Bachelor Abschlusspräsentation - Analyse und Erweiterung der VoIP Funktionalität in Saros

Florian Pütz
Institut für Informatik
FU Berlin

25.11.2010

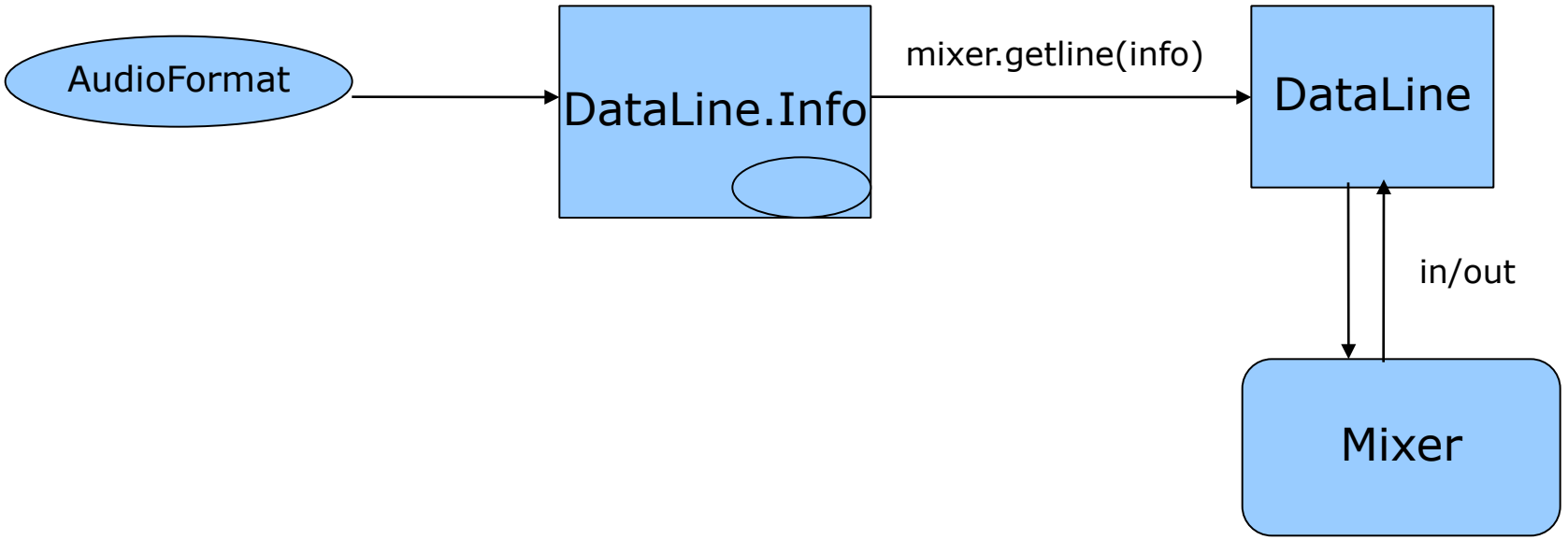
Gliederung

- Analyse
  - Hardcoded-Audioformat-Problem
  - Streamservice-Problem
  - Analyse der Latenzen
- Erweiterung: Konferenzfunktionalität
  - Implementierung eigener Strukturen
  - Auswertung von etablierten VoIP Standards
- Zusammenfassung
- Ausblick

Hardcoded-Audioformat-Problem: Beschreibung

- `AudioFormat`-Objekte zur Kapselung von hardware settings
  - zum Zweck der Initialisierung und Nutzung externer Audiogeräte

- Problem: Einstellungen fest im Quellcode niedergeschrieben
  - können bei Inkompatibilität zur Hardware eine `IllegalArgumentException` auslösen



interner Verbindungsaufbau des Audiomoduls zum Audiogerät (schematisch)

Hardcoded-Audioformat-Problem: Lösung

- Einführung eines Reliabilitätstests
  - erstellt temporäre Verbindung zum Audiogerät
  - Basis geschaffen durch das betroffene `AudioFormat`-Objekt
  - falls dabei keine Ausnahme auftritt, fortfahren
- Bereitstellen weiterer Audioformate
  - beinhalten alternative Einstellungen
  - falls der Reliabilitätstest fehlschlägt:
    - Alternative auswählen und Test erneut durchführen

Hardcoded-Audioformat-Problem: verbleibende Risiken

- Zusicherung: getestetes Objekt verursacht keine Inkompatibilitätsfälle

- Aber: Audioqualität nicht vorhersehbar
  - Testen der Wirkung nur durch praktische Anwendung möglich
  - Problem: Variieren der Auswirkungen auf verschiedenen Rechnern

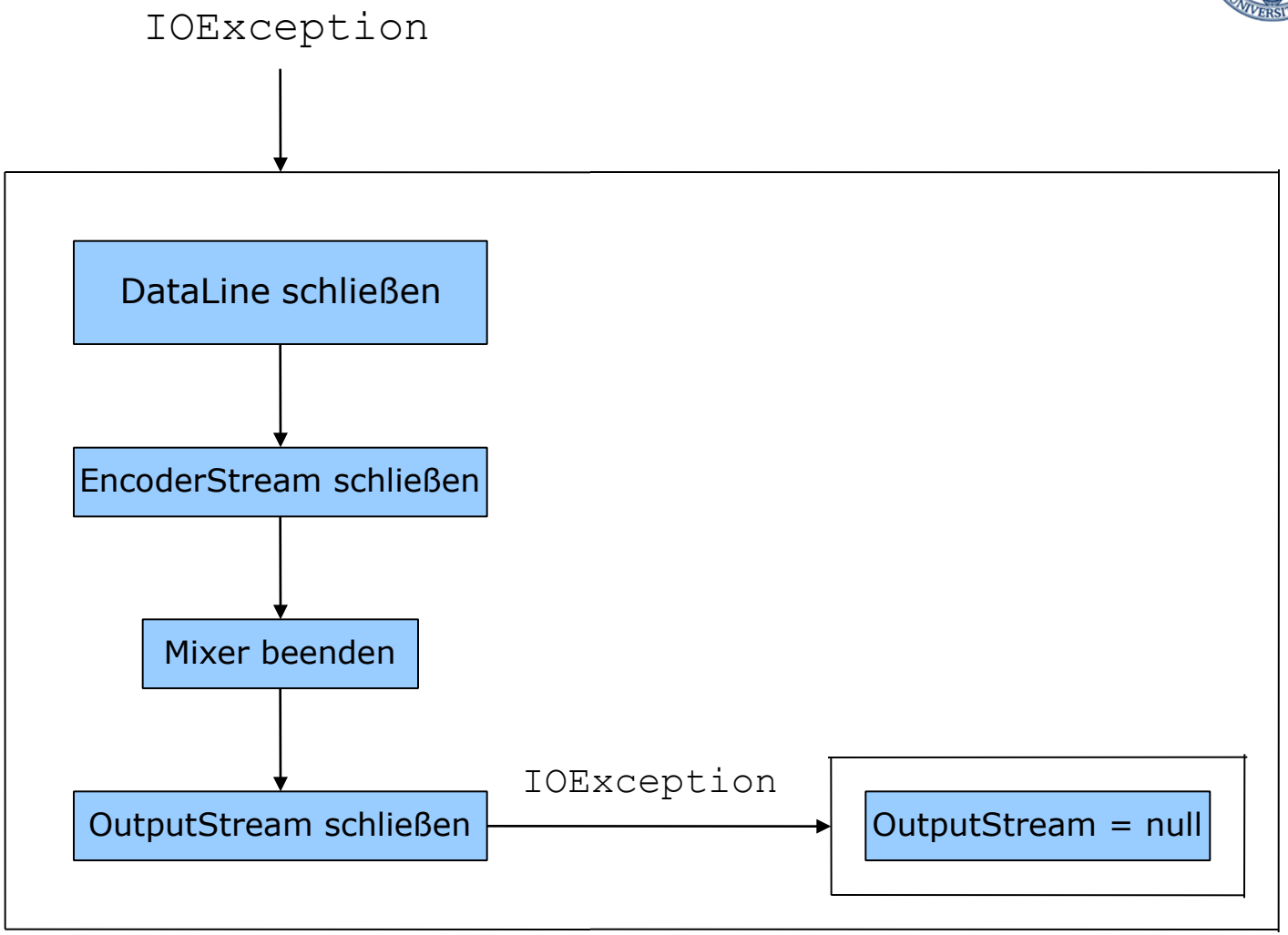
Streamservice-Problem Beschreibung

- Streamservice: Protokoll zum Datenaustausch über das Internet mit Saros
  - ursprünglich für das Screensharing konzipiert
- Problem: Ausnahmefall während der Terminierung einer aktiven Sprachverbindung
  - `OutputStream is closed`
  - Outputstream bereitgestellt durch Streamservice

Streamservice-Problem: Lösung

- Ziel: zuverlässiges Schließen der beteiligten Streams garantieren

- zu diesem Zweck: Einführen eines `finally`-Blocks
  - Aufgabe: im Falle einer Ausnahme beim Beenden der Sprachverbindung Schließen der Streams erzwingen



Routine zur Terminierung der Sprachverbindung im Fall einer `IOException`

Streamservice-Problem: Resultat der Lösung

- Problem besteht weiterhin
- Schließen des Outputstreams noch vor Terminierung
- Folgerung: Das Problem ist dem Streamservice zuzuordnen.

Latenzen: Beschreibung

- Übertragung eines Sprachsignals: bis zu 2 Sekunden
  - Testszenario: lokale Umgebung
  - besteht aus 2 Programminstanzen auf einem Rechner
- Ansätze zur Analyse: direkte und iterative Manipulation der Pakete
  - Streamservice
  - Implementierung des Audiosystems
  - Verwendung des JSpeex Codecs

Latenten Lösungsansatz: Implementierung des Soundsystems

- Soundsystem Aufgaben:
  - Initialisieren der beteiligten Geräte
  - Aufnahme und Wiedergabe des Gesprochenen

- Ansatzpunkt: Stellen der direkten Manipulation von Sprachpaketen
  - iteratives Aufnehmen und Wiedergeben der Daten
  - Aufnahme Ablauf: Eingabe der Daten über das Mikrophon
  - Kodieren der Daten zur Verwendung auf digitaler Ebene
  - respektive Wiedergabe: dekodieren und ausgeben

Latenzen Lösungsansatz: Die Nutzung des Streamservices

- zentrale Komponente für Versand und Empfang von Sprachdaten
- direkt beteiligt am iterativen Lese- und Schreibprozess
- Vorschlag: temporärer Austausch des Streamservices gegen ein etabliertes VoIP Protokoll
  - Ziel: ermitteln, ob die Nutzung des Streamservices Latenzen verursacht

Latenzen Lösungsansatz: Verwendung von Speex und dessen Javaportierung JSpeex

- für die Kodierung und Dekodierung der Signale verantwortlich
- ebenfalls direkt am Lese- und Schreibprozess beteiligt
- Austausch des Codecs gegen eine Freeware bzw. OpenSource Alternative
  - für Testzwecke geeignet: iLBC Codec unter einer eigenen Freeware Lizenz

Konferenzfunktionalität: Entwurf eigener Strukturen

- Streamservice als Protokoll: nur im Saros Projekt bekannt
  - folglich: keine freie Lösung auf Basis des Streamservices verfügbar
- Module des VoIP Moduls sind selbst eigene Entwicklungen
  - d.h. insbesondere: befolgen eine eigene Architektur

Konferenzfunktionalität:
Entwurf eigener Strukturen - Beschreibung

- Ziele des Entwurfs:
  - zentrales Erfassen aller Teilnehmer
  - Anbinden aller Teilnehmer an den Streamservice
  - host-gesteuertes System

- Host: Wahl und Aufgaben
  - ausgehende Signale zunächst an Host schicken
  - Host verteilt Signale an alle anderen Teilnehmer
  - zunächst geplant: Host der Saros Sitzung ist Host der VoIP Sitzung
    - später: Auswahl nach bestimmten Kriterien (Bandbreite, Latenz)

Konferenzfunktionalität:
Entwurf eigener Strukturen - Beschreibung

- Realisieren der Erfassung mithilfe der Datenstruktur `AudioMulticomManager`
  - basiert auf einer HashMap
  - soll einfachen Zugriff über numerische Keys ermöglichen
  - Kontrolle liegt beim `AudioServiceManager`
- gespeichert werden modifizierte `User` Objekte
  - Basis dazu ist in Saros bereits enthalten

Konferenzfunktionalität: Entwurf eigener Strukturen – Verwerfen

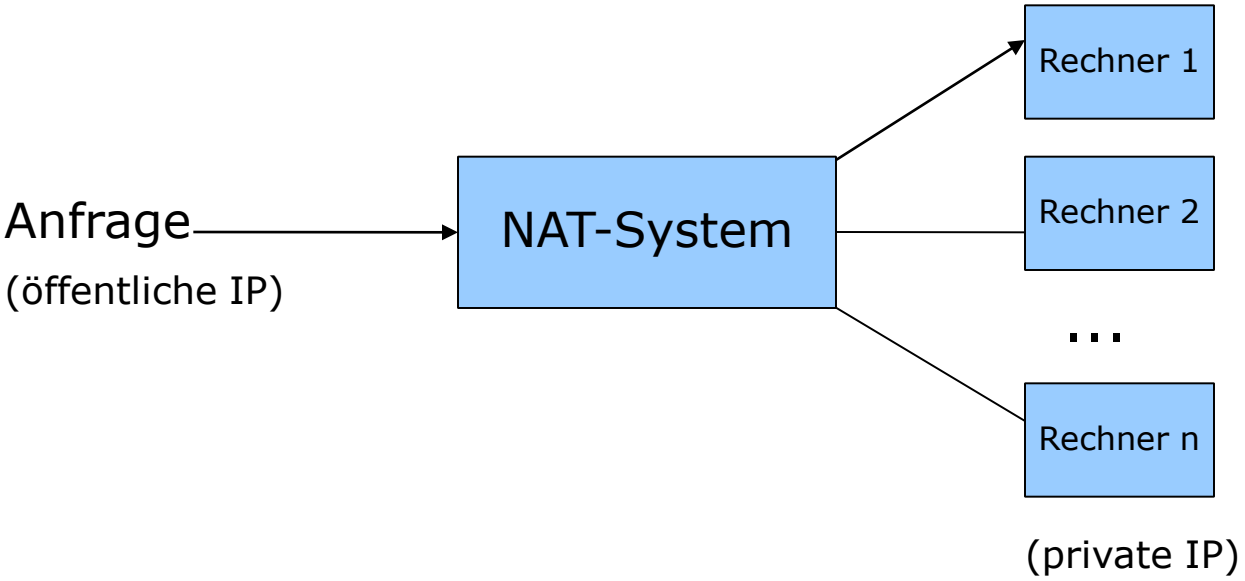
- Latenzproblem besteht weiterhin
  - weitet sich auf die Konferenzkommunikation aus
- Nutzung verfügbarer Lösungen liefert:
  - alternatives Übertragungsprotokoll
    => löst das Streamservice-Problem
  - bereits implementierte Konferenzfunktionalität

Evaluation etablierter VoIP Software

- Evaluationskriterien:
  - Verfügbarkeit einer Java Implementierung
  - rechtliche Absicherung: Lizenzkompatibilität zur GPL 2.0
  - vorhandene Konferenzfunktionalität
  - Lösung des NAT-Problems

- NAT: Verfahren zur Abbildung einer öffentlichen Adresse auf mehrere Geräte in einem Netzwerk
  - Verteilung interner IP Adressen
  - Steuerung über NAT Router (Hardware) oder NAT Software

Evaluation etablierter VoIP Software: NAT Problem



NAT Traversal (schematisch)

Evaluation etablierter VoIP Software: Eclipse Communication Framework

- Zweck: Schnittstelle für Implementierung von Kommunikationsstrukturen
  - Beispiele: Point-to-Point, Publish and Subscribe
- ECF bietet: Spezifikationen und Implementierungen von Strukturen
- vorzustellende Lösung implementiert Call API des ECFs
  - Spezifikation von Telefoniesitzungen

Evaluation etablierter VoIP Software: ECF basierte VoIP Lösung

- während des „Google Summer of Code 2007“ entstanden
- verwendet Jingle als Protokoll
  - basierend auf der Smack API
- Audiosystem bereitgestellt durch das Java Media Framework

Evaluation etablierter VoIP Software: ECF basierte VoIP Lösung - Probleme

- ECF muss zusätzlich Installiert werden
- Jingle: keine Unterstützung für Konferenzschaltung
  - erstes Ausschlusskriterium
- Lizenzlage: Lösung steht unter Eclipse Public License (EPL)
  - aufgrund des schwachen Copyleftes inkompatibel zur GPL 2.0
  - zweites Ausschlusskriterium

Evaluation etablierter VoIP Software: Skype4Java

- Kommunikation per Skype bereits in Saros vorhanden
- Zweck der Evaluierung: Einblick in die API erhalten
  - im Hinblick auf künftige Möglichkeiten mit SkypeKit
- Arbeitsweise der API: Abbilden von Befehlen auf Objekte, Attribute und Methoden

Evaluation etablierter VoIP Software: Skype4Java - Nachteile

- Vorteil: einfache Handhabung der API
- Nachteile:
  - ähnliche Funktionalität bereits in Saros integriert
  - Lizenzlage: Skype4Java steht unter Apache License 2.0
    - inkompatibel zur GPL 2.0

Evaluation etablierter VoIP Software: H.323

- Spezifikation eines paketbasierten Übertragungsprotokolls
  - herausgegeben durch die International Telecommunication Union (ITU)
  - geschaffen für die Bereiche der Audio- und Videokommunikation
- Vorteil: hohe Erweiterbarkeit durch Kompatibilität zu weiteren Spezifikationen der ITU

Evaluation etablierter VoIP Software: H.323 - Probleme

- NAT Traversal nur für Videoübertragung möglich
  - Spezifikation H.460

- keine OpenSource Software API in Java vorhanden
  - Anfrage an JAIN Initiative 2000 scheiterte
  - proprietäre Lösung: J323 von IBM

Evaluation etablierter VoIP Software: Session Initiation Protocol

- 2002 herausgegeben durch die Internet Engineering Task Force (IETF)
  - Zweck: Spezifizierung von Kommunikationssitzungen
- Kommunikation über Klartextnachrichten
  - ähnlich aufgebaut wie HTTP
- Datenübertragung mithilfe von RTP
  - ebenfalls herausgegeben durch die IETF

Aufbau einer SIP Request Message[1]

Evaluation etablierter VoIP Software: SIP und SDP

- Session Description Protocol
  - weitere Spezifikation des IETF
  - 2006 erschienen

- zur Erweiterung des Bodys
  - zusätzliche Informationen zB. über verwendete Medien und Techniken zur Verschlüsselung

Evaluation etablierter VoIP Software: Referenzimplementierung JAIN-SIP

- herausgegeben durch Initiative „Java APIs for Integrated Networks“
  - inzwischen mehr als 80 Firmen beteiligt
  - stellt APIs für Telekommunikationsanwendungen
- JAIN-SIP Projekt:
  - gegründet durch „National Institute of Standards and Technologies“ USA
  - implementiert SIP, SDP und weitere Standards
  - Public Domain

Evaluation etablierter VoIP Software: Das SIP-Communicator Projekt

- Instant Messaging und VoIP Client
  - nutzt iptel.org als VoIP Provider
  - ermöglicht insb. NAT Traversal
  - verfügt über Konferenzfunktionalität
- greift auf JAIN-SIP API zurück
  - zum Aufbau des SIP Stacks für VoIP Kommunikation
- steht unter LGPL

Evaluation etablierter VoIP Software: SIP Vor- und Nachteile

- Vorteile:
  - Erweiterbarkeit durch zusätzliche Spezifikationen des IETF
  - mit JAIN-SIP: staatlich geförderte Referenzimplementierung
    - beinhaltet bereits Erweiterungen zu SIP
  - in Saros integrierbare VoIP-Lösung vorhanden durch den SIP-Communicator
- Nachteil: keine integrierte Lösung für das NAT-Problem
  - Lösung an Provider delegiert

Evaluation etablierter VoIP Software: IAX

- IAX: Übertragungsprotokoll für Asterisk
  - Software Telefonanlage

- Zweck von Asterisk: Annahme und Weiterleitung von Anrufen
  - an weitere Asterisk Telefonanlagen oder an Endgeräte
  - erfolgt anhand des Dial Plans
    - Tabelle zur Steuerung der Telefonanlage

Evaluation etablierter VoIP Software: Gründe zur Entscheidung für IAX

- mit NJIAX vollständige Java API
  - herausgegeben von Nomasystems
  - steht unter LGPL 2.1
- klare Zielsetzung: Unterstützung für NAT-basierte Netzwerke durch Port Forwarding
- Konferenzfunktionalität wird unterstützt
- Nachteil: Betrieb einer externen Asterisk Telefonanlage

Evaluation etablierter VoIP Software:
IAX Prototyp – Implementierung und Auswertung

- Aufgabe: Implementierung eines Anrufsignals
- zu diesem Zweck: Implementieren eines Audiosystems
  - Schnittstelle von NJIAX vorgegeben
  - Funktionalität dem VoIP Modul von Saros entliehen
- 2. Schritt: Aufbau eines VoIP Clients
  - erhält einen `Peer` als Repräsentation eines Teilnehmers
    - beinhaltet Benutzername, Passwort, IP Adresse der verwendeten Asterisk Telefonanlage

Evaluation etablierter VoIP Software: IAX Prototyp – Implementierung und Auswertung

- letzter Schritt: Konstruktion der Funktionalität eines Anrufes
  - Initialisierung eines `NewCall` Objektes
    - Repräsentation eines NewCall Signals innerhalb der NJIAX API
    - Parameter: Telefonnummer des Angerufenen, `Peer` Objekt des Anrufers
  - Initialisieren und Starten der Audiokomponenten

- Probleme:
  - Nutzung des Audiosystems unklar
  - Asterisk: Verbindungsversuche werden zurückgewiesen

Evaluation etablierter VoIP Software:
IAX Prototyp – Abschluss

- Generieren und Versenden eines Anrufsignals möglich

- verbleibende Schritte:
  - Konfiguration des Dial Plans
  - Korrektur des Audiosystems
  - Implementierung: Routine zur Annahme von Anrufsignalen

- abschließend: IAX-basierte Lösung realisierbar

Zusammenfassung

- Analyse: drei Probleme bearbeitet
  - Hardcoded-Audioformat-Problem: Lösungsvorschlag gegeben
  - Streamservice-Problem: konnte nicht im VoIP-Modul lokalisiert werden
  - Latenzen: drei kritische Abschnitte im Modul vorgestellt
- Erweiterung:
  - eigene Idee vorgestellt
  - fünf etablierte VoIP-Technologien ausgewertet
    - IAX praktisch untersucht
    - SIP und IAX befürwortet

Ausblick

- weitere Schritte abhängig von gewählter Basis
- auf Basis von SIP: Auswertung einer VoIP-Lösung mittels des SIP-Communicators
  - Aufbau einer Zusammenarbeit mit dem Projekt SIP-Communicator
- auf Basis von IAX: Auswertung von Clients, die nicht in Java implementiert wurden
  - Beispiel: YATE (C++)

Vielen Dank!

Bilderverzeichnis

[1] Oracle Blogs - Enterprise Tech Tips

http://blogs.sun.com/enterprisetechtips/resource/SipMessage.jpg ,

zuletzt zugegriffen am 14.11.2010

Literaturverzeichnis

Website des iLBC Projektes, von: iLBCfreeware, Global IP Solutions, Stand: 2007, http://ilbcfreeware.org/ , letzter Zugriff: 26.07.2010

"Frequently Asked Questions about the GNU Licenses", von: Project, Free Software Foundation, Stand: 03.07.2010, http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility , letzter Zugriff: 26.07.2010

"Frequently Asked Questions about the GNU Licenses", von: Project, Free Software Foundation, Stand: 03.07.2010, http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility , letzter Zugriff: 26.07.2010

"Java Specification Requests – JSR81: JAIN H323", von: Orit Levin, Stand: 2000 http://jcp.org/en/jsr/detail?id=81 , letzter Zugriff: 26.07.2010

"JAIN General Q&A", von: Oracle, Sun Developer Network, Stand: 2010, http://java.sun.com/products/jain/qa.html , letzter Zugriff: 26.07.2010

SIP-Communicator Website, von: Emil Ivov et.al, Stand: 2010, http://sip-communicator.org/ , letzter Zugriff: 26.07.2010

njiax Projektwebsite, von: Nomasystems, mrrubinos, mpquique, emiliano.rguez, Stand: 2007, http://code.google.com/p/njiax/ , letzter Zugriff: 26.07.2010

Skype4Java Projektseite, von: Koji Hisano et al., Stand: 24.07.2010 http://de.sourceforge.jp/projects/skype/ , letzter Zugriff: 26.07.2010

"RFC 3261 – SIP: Session Initiation Protocol", von: J. Rosenberg et al, Internet Engineering Task Force, Stand: Juni 2002, http://tools.ietf.org/html/rfc3261 , letzter Zugriff: 26.07.2010

"RFC 4566 – SDP: Session Description Protocol", von: M. Handley, V. Jacobson, C. Perkins, Internet Engineering Task Force, Stand: Juli 2006, http://tools.ietf.org/html/rfc4566 , letzter Zugriff: 26.07.2010