>system</code>	Benutzername
<code>server.httpManagement.userPassword</code>	<code>manager</code>	Passwort

Tabelle 4: Konfigurationsparameter

5 Bedienung

Bei der Bedienung der Image Multicast Software muss zwischen dem Betrieb als Client mit einer grafischen Benutzerschnittstelle und dem als Server unterschieden werden. Die Parameter des Servers werden dabei per Konfigurationsdatei gesetzt. Näheres kann den beiden folgenden Abschnitten entnommen werden.

5.1 Client

Nach der erfolgten Installation befindet sich im HOME Verzeichnis ein Unterverzeichnis Namens ‚start‘ in dem sich 2 Skripts befinden um die Software im gewünschten Modus zu starten. Im Folgenden werden der Start und die Steuerung des Clients auf Grund eines Beispiels beschrieben.

Beispiel:

1. Start der Software im Client-Modus: Navigieren sie dazu in einer Eingabeaufforderung in das Verzeichnis *HOME\start* und führen sie die Datei *startClient.bat* aus.
2. Sind alle Systemvoraussetzungen (siehe Kapitel Installation) erfüllt erscheint folgendes Fenster auf ihrem Bildschirm.

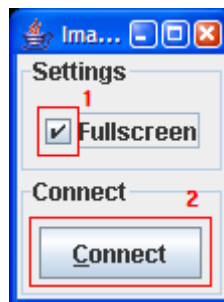


Abbildung 7 Client-Bedienelemente

Beschreibung der Bedienelemente:

- 1 ... Mittels der *Fullscreen*-Checkbox kann gewählt werden ob zur Darstellung der empfangenen Bilder der gesamte Bildschirm oder ein Fenster genutzt wird. Befindet man sich in einem der beiden Modi kann mittels Betätigung der *Escape* Taste in den jeweils anderen gewechselt werden.
- 2 ... Durch die Betätigung des *Connect*-Buttons wird versucht innerhalb von 2 Minuten den Server zu finden um sich danach auf den Datenstrom zu synchronisieren.
3. Betätigen der *Fullscreen* Checkbox (1) um empfangene Bilder in einem Fenster am Desktop darzustellen (der Haken ist nicht mehr zu sehen).
4. Klicken des *Connect*-Buttons (2) um eine Verbindung zum Server aufzubauen.
Hinweis: Bitte haben sie etwas Geduld, es kann bis zu einigen wenigen Minuten dauern bis der Kanal gefunden wurde und sich der Client gegebenenfalls auf einen bereits bestehenden Datenstrom aufsynchronisiert hat.
5. Anfangs, solange noch keine Bilder empfangen wurden, wird wie in der nächsten Abbildung ersichtlich der Linux-Pinguin im Darstellungsfenster angezeigt.

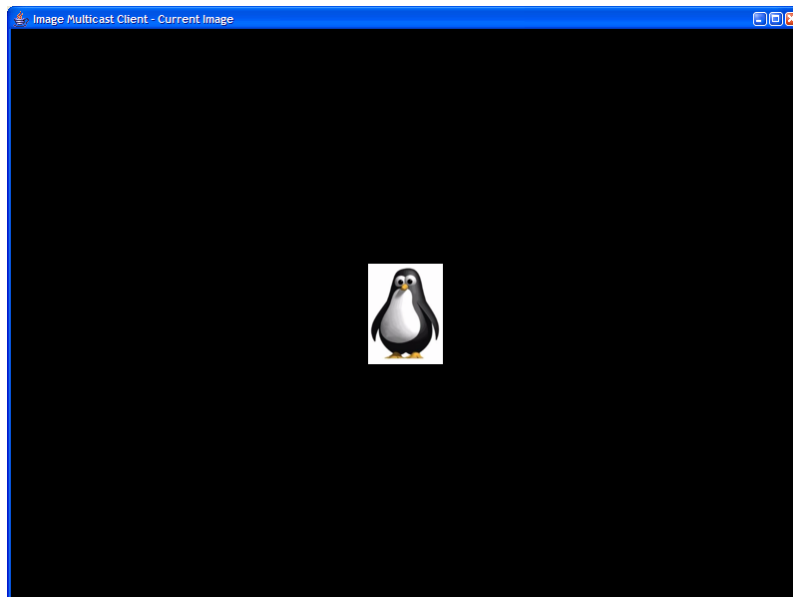


Abbildung 8: Anfangszustand des Darstellungsfensters.

6. Danach erfolgt die Darstellung der empfangenen Daten.

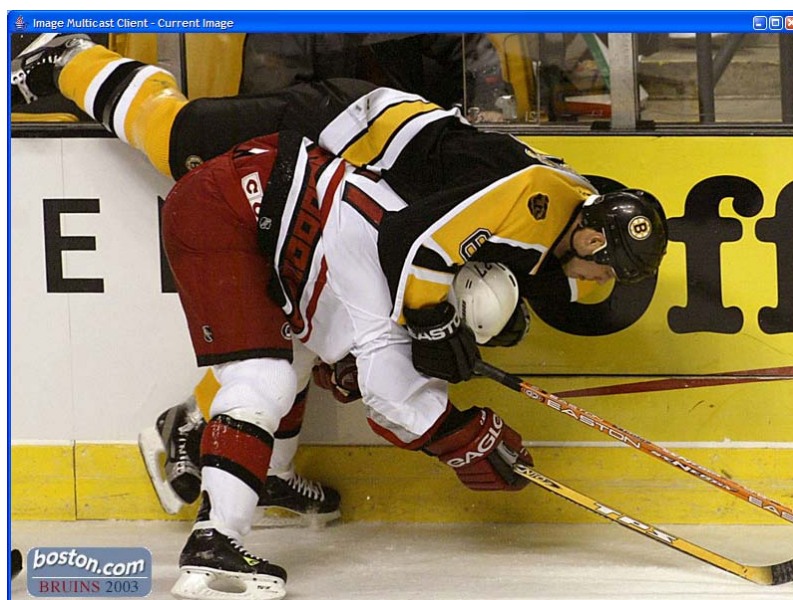


Abbildung 9 Beispiel für ein empfangenes Bild.

5.1.1 Log-Nachrichten

Der Status in dem sich der Client aktuell befindet ist aus den Log-Nachrichten in der Eingabeaufforderung ersichtlich.

Beispiel:

```
INFO 2005-11-27 17:03:07,281 imc.client.IMCSwingClient - IMCSwingClient
Version 1.0.0.0 initialized
INFO 2005-11-27 17:03:08,953 imc.client.Receiver - Locating
IMCChannel/ImageMulticast
INFO 2005-11-27 17:03:16,343 imc.client.Receiver - Located
IMCChannel/ImageMulticast in 7 seconds
INFO 2005-11-27 17:03:27,515 imc.client.Receiver - Expected data is Blaue
Berge.jpg with a length of 28521 bytes...
```

```
INFO 2005-11-27 17:03:29,578 imc.client.Receiver - Received data Blaue
Berge.jpg of size 28521 bytes in 1 seconds.
INFO 2005-11-27 17:03:30,671 imc.client.Receiver - Expected data is
bruins4_800600.jpg with a length of 135115 bytes...
INFO 2005-11-27 17:03:31,562 imc.client.Receiver - Received data
bruins4_800600.jpg of size 135115 bytes in 0 seconds.
```

Hinweis: Durch Änderungen in der Datei *HOME\conf\log4j.properties* kann diese Ausgabe z.B. auch in eine Datei erfolgen.

5.2 Server

Nach der erfolgten Installation befindet sich im HOME Verzeichnis ein Unterverzeichnis Namens ‚start‘ in dem sich 2 Skripts befinden um die Software im gewünschten Modus zu starten. Im Folgenden werden der Start und die Steuerung des Servers auf Grund eines Beispiels beschrieben.

Beispiel:

1. Konfiguration der Serverparameter: Durch editieren der Datei *HOME\conf\imc.properties* können Parameter wie z.B. das Verzeichnis in dem sich die zu übertragenden Bilder befinden oder welche Dateierweiterungen gültig sind verändert werden.
Eine vollständige Liste der Parameter inklusive Beschreibung kann dem Kapitel Installation entnommen werden.
2. Start der Software im Server-Modus: Navigieren sie dazu in einer Eingabeaufforderung in das Verzeichnis *HOME\start* und führen sie die Datei *startServer.bat* aus.
3. Sind alle Systemvoraussetzungen (siehe Kapitel Installation) erfüllt startet der Server und beginnt damit Status Nachrichten im Fenster der Eingabeaufforderung auszugeben.

5.2.1 Log-Nachrichten

Der Status in dem sich der Server aktuell befindet ist aus den Log-Nachrichten in der Eingabeaufforderung ersichtlich.

Beispiel:

```
INFO 2005-11-27 17:03:05,718 imc.server.Sender - Creating Channel
IMCChannel for Application ImageMulticast
INFO 2005-11-27 17:03:07,281 imc.server.management.HttpMngmt - initialized
and started.
INFO 2005-11-27 17:03:07,281 imc.server.IMCServer - IMCServer Version
1.0.0.0 initialized
INFO 2005-11-27 17:03:07,281 imc.server.IMCServer - Row row row your boat
...
HttpAdaptor version 3.0.1 started on port 8080
INFO 2005-11-27 17:03:07,500 imc.server.Sender - Receiver member Count is
0. Will try to send after 5 Second
INFO 2005-11-27 17:03:12,500 imc.server.Sender - Receiver member Count is
0. Will try to send after 5 Second
INFO 2005-11-27 17:03:17,500 imc.server.Sender - Receiver member Count is
0. Will try to send after 5 Second
```

```

INFO 2005-11-27 17:03:22,500 imc.server.Sender - Receiver member Count is
0. Will try to send after 5 Second
DEBUG 2005-11-27 17:03:27,500 imc.server.Sender - Header packet sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 1 sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 2 sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 3 sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 4 sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 5 sent...
DEBUG 2005-11-27 17:03:27,578 imc.server.Sender - Data packet 6 sent...
INFO 2005-11-27 17:03:27,578 imc.server.Sender - Sent data Blaue Berge.jpg
of size 28521 bytes in approx. 20.094 seconds.
DEBUG 2005-11-27 17:03:30,671 imc.server.Sender - Header packet sent...
DEBUG 2005-11-27 17:03:30,734 imc.server.Sender - Data packet 1 sent...
DEBUG 2005-11-27 17:03:30,734 imc.server.Sender - Data packet 2 sent...
DEBUG 2005-11-27 17:03:30,734 imc.server.Sender - Data packet 3 sent...

```

...

Hinweis: Durch Änderungen in der Datei `HOME\conf\log4j.properties` kann diese Ausgabe z.B. auch in eine Datei erfolgen.

5.2.2 Steuerung via HTTP

Via HTTP und der Nutzung der Java Management Extension kann der Server einfache Kommandos auch per HTTP empfangen.

Aktuell stehen folgende Kommandos zur Verfügung:

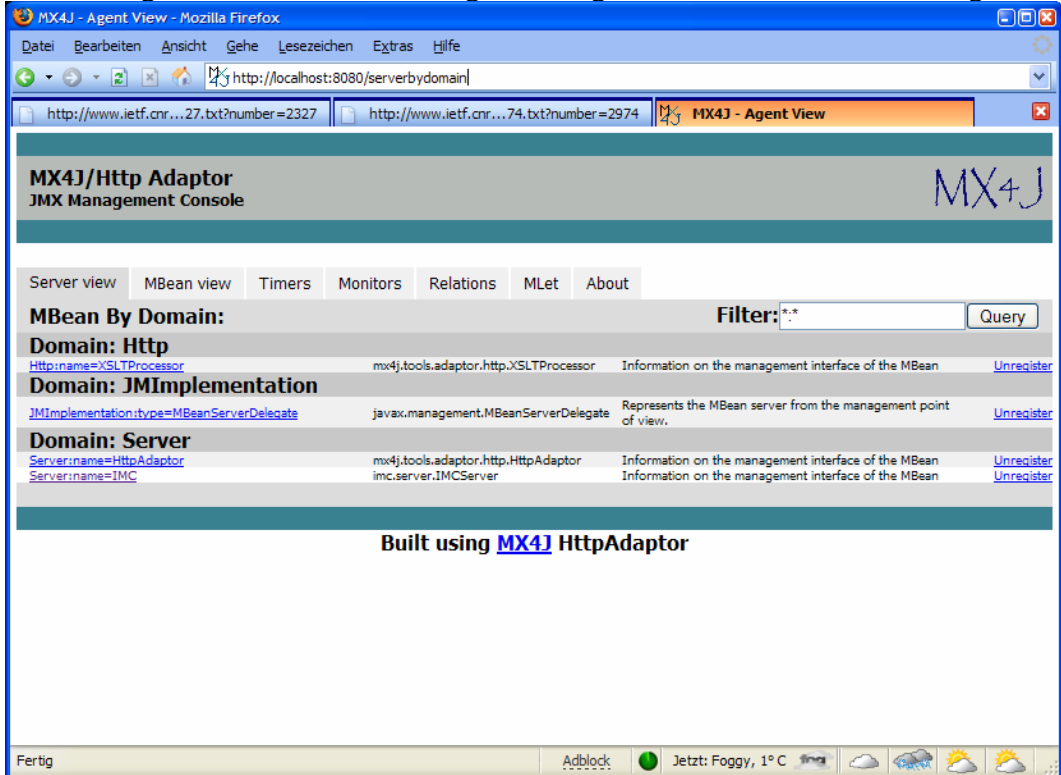
Kommando	Beschreibung
Kill	Beendet den Server ohne vorher reservierte Ressourcen „sauber“ frei zugeben.
Shutdown	Beendet den Server „sauber“.
Status	Liefert die aktuelle Anzahl an versendeten Dateien.

Tabelle 5: Managementkommandos

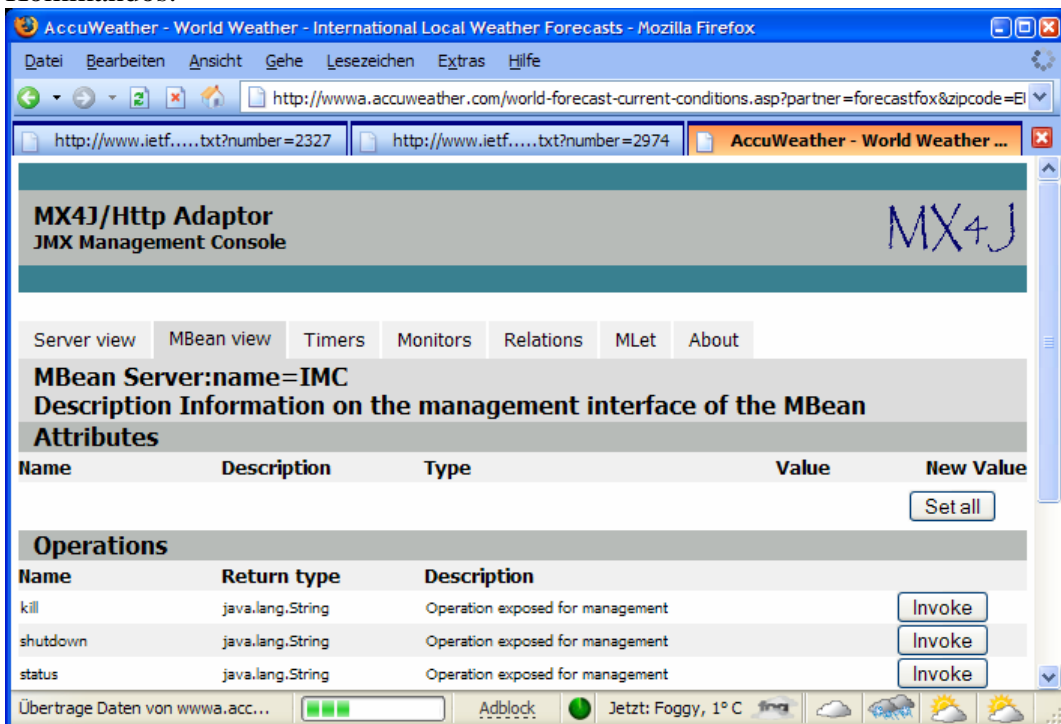
Beispiel:

1. Öffnen sie einen Internet-Browser (z.B. Firefox)
2. Geben die in der Adress-Leiste die Adresse des Servers an (in der Datei `imc.properties` konfigurierbar) z.B. <http://localhost:8080/>
3. Geben sie ihren Benutzernamen und ihr Passwort an.

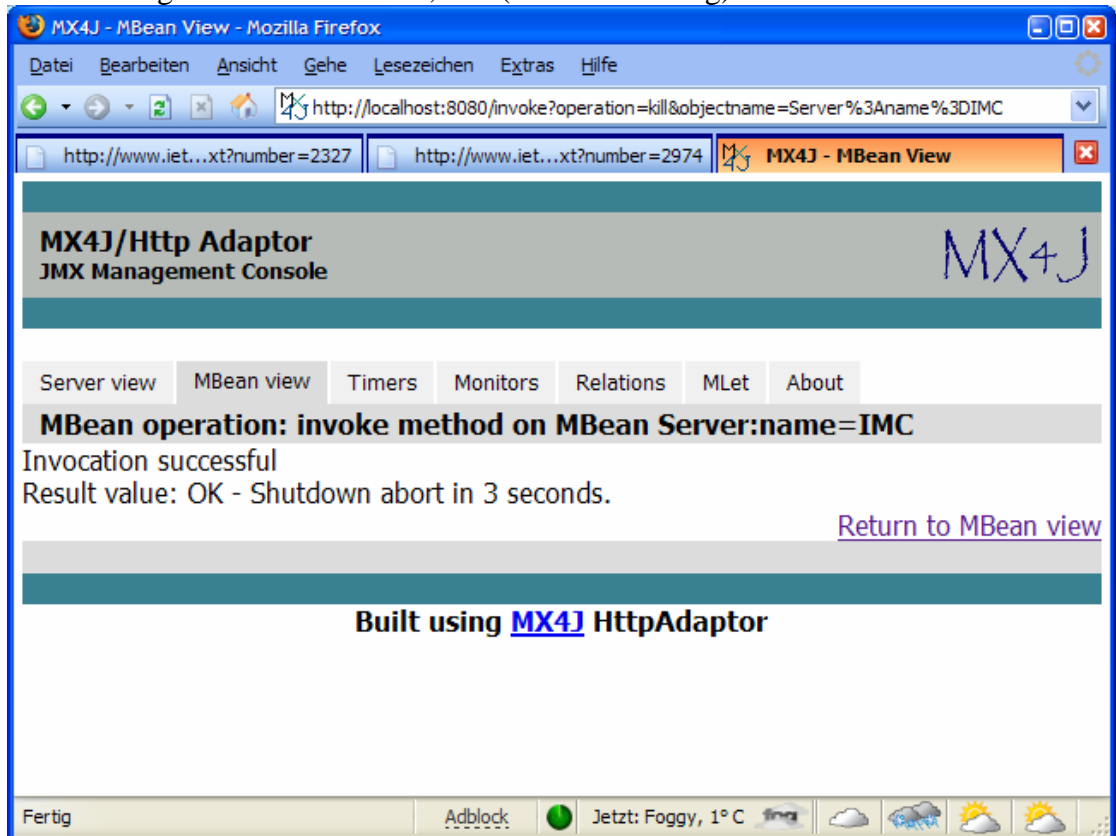
- Nach erfolgreicher Authentifizierung wird folgender Inhalt im Browser dargestellt.



- Durch Klicken auf `Server:name=IMC` gelangt man zu den IMC Server Kommandos.



6. Das jeweilige Kommando kann durch das Anklicken des zugehörigen Invoke-Buttons ausgeführt werden. z.B. ‚kill‘ (siehe Abbildung)



6 Wartung

Eine explizite Wartung der Software ist nicht erforderlich. Sollte die Applikation von einem neuen Server aus betrieben werden so sind auf diesen die Installationsschritte auszuführen.

Bei Problemen wenden sich bitte an folgenden Kontakt:

Name: Geith Alois

E-Mail: h9450190@wu-wien.ac.at

7 Referenzen

[CDKF02]

Cain, B.; Deering, S.; Kouvelas, I.; Fenner, B.; Thyagarajan, A.: Request for Comments: 3376 - Internet Group Management Protocol, Version 3.

<http://www.networksorcery.com/enp/rfc/rfc3376.txt>, Oktober 2002, Abruf am 03-12-2005.

[CHKW98]

Chiu, Dah Ming; Hurst, Stephen; Kadansky, Miriam; Wesley, Joseph: TRAM: A Tree-based Reliable Multicast Protocol.

http://www.experimentalstuff.com/sunr/techrep/1998/smlr_tr-98-66.pdf, Juli 1998, Abruf am 01-11-2005.

[Deer89]

Deering, S.: Request for Comments: 1112 - Host Extensions for IP Multicasting. <http://www.networksorcery.com/enp/rfc/rfc1112.txt>, August 1989, Abruf am 03-12-2005.

[HaJa98]

Handley, M.; Jacobson, V.: Request for Comments: 2327- SDP: Session Description Protocol. <http://www.ietf.cnri.reston.va.us/rfc/rfc2327.txt?number=2327>, April 1998, Abruf am 10-11-2005.

[HaPW00]

Handley, M.; Perkins, C.; Whelan, E.: Request for Comments: 2974 - Session Announcement Protocol. <http://www.ietf.cnri.reston.va.us/rfc/rfc2974.txt?number=2974>, Oktober 2000, Abruf am 10-11-2005.

[RoKH98]

Rosenzweig, Phil; Kadansky, Miriam; Hanna, Steve: The Java™Reliable Multicast™Service: A Reliable Multicast Library. http://research.sun.com/technical-reports/1998/smli_tr-98-68.pdf, September 1998, Abruf am 10-11-2005.