Java-based Personal Communication Management

Dipl.-Ing. Peyman Farjami, Dipl.-Ing. Stefan Kluwe,
Lehrstuhl für Kommunikationsnetze RWTH-Aachen
pemi@comnets.rwth-aachen.de

Kurzfassung

Durch die weltweite Verfügbarkeit des Internet und die Plattformunabhängigkeit der Java-Technologie ergeben sich für die Mobilität in Telekommunikationsnetzen neue Perspektiven. Am Beispiel des Intelligenten Kommunikationsmanagers (IKM) wird die Nutzung dieser Möglichkeiten für die Realisierung einer adaptiven persönlichen Kommunikationsumgebung erläutert. Somit ist der Benutzer dieses Systems in der Lage, jederzeit, jederorts und plattformunabhängig auf die Dienste des IKM über die gewohnte graphische Benutzeroberfläche (GUI) zuzugreifen, um z.B. einen eingehenden Anruf/Fax online übers Internet zu seinem aktuellen Anschluß weiterzuleiten oder sein Benutzerprofil aus der Ferne an seine aktuellen Anforderungen anzupassen.

1 Einleitung

Über die ständige Weiterentwicklung der Architektur der drahtgebundenen und drahtlosen Telekommunikationsnetze und deren Dienste hinaus, hat sich in den letzten Jahren das Internet als ein weltweit verfügbarer Dienst etabliert. Eine rapide steigende Zahl von Nutzern verwendet das Internet für den Weltweiten Zugriff auf multimediale Information sowie für den Versand multimedialer Dokumente mit Hilfe elektronischer Post.
Resultierend aus dem Angebot einer Vielzahl neuer Dienste und dem Ausbau der digitalen Mobilfunknetze, wird dem Telekommunikationsteilnehmer ermöglicht völlig orts- und zeitunabhängig erreichbar zu sein. Mit der wachsenden Mobilität verliert der Einzelne jedoch den Einfluß auf seine Erreichbarkeit. Eine mögliche Lösung dieser Problematik ist es, dem Teilnehmer die Kontrolle über seine örtliche bzw. zeitliche Erreichbarkeit zu überlassen. Um dem Telekommunikationsteilnehmer zu diesem Zweck umfassende Möglichkeiten zur Organisation und automatisierten Handhabung der persönlichen Kommunikation zu bieten, wurde am Lehrstuhl für Kommunikationsnetze der RWTH Aachen der IKM (Intelligenter Kommunikationsmanager) entwickelt. Diese persönliche Kommunikationsumgebung integriert die anrufer/absender-spezifische Behandlung von Diensten wie Sprache, Fax, Pager/SMS-Nachrichten sowie Email in einer einzigen Anwendung. Dieses persönliche Kommunikationsmanagement basiert auf der Festlegung bestimmter Kriterien in den sogenannten Benutzerprofilen (Filterskripten, Userprofiles).
Um dem Benutzer mehr Mobilität und Flexibilität zu ermöglichen, muß er in der Lage sein, auch aus der Ferne auf die Dienste des IKM zuzugreifen. Bedingt durch den großen Funktionsumfang eines solchen Kommunikationsmanagers ist zur Konfiguration eine graphische Benutzeroberfläche (GUI) erforderlich. Die Probleme für eine Fernabfrage bzw. Fernkonfiguration über ein GUI ergeben sich durch die Restriktionen wie Hardware-, Betriebssystem- (UNIX, Windows, Macintosh ...) und Netz-Abhängigkeit (PSTN, ISDN, B-ISDN ...) sowie die Installation der Remote-Software auf der entsprechenden Plattform vor jedem Fernzugriff.
Vor diesem Hintergrund wurde die Plattformunabhängigkeit der Java-technologie in Verbindung mit dem Internet als ein netzübergreifender Dienst zur Aufhebung der obengenannten Einschränkungen eingesetzt. Somit kann der Benutzer des IKM jederzeit und jederorts, plattformunabhängig über die gewohnte graphische Benutzeroberfläche den gesamten Funktionsumfang des IKM in Anspruch nehmen. Hierfür benötigt er lediglich einen java-fähigen Web-Browser und einen beliebigen Internet-Zugang.

2 Java-Technologie

Java wurde Anfang der 90er Jahre ursprünglich entwickelt, um in Unterhaltungsgeräten wie Fernsehern und Videorecordern eine flexible Programmierung und eine Kommunikation der Geräte untereinander zu realisieren. Erst im Zuge der zunehmenden Verbreitung des Internets und des World-Wide-Web (WWW), ab 1994, wurde Java zu einer Programmiersprache für das Internet weiterentwickelt, deren Code auf den verschiedensten Plattformen ohne Änderung ausführbar ist. Das bedeutet, daß der Quelltext bei der Compilierung nicht in eine plattformabhängige Maschinensprache (die die prozessorspezifischen Befehle enthält), sondern in einen plattformunabhängigen *Bytecode* übersetzt und in den sogenannten Java-

Klassen gespeichert wird. Dieser Bytecode wird erst unmittelbar vor der Ausführung durch den Java-Interpreter in die entsprechenden Maschinenbefehle der jeweiligen Plattform umgesetzt, siehe Abbildung 1.

Die Entwicklung einer Java-Anwendung kann auf zwei verschiedene Arten erfolgen, die festlegen auf welche Weise die Anwendung später ausgeführt werden soll:

- **Standalone-Anwendungen** können über einen Java-Interpreter wie ein ausführbares Programm von der Kommandozeile gestartet werden.
- **Applets** werden mittels eines Web-Browsers über das Internet angefordert und mit Hilfe des HTTP-Protokolls zum entfernten Host übertragen. Aufruf und Ausführung finden innerhalb des Browsers statt, der eine virtuelle Java-Maschine beherbergt.

Die Ausführung und Interpretation des Bytecodes übernimmt dabei eine sogenannte virtuelle Maschine (VM). Für jede unterstützte Hardwareplattform/jedes Betriebssystem ist die Implementierung einer virtuellen Maschine erforderlich, die den plattformunabhängigen Bytecode ausführt. Des weiteren besitzt Java ein System zur automatischen Speicherverwaltung, das auf Objekten und auf deren Referenzen beruht. Durch die sogenannte *Automatic Garbage Collection* wird der nicht referenzierte Speicher automatisch freigegeben.

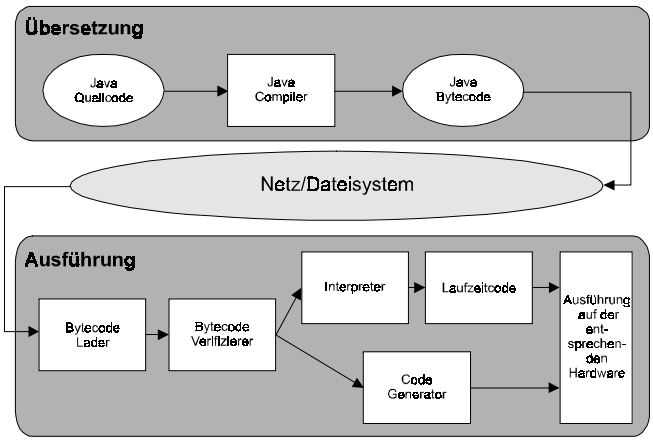

Abbildung 1: Der Weg vom Quellcode zur Ausführung

2.2 RMI - Remote Method Invocation

Neben der Übertragung von Daten und Dateien, ist es auch möglich Netze zur Interprozesskommunikation zu verwenden. Die Interprozesskommunikation bietet den auf verschiedenen Hosts ausgeführten Prozessen die Möglichkeit miteinander zu kommunizieren. Eine weit verbreitete Variante der Interprozesskommunikation ist Remote Procedure Call (RPC). Durch den RPC erfolgt eine Abstraktion der Kommunikationsschnittstelle auf Prozedurebene, d.h. die Art und Weise wie zwei Hosts miteinander kommunizieren bleibt dem Anwendungsentwickler durch den RPC verborgen. Der Aufruf einer Prozedur auf einem anderen Host unterscheidet sich für den Entwickler (fast) nicht von einem lokalen Prozeduraufruf. Bei verteilten Prozessen fehlt ein gemeinsamer Adreßraum, der sonst für den Austausch von Argumenten dient. Argumente werden über das Netz an den entfernten Host gesandt, dort wird die Prozedur ausgeführt und die Rückgabewerte an den aufrufenden Host übergeben. In objektorientierten Systemen kommunizieren Objekte miteinander, statt Prozeduren werden die mit dem Objekt verbundenen Methoden aufgerufen. Für die Kommunikation zwischen Objekten in verschiedenen Adreßräumen ist somit auch eine objektorientierte Variante der Interprozesskommunikation erforderlich. Für Java wurde ein System zum Methodenaufruf in Systemen verteilter Objekte namens Remote Method Invocation (RMI) entwickelt. Die RMI wurde unter anderem mit dem Ziel entwickelt den Aufruf entfernter Methoden für den Entwickler so transparent zu gestalten, daß er sich nicht von einem Aufruf einer lokalen Methode unterscheidet. Ein Remote-Object ist ein Objekt, das sich in einer anderen virtuellen Maschine als das lokale Objekt befindet. Das lokale Objekt kann die Methoden des entfernten Objektes aufrufen, das sich auch auf einem anderen Host befinden kann. Ein entferntes Objekt wird durch ein Remote Interface beschrieben, eine Java-Schnittstelle, welche die Header der Methoden des Objektes enthält. Der Aufruf einer Methode eines Remote-Objects wird Remote Method Invocation genannt. Dieser Aufruf entspricht in der Syntax exakt dem Aufruf einer lokalen Methode.

2.2 Sicherheit bei der Ausführung von Java-Anwendungen

Wird eine unbekannte Anwendung mit Hilfe eines Web-Browsers über das Internet geladen, besteht keinerlei Kenntnis darüber, zur Durchführung welcher Aktionen diese Anwendung imstande ist. Schlimmstenfalls wird das System mit einem Virus infiziert, der vielleicht die Fähigkeit besitzt die Festplatte zu formatieren. Um dieses zu verhindern ist in die virtuelle Maschine ein Sicherheitsmechanismus eingebaut. Dieser Sicherheitsmechanismus wird durch einen *Security Manager* realisiert, der Zugriffe auf alle sicherheitsrelevanten Bereiche überwacht.

Das Prinzip, allen Code vor der Ausführung zu überprüfen nennt sich *Bytecode Verifikation*. Unabhängig davon, woher der auszuführende Code geladen wird, ob von der lokalen Festplatte oder von irgendeinem Host aus dem Internet, erfolgt eine Gültigkeitsprüfung des Codes. Nach dieser Überprüfung kann der Java Interpreter den Bytecode übernehmen, ohne während

der Ausführung eine zeitintensive Prüfung durchführen zu müssen. Prinzipiell ist die Ausführung der folgenden Aktionen durch ein über das Netz geladenes Applet nicht möglich, bzw. bedarf einer Bestätigung des Benutzers:

- Laden von Bibliotheken (.dll's)
- Ausführung nativer (plattformspezifischer) Methoden
- Aufruf anderer Anwendungen
- Netzverbindungen zu anderen Hosts (mit Ausnahme des Hosts, von dem das Applet geladen wurde)
- Lesen sicherheitsrelevanter Systemeinstellungen

Lokal geladene Java-Anwendungen sind von diesen Einschränkungen nicht betroffen. Prinzipiell ist es aber auch in diesen möglich durch die Installation eines Security-Managers bestimmte Aktionen zu unterbinden.

3 Intelligentes Kommunikationsmanagement

Der Intelligente Kommunikationsmanager (IKM) hat die Aufgabe seinem Benutzer die individuelle Organisation seiner persönlichen Kommunikation zu ermöglichen, d.h. die übersichtliche Handhabung und Verwaltung aller eingehenden Kommunikationsereignisse. Der Begriff *Kommunikationsereignis* soll hier als Abstraktion für eine über ein Kommunikationsnetz empfangene Nachricht oder einen eingehenden Verbindungswunsch dienen. Folgende Kommunikationsereignisse werden vom IKM behandelt: eingehender Telefonanruf bzw. eingehendes Fax, Empfang einer Pager/SMS-Nachricht, einer Voice-Nachricht und einer Email.
Die eingehenden Kommunikationsereignisse werden individuell, in Abhängigkeit vom Anrufer/Absender und der Tageszeit behandelt.
Durch das Erstellen eines Benutzerprofils kann der Anwender die Konditionen festlegen, in welcher Form ein auftretendes Ereignis behandelt werden soll. Ein solches Profil enthält alle Parameter, die zur automatische Bearbeitung des jeweiligen Ereignisses notwendig sind.

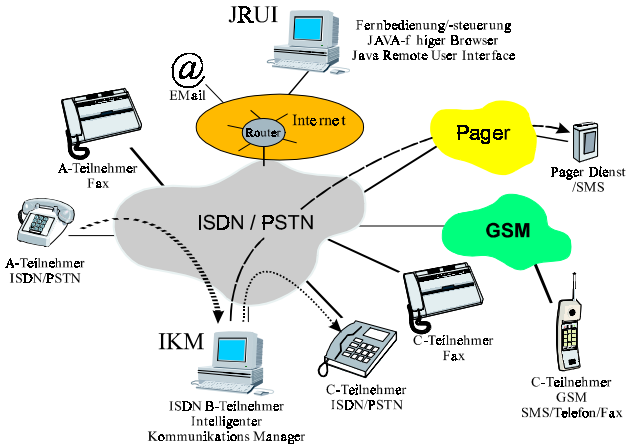

Abbildung 2: Persönliches Kommunikationsmanagement

Die Identifikation eines eingehenden Kommunikationsereignisses geschieht entweder durch die übermittelte Rufnummer (CLIP in ISDN/GSM) oder durch vereinbarte DTMF-Zahlenkombinationen (Anrufe aus dem analogen Netz). Für die spezifische Behandlung der eingehenden Emails können alle Felder des Email-Headers (Email-Adresse, Attachment, Subject ...) zur Identifikation des Absenders herangezogen werden.
Die folgenden Funktionen werden vom IKM unterstützt:

- Weiterleitung eines Anrufes/Faxes zu einem beliebigen Anschluß bzw. zu der Rufnummer, unter der der Benutzer registriert ist. Alle wichtigen Anrufe können z.B. zum Mobiltelefon des Benutzers weitergeleitet werden.
- Abspielen einer anruferspezifische Ansage und Aufzeichnen einer Sprachnachricht.
- Ablehnen eines eingehenden Anrufes/Faxes, dadurch können störende Anrufe bzw. der Empfang von Werbefaxen verhindert werden.
- Versenden einer aufgezeichneten Sprachnachricht bzw. eines empfangen Faxes an eine beliebige Emailadresse. Somit können aus der Ferne Sprach- bzw. Fax-Nachrichten ohne ein Telefon oder Fax-Gerät per Email abgerufen werden. Umgekehrt können auch eingehende Voice-Emails per Telefon abgehört werden.
- Ablehnen von bestimmten z.B mit Viren behafteten Emails.
- Weiterleitung von Emails abhängig von den Parametern des Email-Headers.
- Automatisches Beantworten von Emails mit einem absenderspezifischen Inhalt.
- Weiterleiten von Emails zu einem beliebigen Fax-Gerät. Somit können Emails auch dort empfangen werden, wo kein Internet-Anschluß vorhanden bzw. das entsprechende Endgerät verfügbar ist.

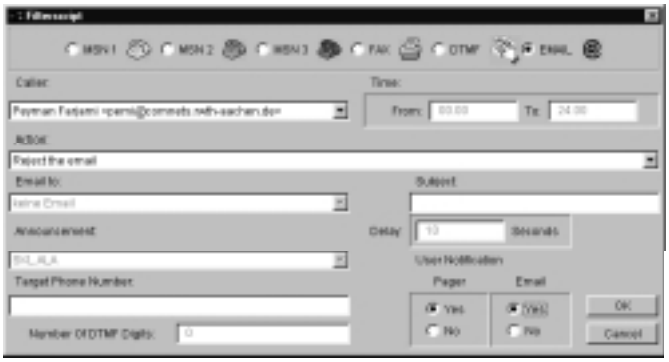

Abbildung 3: Das Java-Applet zum Erstellen eines User-Profiles

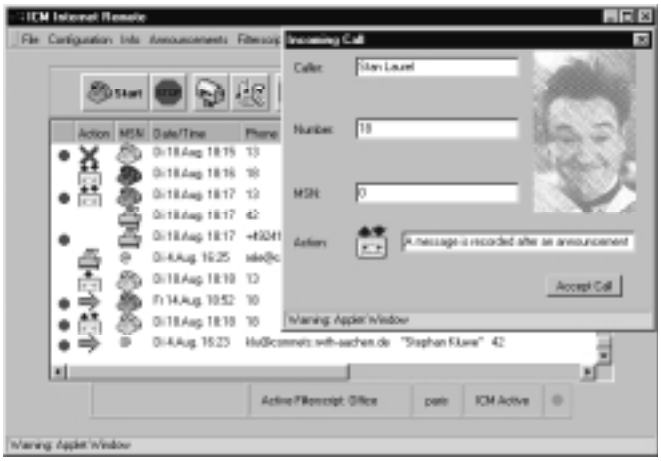

Abbildung 4: Java Remote User Interface

Der Benutzer kann über die vom IKM durchgeführten Aktivitäten per Pager/SMS-Nachrichten und/oder Email benachrichtigt werden. Somit ist der Benutzer in der Lage, auch unterwegs auf bestimmte Kommunikationsereignisse zu reagieren.

Um dem Benutzer mehr Mobilität zu ermöglichen, werden ihm verschiedene Methoden angeboten, eine Konfiguration des IKM und den Abruf eingegangener Nachrichten unabhängig vom Standort durchzuführen.

Eine Variante ist die Fernkonfiguration über ein sprachgeführtes Menü, das durch die DTMF-Detektion gesteuert wird. Diese auch von den sogenannten Service-Centern (Operator) verwendete Lösung bietet jedoch auf Grund der fehlenden graphischen Benutzeroberfläche nur eingeschränkte Möglichkeiten zur Fern-Konfiguration/Abfrage.

Eine erweiterte Möglichkeit zum Fernzugriff auf den IKM stellt eine ISDN-Datenverbindung basierend auf dem X.75 Protokoll dar. Diese Rechner-zu-Rechner-Verbindung läßt zwar weitgehend den Fernzugriff auf den IKM aus über die gewohnte graphische Benutzeroberfläche zu, jedoch ist auch die Lösung folgenden Einschränkungen unterworfen. Hierfür kann der Fernzugriff nur an einem ISDN-Anschluß erfolgen, der angeschlossene Rechner muß ein PC mit dem Betriebssystem Windows 95 sein und die Software zur Fernkonfiguration muß vorher auf dem Rechner installiert werden.

Die obengenannten Restriktionen verdeutlichen die Notwendigkeit einer plattformunabhängigen Möglichkeit zum Fernzugriff auf Kommunikationsmanagement-Dienste wie IKM.

Aus diesem Grund wurde hier eine java-basierte Lösung entwickelt, die es dem Benutzer ermöglicht, absolut unabhängig von der zur Verfügung stehenden Plattform (Kommunikationsnetz, Betriebssystem, Hard- und Software) aus der Ferne auf den IKM zuzugreifen und in der gewohnten graphischen Umgebung die angebotenen Kommunikationsdienste in Anspruch zu nehmen. Hierfür benötigt der Benutzer lediglich einen beliebigen Internet-Zugang und einen java-fähigen Web-Browser.

Der Benutzer lädt zunächst von seiner Homepage geschützt durch ein Paßwort das entsprechende Java-Applet, das ihm auf jeder beliebigen Plattform das graphische Benutzer-Interface des IKM zur Verfügung stellt. Durch dieses Applet können z.B. die aufgezeichneten Sprachnachrichten abgerufen bzw. die eingegangenen Faxe angezeigt werden. Darüber hinaus beinhaltet dieses Java-Remote-User-Interface (JRUI) eine sogenannte Multimedia-Mailbox, in der alle eingegangenen Kommunikationsereignisse und deren spezifischen Behandlung dargestellt sind. Des weiteren stellt das entsprechende Java-Applet das Interface zum anschauen bzw. zur Fernkonfiguration von server-seitig vorhandenen Benutzerprofilen bereit (User Profile Management), siehe Abbildung 3.

Ein weiteres Feature des JRUI ist die Online-Benachrichtigung des Benutzers über eingehende Kommunikationsereignisse. Im Falle eines server-seitig eingehenden Kommunikationswunsches wird dies durch das entsprechende Applet auf dem Rechner des Fernbenutzers in Form eines Popup-Fensters angezeigt. Dieses Fenster beinhaltet den Namen, die Rufnummer und wenn vorhanden das Photo des Anrufers/Absenders. Darüber hinaus wird der speziell für diesen Anrufer/Absender voreingestellte Dienst angezeigt. Hier kann der Fernbenutzer spontan entscheiden, ob er den voreingestellten Dienst durchführen läßt oder übers Internet ins Geschehen eingreift und den Anruf (oder das Fax) mit Hilfe des *Accept Call*-Buttons innerhalb dieses Applets (Popup-Fenster) zu seinem aktuellen Anschluß weiterleiten möchte, siehe Abbildung 4.

4 Realisierung

Der IKM besteht server-seitig aus zwei Hauptkomponenten, *Service Node (SN)* und *Internet Service Node (ISN)*, die sich wiederum aus mehreren Modulen zusammensetzen. Der Service Node ist bedingt durch die Telefonie-Hardware eine C++-Implementierung, die das Call-Management des IKM übernimmt, während der Internet Service Node 100% in Java entwikkelt wurde und damit das Bindeglied zwischen dem für die Handhabung der Kommunikation mit dem öffentlichen Telefonnetz zuständigen SN und dem Internet darstellt, siehe Abbildung 5.

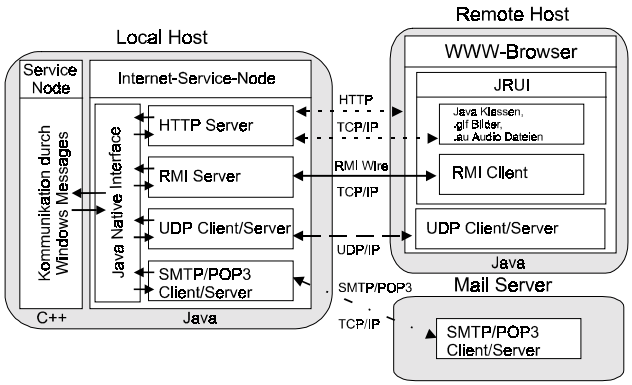

Abbildung 5: Module des IKM

4.1 Die Module des IKM-Servers

Im folgenden werden die Funktionen dieser Komponenten erläutert.

4.1.1 Der Service Node (SN)

Die Behandlung von Kommunikationsereignissen aus dem Telefonnetz wird vom Service Node durchgeführt. Der Service Node (SN) ist ein völlig eigenständiges Modul, das über das CAPI und einen ISDN-Adapter mit dem Telefonnetz verbunden ist. Der SN verhält sich im aktivierten Zustand wie ein ISDN-Endgerät. Geht ein Anruf ein, werden die über den D-Kanal signalisierten Nachrichten über die CAPI-Schnittstelle an den SN weitergeleitet, der die Signalisierung auswertet, einen passenden Eintrag aus dem aktiven Benutzerprofil auswählt und die festgelegte Aktion startet. Soll der Benutzer über ein eingegangenes Kommunikationsereignis mit Hilfe eines Pagers informiert werden, startet der SN das Pager-Modul, welches die Pager-Information daraufhin unabhängig vom SN überträgt. Ist die Weiterleitung eines Telefaxes vorgesehen oder wird ein Fax über das Sprachmenü abgerufen, startet der SN das Fax-Modul, das diese Aufgabe dann selbständig durchführt.

4.1.2 Der Internet Service Node (ISN)

Der Internet Service Node beherbergt die gesamte Funktionalität, die zur Durchführung der Internet-Dienste notwendig ist. Er besteht aus mehreren Modulen, die jeweils mit den anderen IKM Modulen interagieren.
Eine Aufgabe des ISN ist es die Nachrichten, die er vom SN erhält, auszuwerten und für den Fall, daß eine Sprach- oder Faxnachricht aufgezeichnet wurde, die Auswertung des aktiven Kommunikationsprofils zu übernehmen. Enthält dieses einen Eintrag der den Versand der eingegangenen Nachricht in Form einer Email vorsieht, führt der ISN diese Aktion durch.

Die zweite Aufgabe des ISN ist der Datenaustausch über das Internet für die Fernabfrage und Fernkonfiguration mit Hilfe des Java Remote User Interface (JRUI).

4.1.2.1 Der RMI-Server

In den ISN wurde zum Zweck der Kommunikation mit dem entfernten JRUI ein RMI-Server integriert, der für den Transport von Objekten über eine TCP/IP Verbindung zuständig ist. Diese Objekte beinhalten die Daten (wie z.B. die Einträge der Multimedia-Mailbox, oder eines Benutzerprofils), die ein Benutzer mit Hilfe des RJUI beim RMI-Server anfordern kann. Doch der ISN kann nicht nur Anfragen des RJUI beantworten, er kann dieses Modul mit Hilfe des RMI-Servers auch aktiv über aktuell eingehende Nachrichten informieren. Aktiv soll an dieser Stelle aussagen, daß die Benachrichtigung durch die Initiative des ISN erfolgt, nicht durch die periodische Anfrage des JRUI beim ISN. Auf diese Weise kann das JRUI auch über Zustandsänderungen des SN informiert werden. Dies ist der Fall, wenn der Service Node ein- oder ausgeschaltet wurde, oder die Aktivierung eines anderen Benutzerprofils erfolgt ist. Außerdem erfolgt die Anzeige eines eingehenden Anrufes und die Aktualisierung der Multimedia-Mailbox durch den ISN.

4.1.2.2 Der HTTP-Server

Ein weiteres Modul des ISN ist der HTTP-Server. Der HTTP-Server stellt dem RJUI alle Ressourcen zur Verfügung, die sich mit Hilfe des HTTP übertragen lassen. In erster Linie hat der HTTP-Server die Aufgabe, das Laden des JRUI in Form eines Applets zu ermöglichen. Mit Hilfe eines Java-fähigen Web-Browsers können die Klassen aus denen sich das Applet zusammensetzt auf den Host geladen werden, von dem die Fernkonfiguration durchgeführt werden soll. Auch für die Übertragung von eingegangenen Fax- und Sprachnachrichten in Form von GIF- und AU-Dateien wird der HTTP-Server benötigt.

4.1.2.3 Der UDP-Client/Server

Die UDP-Komponente ist nur dann erforderlich, wenn auf dem IKM-Server keine ständige Verbindung zum Internet vorhanden ist. Ist für den Zugriff auf das Internet der Aufbau einer Wählverbindung nötig, werden Datagramme zu Signalisierungszwecken verwendet. Nach dem erfolgreichen Aufbau einer Wählverbindung zum jeweiligen Internet-Service-Provider wird ein Datagramm an den Client-Host gesandt, von dem der Fernzugriff auf den IKM gestartet werden soll. Dieses Datagramm beinhaltet den Namen der HTML-Seite, die den Aufruf des JRUI-Applets enthält, sowie die IP-Adresse des IKM-Servers. Von diesem Host wird der Empfang des Datagrammes durch die Versendung eines weiteren Datagrammes an den ISN bestätigt. Hat der ISN nach Ablauf eines dort zum Zeitpunkt des Datagramm-Versandes gestarteten Timers noch keine Bestätigung erhalten, wird die Wählverbindung zum Internet-Service-Provider beendet.

4.2 Kommunikation und Signalisierung zwischen den Modulen des IKM

Der Informationsaustausch zwischen den Modulen des IKM wurde durch unterschiedliche Methoden realisiert, die im folgenden beschrieben werden.

4.2.1 Informationsaustausch zwischen C++ und Java-Modulen des IKM

Der IKM besteht serverseitig aus hardwareabhängigen C++-Modulen für das Windows-Betriebssystem und Java-Modulen, die in einer virtuellen Maschine ausgeführt werden. Daher können diese Module nicht auf direktem Wege miteinander kommunizieren. Für die Kommunikation zwischen den C++-Modulen werden sogenannte *Windows-Messages* verwendet. Diese stellen den üblichen Weg dar, um unter Windows Nachrichten zwischen Anwendungen bzw. Threads auszutauschen. Um den Versand von Windows-Messages auch von einer Java-Anwendung zu einer Windows-Anwendung zu ermöglichen, wird das JNI (Java Native Interface) verwendet, das eine Standard-Schnittstelle für Aufrufe betriebssystemabhängiger, sogenannter *nativer* Methoden darstellt. Das JNI bietet dem in einer virtuellen Maschine ausgeführten Java-Bytecode die Möglichkeit, mit Anwendungen und Laufzeitbibliotheken, die in anderen Programmiersprachen (wie C oder C++) erstellt wurden, zu kooperieren. Eine Java-Anwendung, die nativen Code ausführt kann mit Hilfe des JNI unabhängig von der Implementierung der verwendeten virtuellen Maschine erstellt werden.

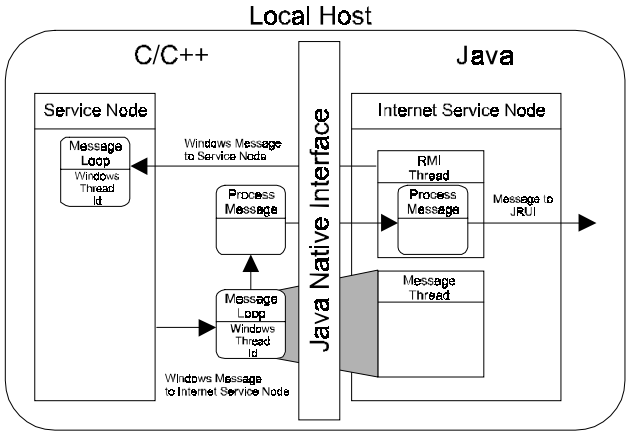

Abbildung 6: Austausch von Windows-Messages zwischen C++ und Java-Modulen

Im Gegensatz zu dem oben beschriebenen Fall (Java-Anwendung sendet Windows-Messages zur einer Windows-Anwendung) ist das Empfangen von Windows-Messages von einer Java-Anwendung nicht ohne weiteres möglich. Als Voraussetzung für den Empfang einer Windows-Message, muß eine Anwendung eine Fenster- oder Thread-Id besitzen. Da eine Java-Anwendung in der virtuellen Maschine ausgeführt wird und aus diesem Grund keine Thread-Id besitzt, kann sie auch keine Windows-Messages empfangen. Zur Lösung dieses Problems wurde ein Mechanismus entwickelt, der es erlaubt Windows-Messages an eine Java-Anwendung zu versenden. Dies ist in Abbildung 6 schematisch dargestellt. Hier wird zunächst über eine native Java-Klasse ein Thread gestartet (Message-Thread), der eine bestimmte Thread-Id besitzt. Dieser Thread ruft die native Message-Loop auf, die auf eingehende Windows-Messages wartet. Die an diese Thread-Id gesendeten Windows-Messages können dann durch die Methode *ProcessMessage* an die entsprechenden Java-Klassen
weitergeleitet werden.

4.2.2 Kommunikation zwischen lokalen (Server) und entfernten (Client) IKM-Modulen

Um das JRUI als Instrument zur Fernabfrage eingegangener Sprach- und Faxnachrichten und zur Fernkonfiguration sämtlicher Einstellungen des IKM verwenden zu können, ist eine Möglichkeit zur Anfrage und zur Übertragung dieser Nachrichten und Daten über das Internet erforderlich. Ebenso müssen die Daten auf der Seite des IKM-Servers aktualisiert werden können, wenn mit Hife des JRUI Konfigurationseinstellungen geändert, oder Einträge eines Benutzerprofils editiert oder neue hinzugefügt wurden.
Zur Realisierung dieser Kommunikation über das Internet wurden verschiedene Varianten in Erwägung gezogen. Eine Möglichkeit stellt die Übertragung der

Information mit Hilfe der Protokolle TCP oder UDP dar. Dieses hätte jedoch die Entwicklung eines Anwendungsschicht-Protokolles erfordert, das die Übertragung der die Daten enthaltenden Objekte durchführt. Auf der Server-Seite müssen die Objekte so konvertiert werden, daß sie durch TCP (datenstromorientiert) oder UDP (paketorientiert) transportiert werden können. Nach dem Transfer ist auf der Seite des Clients (JRUI) die Zusammensetzung der übertragenen Objekte erforderlich.
Da hier zwischen den Instanzen ISN und (JRUI) Objekte ausgetauscht werden sollen, bietet sich alternativ die Nutzung einer Architektur zur Kommunikation verteilter Objekte an.
Ein möglicher Einsatz ist CORBA (Common Object Request Broker Architecture), eine Architektur, die eine Kommunikation zwischen über ein Netz verteilten Objekten möglich macht. Das besondere an CORBA ist, daß es eine Integrationstechnologie und keine Technologie zur Programmierung verteilter Anwendungen ist. Integrationstechnologie bedeutet, daß die miteinander ausgetauschten und kommunizierenden Objekte in verschiedenen Programmiersprachen implementiert sein können. CORBA realisiert die von einer Programmiersprache unabhängige Beschreibung von Objekten. Der Client und der Server benötigen lediglich Instanzen, die die Objekte von den lokalen Beschreibung in die „CORBA-Schablone“ und umgekehrt konvertieren. Auf der Seite des Clients wird dieser Dienst durch die als *Stub* bezeichnete Schnittstelle erbracht. Serverseitig befindet sich die *Skeleton*-Schnittstelle. Die Beschreibung der Schnittstellen erfolgt durch die IDL (Interface Description Language), die in der Syntax C bzw. C++ sehr ähnelt. Für die Identifizierung und Lokalisierung von Objekten, sowie den Verbindungsauf- und abbau und die Übertragung ist der ORB (Object Request Broker) zuständig.
Eine Alternative zu CORBA stellt die RMI dar. Im Gegensatz zu CORBA ist sie jedoch auf die Kommunikation verteilter Java-Objekte beschränkt. Der Vorteil von RMI ist die nahtlose Integration dieser Technologie in die Programmiersprache Java. Der Aufruf von Methoden eines Objektes, das sich auf einem anderen Host befindet ist identisch mit dem Aufruf eines lokalen Objektes. Das hohe Maß an Übersichtlichkeit und Konsistenz, das bei der Entwicklung einer verteilten Anwendung mit Hilfe von RMI zu erreichen ist, war ausschlaggebend für den Einsatz dieser Architektur für die Kommunikation zwischen ISN und JRUI. Darüber hinaus besteht für die Kommunikation zwischen den Komponenten ISN und JRUI keine Notwendigkeit offene Schnittstellen anzubieten, da beides vollständig in Java implementierte Module sind.

5 Ausblick

Eine weitere Möglichkeit zur Nutzung des Internet als Kommunikationsmedium brächte die Integration der momentan viel diskutierten Internet-Telefonie in die Dienste des IKM mit sich. Der IKM könnte die Weiterleitung eingehender Sprachanrufe zu einem Teilnehmer im Internet übernehmen, genauso wie es möglich wäre ein Gespräch aus dem Internet an einen Teilnehmer im öffentlichen Telefonnetz weiterzuleiten. Des weiteren kann der IKM um eine Datenbankabfrage per SMS erweitert werden. Der SMS bietet aufgrund seiner gesicherten Übertragung ein geeignetes Medium zur Information des Benutzers über wichtige Ereignisse und zur kurzen Abfrage von Datenbanken. Dieser Dienst bietet dem Benutzer die Möglichkeit zur Abfrage von z.B. Adresse, Telefon/Fax/Mobiltelefon-Nummer, Emailadresse, Geburtstagen u.ä. der in der persönlichen Datenbank des Benutzers vorhandenen Personen. Die entsprechenden Abfragen werden in Form von SMS-Nachrichten vom mobilen Endgerät versandt. Sie werden auf dem Server entsprechend bearbeitet und die Ergebnisse der Abfragen werden wieder als SMS-Nachrichten an den Benutzer zurückgeschickt. Über eine persönliche Datenbank hinaus, kann dieser Dienst unter Benutzung von Agententechnologie mit einem Internet-basierten Informationsrecherche-System kombiniert werden. Eine SMS-Abfrage würde dann in einem an IKM angeschlossenen Agenten-system einen mobilen Agenten beauftragen, der selbständig eine Suche nach bestimmten Informationen in unterschiedlichen Datenbanken startet. Der Einsatz von mobilen Agenten kann weiterhin dazu benutzt werden, um einen Fernzugriff auf Kommunikationsdienste nicht nur plattformunabhängig zu erlauben, sondern auch unabhängig vom benutzten Endgerät (Laptop, Communicator, PDA, Mobiltelefon ...) dem Fernbenutzer die gewohnte Benutzerumgebung zur Verfügung zu stellen, das sogenannte VHE (Virtual Home Environment), Ein Konzept, das ein Bestandteil von UMTS ist. Durch dieses Konzept wird einem UMTS-Teilnehmer unabhängig von dem Kommunikationsnetz oder dem Ort, in dem er sich gerade aufhält und unabhängig von dem von ihm benutzten Endgerät, die gewohnten persönlichen Dienste, Features und Benutzerschnittstelle zur Verfügung gestellt. Dieser Ansatz wird im Rahmen des ACTS-Projektes *CAMELEON* unter Benutzung von Agententechnologie verfolgt und untersucht.

6 Abkürzungen

ACTS Advanced Communications Technologies & Services

CAMELEON	Communication Agents for Mobility Enhancements in a Logical Environment of Open Networks
CAPI	Common-ISDN Application Programming Interface
CORBA	Common Object Request Broker Architecture
DTMF	Dual Tone Multi Frequency
GUI	Graphical User Interface
IDL	Interface Description Language
IKM	Intelligenter Kommunikations-manager
ISN	Internet Service Node
JNI	Java Native Interface
JRUI	Java Remote User Interface
RPC	Remote Procedure Call
RMI	Remote Method Invocation
SN	Service Node
SMS	Short Message Service
UMTS	Universal Mobile Telecommunication Systems
VHE	Virtual Home Environment
VM	Virtuelle Maschine

7 Literatur

[1] A. Badach. ISDN und CAPI. VDE-Verlag, Berlin, Germany, 1993.

[2] T. Berners-Lee et. Al. Hypertext transfer Protocol - HTTP/1.0. IETF, May 1996.

[3] P. Farjami. Intelligneter Kommunikationsmanager am ISDN-Basisanschluß: Konzeption und Realisierung eines durch Smart Card geschützten Zugangsprotokolls. Diplomarbeit, RWTH Aachen, Lehrstuhl für Kommunikationsnetze, Oktober 1996.

[4] E. R. Harold. Java Network Programming. O'Reilly & Associates, Inc., Sebastopol, USA, 1997

[5] F. Reif. Intelligneter Kommunikationsmanager am ISDN-Basisanschluß: Konzeption und Realisierung eines CAPI 2.0 Protokollautomaten für erweiterte UPT-Dienste. Diplomarbeit, RWTH Aachen, Lehrstuhl für Kommunikationsnetze, Oktober 1996.

[6] SUN Microsystems Inc. Java Remote Method Invocation Specification, 1997.

[7] S. Kluwe. Intelligneter Kommunikationsmanager am ISDN-Basisanschluß: Konzeption und Realisierung eines plattformunabhängigen Zugangsprotokolls über das Internet. Diplomarbeit, RWTH Aachen, Lehrstuhl für Kommunikationsnetze, November 1997.